REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA



Faculté de Technologie

Département du Génie Electrique

Mémoire de fin d'études de Master

Filière : Télécommunications

Spécialité : Systèmes des Télécommunications

Implémentation d'un IP CORE pour un réseau de neurones sur une plateforme FPGA

Présenté par :

Mr. RAMLI Anes

Mr. REBOUH Dahmane

Proposé et Dirigé par :

Dr. BENGHERBIA Billel

Mr. KARA Redha

Promotion 2019

Dédicaces

Je dédie ce modeste travail. Tout d'abord je tiens à remercier Dieu le tout puissant de m'avoir donné la santé, la volonté et le courage.

Ensuite, à mon père et ma mère pour leur soutien et leurs encouragements et qu'aucune dédicace ne pourra compenser leur sacrifice pour mon bien être et mon bonheur.

A mes chers frères Amine, Wael et mouad pour lequel j'ai beaucoup d'estimes.

A mes sœur Nesrine et Hanene que je lui souhaite une longue vie pleine de joie et de réussite.

A mes oncles, mes tantes et toute ma famille.

A mes amis, en particulier mon collègue du présent projet REBOUH Dahmane qui n'a jamais manqué de travailler et ceux de la promotion de la 2ème année master systèmes de télécommunications.

A notre encadreur Dr.BENGHERBIA Billel. Et à tous ceux qui nous ont soutenus pour l'achèvement de ce projet avec réussite.

RAMLI Anes

Dédicaces

Je dédie ce modeste travail. Tout d'abord je tiens à remercier Dieu le tout puissant de m'avoir donné la santé, la volonté et le courage.

A la mémoire de mon père Puisse Dieu, le tout puissant, l'avoir en sa sainte miséricorde.

Ensuite, à ma mère pour leur soutien, leurs prières, et leur encouragement et qu'aucune dédicace ne pourra compenser leur sacrifice pour mon bien être et mon bonheur.

A mes chers frères Hichem et Khaled pour lequel j'ai beaucoup d'estimes.

A mes amis, ma famille, en particulier mon collègue du présent projet RAMLI Anes qui n'a jamais manqué de travailler et ceux de la promotion de la 2ème année master systèmes de télécommunications.

A notre encadreur Dr.BENGHERBIA Billel, a notre co-encadreur Mr.KARA Redha, Et à tous ceux qui nous ont soutenus pour l'achèvement de ce projet avec réussite.

REBOUH Dahmane

Remerciements

Avant tout nous remercions ALLAH de nous avoir accordé des connaissances de la science et de nous avoir aidés à réaliser ce travail.

Nous adressons aussi nos remerciements à nos familles pour tout l'aide, qu'elles nous ont apportées tout le long de nos études.

Nous tenons à remercier chaleureusement et respectueusement notre promoteur Dr. Bengherbia billel et notre Co promoteur Mr. Kara pour son aide et son soutien tout au long de notre projet.

Nous remercions aussi tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire de fin d'étude, sans oublier d'adresser nos remerciements à tous les enseignants du département Génie électrique notamment la spécialité télécommunication.

Nos vifs remerciements aux membres de jury pour nous avoir fait l'honneur et le plaisir d'accepter de juger notre travail.

Anes et Dahmane

الملخص

في هذا العمل، قمنا بتطوير و تنفيذ شبكة عصبية اصطناعية من نوع ADALINE « شبكة خطية تكيفية » على بطاقة FPGA من فئة « Artix A7-100T » من Xilinx مناسبة لنظام يعمل على كشف ترددات و سعات الاشارات التوافقية، هذه الشبكة العصبية نُفِّذت على شكل IP-Core (نواة ذات ملكية فكرية) مخصصة، يديرها معالج MicroBlaze. مع ضمان أنَّ تدرب الشبكة العصبية سيكون داخل الدارة FPGA.

الكلمات المفتاحية : شبكة عصبية اصطناعية، شبكة خطية توافقية، FPGA، الاشارات التوافقية، نواة ذات ملكية فكرية، تعلُّم.

Résumé

Dans le présent travail, nous avons développé et implémenté un réseau de neurones artificiel de type ADALINE « Réseau d'Adaptation Linéaire » à base d'une carte FPGA de type « Artix A7-100T » de Xilinx. Pour un système qui sert à détecter les fréquences et les amplitudes des signaux harmoniques. Ce réseau de neurones est implémenté comme un IP-Core (noyau de propriété intellectuelle) personnalisé, géré par un processeur soft-core MicroBlaze, en assurant que l'apprentissage de réseau de neurones sera au sein du circuit FPGA.

Mots clés : réseau de neurones artificiel, réseau d'adaptation linéaire, FPGA, les signaux harmoniques, IP-Core, apprentissage.

ABSTRACT

In the present work, we have developed and implemented an artificial neural network of type ADALINE « Adaptive Linear Element » based on Xilinx's « Artix A7-100T » FPGA carte. For a system that detects frequencies and amplitudes of harmonic signals. This neural network is implemented as a custom IP-Core (intellectual property), managed by a soft-core MicroBlaze processor. ensuring that the neural network's learning will be within the FPGA circuit.

Key words: artificial neural network, adaptive linear element, FPGA, harmonic signals, IP-Core, learning.

TABLE DES MATIERS

LISTE DES FIGURES I

LISTE DES TABLEAUX II

LISTE DES ACRONYMES III

Introduction générale 1
CHAPITRE I : LES RESEAUX DE NEURONES ARTIFICIELS 3
1. Introduction 4
2. Le neurone 4
3. Bases Biologiques 5
4. Neurone formel (Artificiel) 6
4.1. Définition 6
4.2. Les paramètres 6
5. Principe de fonctionnement 7
6. Architecture des réseaux de neurones 8
6.1. Réseaux statiques 8
6.2. Réseaux dynamiques 8
6.3. Réseaux auto-organisés 9
7. Apprentissage des neurones artificiels 9
7.1. Définition 9
7.2. Apprentissage supervisé 9
7.3. Apprentissage non supervisé 10
8. Perceptron 10
8.1. Perceptron mono couche 10
8.2. Perceptron multicouche 10
8.2.1. Rétro propagation du gradient pour un Perceptron Multi–Couches : 12
8.2.2. Mise à jour des poids : 12
9. Modèle Adaline 13
9.1. Prédiction du signal avec Adaline 14
9.1.1. Création d'un réseau Adaline sous Matlab avec toolbox 14
9.1.2. Création d'un réseau Adaline sous Matlab sans toolbox : 15
10. Conclusion : 17
CHAPITRE II : LES CIRUITS PROGRAMMABLES FPGA 18
1. Introduction 19
2. Les circuits programmables FPGA 19
2.1. Définition 19
3. Structure interne 21
3.1. Logique des FPGA 22
3.2. Interconnexion 23

3.3. Entrée et sortie 25
3.4. Horloge 26
3.5. Configuration 27
3.6. Consommation D'énergie 28
4. Les Familles et caractéristiques des FPGA 29
4.1. Xilinx 29
5. MicroBlaze 30
5.1. Définition 30
5.2. Caractéristiques 31
5.3. Bus et protocole AXI 31
6. Conclusion 32
CHAPITRE III : CONCEPTION ET IMPLEMENTATION FPGA 33
1. Introduction 34
2. Les phases d'étude et d'implémentation 34
3. Synthèse de RNA et simulation sur Matlab 35
3.1. Organigramme d'apprentissage 35
3.2. Résultats d'essai de réseau 377
4. System Generator 38
4.1. Méthodologie de conception pour la mise en œuvre sur FPGA avec Xilinx System Generator 38
4.2. Blocksets de System Generator 40
5. Synthèse de RNA et simulation sur System Generator 42
5.1. La modélisation de RNA sous Simulink 42
5.2. Génération du bloc de Co-Simulation 455
5.3. Création de l'IP-Core 488
6. La carte Nexys A7-100T 499
6.1. Définition 499
6.2. Caractéristiques 499
7. Vivado Design Suite 522
8. L'implémentation sur la carte FPGA 533
8.1. Diagramme des blocs 533
8.2. Résultats d'implémentation matérielle de réseau de neurone 555
9. Configuration FPGA 555
9.1. Bitstream 566
9.2. Le logiciel SDK 577
9.3. Configuration JTAG 577
9.4. L'organigramme d'exécution sur la carte 588
10. Test de résultats dans la carte 60
10.1. La configuration sur SDK 60
10.2. Résultats d'exécution sut la carte 60
10.2.1. Discutions de résultats obtenue 62
11. Conclusion 63

Conclusion générale 64

Références bibliographiques : IV

LISTE DES FIGURES

Figure I.1. *Représentation d'un neurone réel.* **5**
Figure I.2. *Modèle de base d'un neurone formel.* **7**
Figure I.3. *Le Perceptron : structure et comportement* **10**
Figure I.4. *Modèle du Perceptron Multi–Couches.* **11**
Figure I.5. *Schéma de principe De l'Adaline* **13**
Figure I.6. *Script Matlab utilisé 1* **14**
Figure I.7. *Signal fourni en apprentissage au réseau Adaline* **14**
Figure I.8. *Erreur entre le signal initial et le signal prédit par le réseau de neurones* **15**
Figure I.9. *Script Matlab utilisé 2* **15**
Figure I.10. *Signal fourni en apprentissage au réseau Adaline* **16**
Figure I.11. *Erreur entre le signal initial et le signal prédit par le réseau de neurones* **16**
Figure II.1 *Structure interne du FPGA* **20**
Figure II.2 *Structure interne simplifiée du FPGA* **21**
Figure II.3. *Une cellule logique est la pierre de fondation d'un FPGA* **23**
Figure II.4. *Bloc d'entrée / sortie générique* **25**
Figure II.5. *Réseaux de distribution d'horloge.* **27**
Figure II.6. *Configuration de FPGA basée sur la RAM statique* **28**
Figure II.7 *Inverseur CMOS.* **29**
Figure II.8. *Schéma fonctionnel de base de MicroBlaze* **30**
Figure III.1. *Phases d'implémentation sur FPGA* **34**
Figure III.2. *Structure du réseau* **35**
Figure III.3. *L'organigramme d'apprentissage* **36**
Figure III.4. *La courbe d'apprentissage* **37**
Figure III.5. *La courbe d'erreur* **37**
Figure III.6. *Conceptions de Flux du System Generator* **39**
Figure III.7. *Environnement de Co-simulation matériel-logiciel basé sur FPGA* **40**
Figure III.8. *La liste des blocksets de System Generator* **40**
Figure III.9. *La liste des blocs : éléments de base* **41**
Figure III.10. *Réseau de neurones artificiels sous Xilinx System Generator* **433**
Figure III.11. *Sous-système 'Nerf'* **444**
Figure III.12. *Sous-système 'Adder'* **444**
Figure III.13. *Configuration de Co-Simulation.* **455**
Figure III.14. *Bloc de Co-Simulation* **466**
Figure III.15. *Courbe d'erreur: (a) System Generator (b) Co-Simulation.* **477**
Figure III.16. *Configuration d'interface AXI* **488**
Figure III.17. *Configuration d'IP-Core.* **499**
Figure III.18. *Légende de la fonction Nexys A7-100T* **51**
Figure III.19. *Diagramme des blocs de réseau de neurones* **544**
Figure III.20. *Options de configuration du Nexys A7 DDR* **566**
Figure III.21. *Organigramme d'exécution sur IP-Core et MicroBlaze* **599**

Figure III.22. *Chargement de bitstream sur la carte* **60**
Figure III.23. *Graphe de signal cible* **61**
Figure III.24. *Spectre de signal cible* **61**
Figure III.25. *Résultats d'exécution sur la carte* **62**

LISTE DES TABLEAUX

Tableau III.1. *Les composants de la carte Nexys A7-100T* 51
Tableau III.2. *Résultats de synthèse et d'implémentation de réseau de neurones* 55

LISTE DES ACRONYMES

ADALINE: Adaptive Linear Element

AXI: Advanced eXtesible Interface

CLB: Configurable Logical Bloc

CMOS: Complementary Metal Oxide Semiconductor

DSP: Digital Signal Processing

FPGA: Field Programmable Gate Array

FPU: Floating Point Unit

HDL: Hardware Description Language

IC: Integrated Circuit

IO: In Out

IP: Intellectual Property

ISE: Integrated Synthesis Environment

JTAG: Joint Test Action Group

LED: light emitting diode

LUT: Look Up Table

MAC: Media Access Control

MLP: MultiLayer Perceptron

MSPS: Mega Sample Per second

PCB: Printed Circuit Board

PCI: Peripheral Component Interconnect

PDM pulse density modulation

PDM: Pulse Density Modulation

PLL: Phase Locked Loop

Pmod: Peripheral module

PWM: pulse wave modulation

PWM: Pulse Width Modulation

RAM: Random Access Memory

RISC: Reduced Instruction Set Computing

RNA: Réseaux de Neurone Artificiel

RTL: Register Transfert Level

RVB: Rouge Vert Bleu

SoC: System on a Ship

SPI: Serial Peripheral Interface

SRAM: Static Random-Access Memory

UART: Universal Asynchronous Receiver/Transmitter

USB: Universal Serial Bus

VGA: video graphics array

VHDL: VHSIC Hardware Description Language

XSG: Xilinx System Generator

Introduction générale

Dans la vie courante il existe plusieurs technologies ont permis aux scientifiques d'élaborer et de perfectionner des méthodes pour différents domaines, dans ces méthodes il existe une méthode qui est utilisée dans plusieurs domaines de recherches et de différentes manières, cette méthode est celle des réseaux de neurones artificiels (RNA).

L'utilisation des réseaux de neurone artificiels est plus en plus demandée en raison de leur capacité de répondre aux plusieurs exigences dans les divers domaines.

Les Réseaux de Neurones Artificiel (RNA) sont utilisables à implémenter sur les circuits électroniques à savoir : les FPGA (Field programmable Gate Array), les ASIC (Application Specific Integrated Circuits) ou sur les DSP (Digital Signal Processing).

Les circuits programmables FPGA (Field Programable Gate Arrays) sont des ICs numériques qui englobent des blocs logiques configurables ainsi que des interconnexions configurables entre ces blocs. Les concepteurs peuvent configurer de tels périphériques pour réaliser une grande variété de circuits. Les FPGA offrent la possibilité de concevoir des architectures parallèles puissantes qui peuvent réduire considérablement le temp d'exécution de l'algorithme implémentée. Dans le cas des systèmes embarques le développement d'une bibliothèque de modules IP basée sur FPGA est très pratique. Ceci assure la portabilité des modules correspondant entre les cibles différentes. Les FPGA appartiennent à la classe des architectures reconfigurables qui offrent un compromis entre la flexibilité et l'efficacité.

La propriété intellectuelle (PI) fait référence à des fonctions logiques préconfigurées qui peuvent être utilisées dans votre conception. Xilinx fournit une large sélection d'IP optimisée pour les FPGA Xilinx. Celles-ci peuvent inclure des fonctions fournies via plusieurs manieres tel que System Generator. Xilinx et ses sociétés partenaires produisent des IP d'une complexité allant des simples opérateurs arithmétiques et des éléments de retard aux blocs de construction complexes au niveau du système, tels que les filtres DSP (Digital Signal Processing), les multiplexeurs, les transformateurs et la mémoire.

Dans ce cadre que s'insère notre projet, qui se base sur l'implémentation d'un IP-Core pour un réseau de neurones sur une plateforme FPGA.

Le premier chapitre présente quelques définitions sur les réseaux de neurone, nous exposons ces différents principes de fonctionnement et ces architectures, ces types d'apprentissage et les réseaux de neurones les plus utilisés, nous pourrons enfin savoir comment les créer et manipuler avec Matlab, de sorte que : comment créer le réseau, comment l'entraîner, et enfin, comment l'utiliser. (Avec toolbox, et sans toolbox).

Le second chapitre dédié au circuits programmables FPGA, la logique et la structure de base, leurs composants essentiels et leurs Familles et caractéristiques, sa configuration Xilinx, dernièrement le processeur Micro Blaze et ces caractéristiques.

Nous abordons dans le troisième chapitre, et en matière d'applications de cette thèse, nous concevons un réseau de neurones de type ADALINE sur System Generator. Ce dernier permet de générer un IP-Core. Cet IP-Core sera géré par un processeur MicroBlaze dans une architecture embarquée qui sera implémenté sur une carte FPGA de type Nexys A7-100T.

Enfin, nous finissons notre travail par une conclusion générale, résumons notre travail.

CHAPITRE I : Les Réseaux de Neurones Artificiels

1. Introduction

Dans le monde qui nous entoure, il existe plusieurs technologies pour développer et perfectionner des méthodes de traitements et d'analyse telles que prédictions météorologiques, prévision de la consommation d'eau, reconnaissance du code postal, compression de données, filtrage, classification…etc. Parmi ces techniques il existe celles qui utilisent les réseaux de neurones artificiels (RNA).

Les réseaux de neurones artificiels sont présentés comme la vague du futur en informatique. Ce sont en effet des mécanismes d'autoapprentissage qui ne nécessitent pas les compétences traditionnelles d'un programmeur.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau. [1]

Dans ce chapitre nous débuterons par une brève introduction, suivie du détail du modèle du neurone biologique et du neurone formel. Cette partie sera suivie de quelques définitions de réseaux de neurones artificiels. Ensuite, nous étudierons les différentes architectures qui différencient ces réseaux, les types d'apprentissage les plus utilisés (perceptron multicouche (PMC)). Pour expliquer le fonctionnement de ces algorithmes, nous utiliserons quelques exemples avec Matlab (avec les 2 méthodes : toolbox et script). Ces derniers nous permettrons d'illustrer les propriétés, les méthodes d'apprentissages et l'influence des architectures sur le calcul de la réponse du réseau.

2. Le neurone

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone (unique) pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms (10-9 m) entre l'axone du

Neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la synapse. [1]

3. Bases Biologiques

Le neurone est le processeur élémentaire du cerveau. Chaque neurone traite l'information qui lui parvient localement, puis transmet aux autres neurones qui lui sont connectés l'information qu'il a traitée. Ces cellules peuvent apprendre en changeant l'intensité de leurs connexions avec d'autres cellules ou détruire ou même créer de nouvelles connexions. Elles peuvent aussi changer leur(s) règle(s) de traitement de l'information. Ce processus de changement est appelé apprentissage et joue un rôle fondamental dans le comportement du neurone. [2]

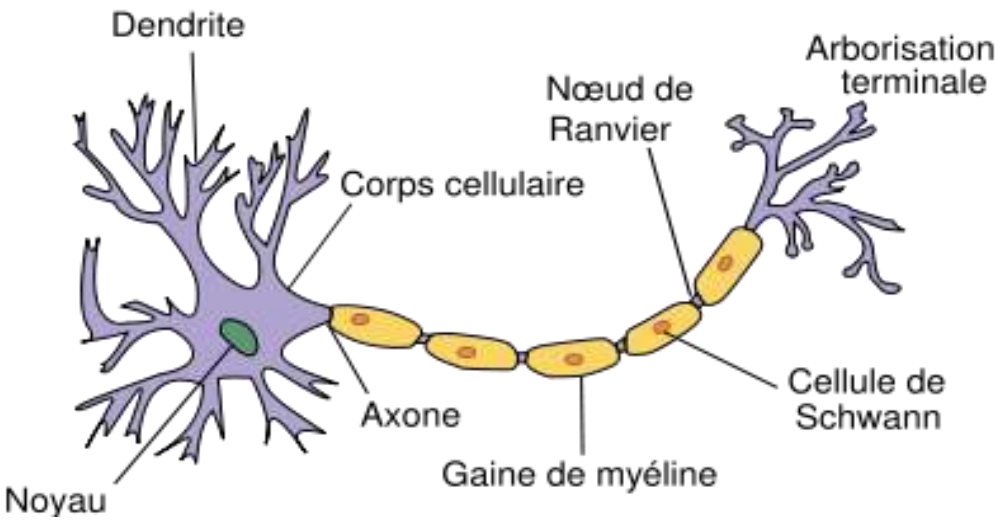


Figure I.1. *Représentation d'un neurone réel.*

La figure 1 montre la représentation graphique d'un neurone. D'après la figure 1, vous pouvez constater que le neurone est composé des parties suivantes :

- *Corps cellulaire* **:** Noyau du neurone.
- *Dendrites :* Récepteurs principaux du neurone.
- *Axone :* Fibre nerveuse de transport pour les signaux émis par le neurone.
- *Synapse :* Connexion entre la dendrite et l'axone de deux neurones [2]

4. Neurone formel (Artificiel)

4.1. Définition

Neurone formel est un processeur élémentaire qui réalise une somme pondérée des signaux qui lui parviennent. La valeur de cette sommation est comparée à un seuil et la sortie du neurone est une fonction non linéaire. [3]

Un neurone est une fonction paramétrée non linéaire de ses variables d'entrée. Naturellement, un réseau de neurones est la composition des fonctions non linéaires de deux neurones ou plus. Les variables du neurone sont souvent appelées entrées du neurone et sa valeur est sa sortie.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit [1] [4]

4.2. Les paramètres

Les paramètres sont affectés aux entrées des neurones ; la sortie du neurone est une combinaison non linéaire des entrées {xi}, pondérées par les paramètres {Wi}, souvent appelées poids.

Le potentiel v le plus fréquemment utilisé est la somme pondérée des entrées, avec un terme constant supplémentaire appelé «biais», [4]

Soient :

- d = » la dimension de l'espace des entrées.

- Pi = » un vecteur d'entrée.

- w = » un vecteur de poids.

- b = » le biais.

- v = » un coefficient (< 0.05).

- e = » l’erreur.

On peut définir la sortie du réseau ainsi :

$$Y(pi) = \sum_{j=1}^{d}(Wj \cdot Pi, j) + b = W \cdot Pi + b \quad (1.1)$$

L'erreur=la cible(n)-la sortie(n)

Les formules à appliquer pour mettre à jour les poids et biais lorsque l'on entre un individu dans le réseau sont donc :

$$W\ new = W\ old + 2 \cdot \nu \cdot e(Pi) \cdot Pi \quad (1.2)$$

$$B\ new = B\ old + 2 \cdot \nu \cdot e(Pi) \quad (1.3)$$

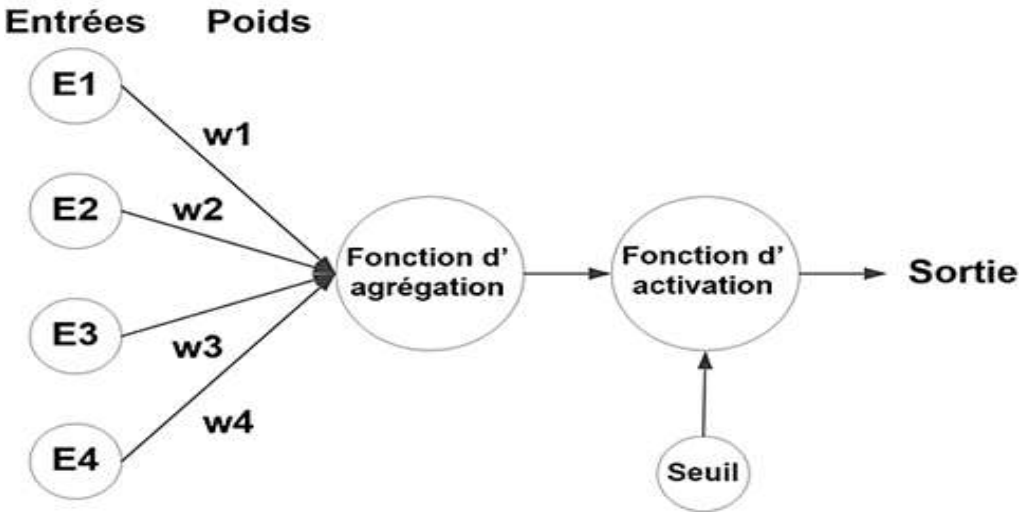


***Figure I.2**. Modèle de base d'un neurone formel.*

Nous pouvons définir un neurone par les trois éléments suivants [2] :

- La fonction d'entrée totale qui définit le prétraitement effectué sur les entrées.
- La fonction d'activation (ou d'état) du neurone qui définit son état interne en fonction de son entrée totale.
- La fonction de sortie qui calcule la sortie du neurone en fonction de son état d'activation.

5. Principe de fonctionnement

Un neurone reçoit des entrées et fournit une sortie, grâce à différentes caractéristiques :

- Des poids accordés à chacune des entrées, permettant de modifier l'importance de certaines par rapport aux autres.

- Une fonction d'agrégation, qui permet de calculer une unique valeur à partir des entrées et des poids correspondants.
- Un seuil (ou biais), permettant d'indiquer quand le neurone doit agir.
- Une fonction d'activation, qui associe à chaque valeur agrégée une unique valeur de sortie dépendant du seuil. [5]

6. Architecture des réseaux de neurones

L'architecture d'un réseau de neurones définit son fonctionnement et joue un rôle important dans son comportement. Elle est en fonction de ses couches et de la structure des connexions de ses neurones ; ces paramètres permettent de distinguer les différentes classes et/ou types d'architecture neuronales. [6]

Donc Il existe trois grands types d'architectures de réseaux de neurones :

6.1. Réseaux statiques

Ce type de réseaux est organisé généralement en couches de neurones. Chaque neurone d'une couche reçoit ses entrées à partir des neurones de la couche précédente ou tout simplement de l'entrée du réseau. Dans tels réseaux il n'existe pas de feed-back (boucles de retour d'informations). Le traitement des données se fait en sens unique et le flux d'informations circule directement de la couche d'entrée à la couche de sortie ; le traitement est donc réalisé en boucle ouverte. Ces réseaux peuvent être utilisés pour les problèmes de classification ou d'approximation des fonctions.

6.2. Réseaux dynamiques

Les réseaux dynamiques ou bien récurrents, sont les réseaux pouvant comporter des boucles (feed-back) entre les neurones. En général, la sortie de chaque neurone peut être envoyée vers l'entrée de tous les autres neurones du réseau. Ainsi, ces boucles ramènent l'information en provenance de la couche de sortie sur la couche d'entrée simultanément avec le signal d'entrée présent au même instant.

Un réseau dynamique peut donner une sortie différente en lui présentant la même entrée à des instants différents, contrairement aux réseaux statiques qui donnent la même sortie pour une

même entrée. Au début, ces réseaux étaient souvent utilisés pour les problèmes de classification et de mémorisation. Actuellement, dans plusieurs travaux la structure interne dynamique de ces réseaux est exploitée pour l'identification ou la commande des systèmes dynamiques.

6.3. Réseaux auto-organisés

Les réseaux de neurones auto-organisés sont des réseaux qui changent leurs structures internes pendant l'apprentissage. Ainsi les neurones se regroupent typologiquement suivant la représentation des exemples. Ces réseaux sont des dérivées des modèles de Kohonen. [6]

7. Apprentissage des neurones artificiels

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. Elle ne concerne cependant pas tous les modèles, mais les plus utilisés.

Une fois qu'un réseau a été structuré pour une application particulière, ce réseau est prêt à être formé. Pour démarrer ce processus, les poids initiaux sont choisis de manière aléatoire. Ensuite, l'apprentissage commence.

7.1. Définition

L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider à priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issu de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation.

Au niveau des algorithmes d'apprentissage, Il existe deux approches d'apprentissage :

7.2. Apprentissage supervisé

Apprentissage supervisé implique un mécanisme fournissant au réseau le résultat souhaité, soit en "évaluant" manuellement les performances du réseau, soit en fournissant les résultats souhaités avec les entrées. La grande majorité des réseaux utilisent l'apprentissage supervisée.

Les coefficients synaptiques sont évalués en minimisant l'erreur (entre sortie souhaitée et sortie obtenue) sur une base d'apprentissage. [7]

7.3. Apprentissage non supervisé

Apprentissage non supervisé est l'endroit où le réseau doit donner un sens aux entrées sans aide extérieure, il est utilisé pour effectuer une caractérisation initiale des entrées.

On ne dispose pas de base d'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales. [7]

8. Perceptron

8.1. Perceptron mono couche

Avant d'aborder le comportement collectif d'un ensemble de neurones, nous allons présenter le Perceptron (un seul neurone) en phase d'utilisation. L'apprentissage ayant été réalisé, les poids sont fixes. Le neurone de la figure I.3 réalise une simple somme pondérée de ses entrées, compare une valeur de seuil, et fourni une réponse binaire en sortie.

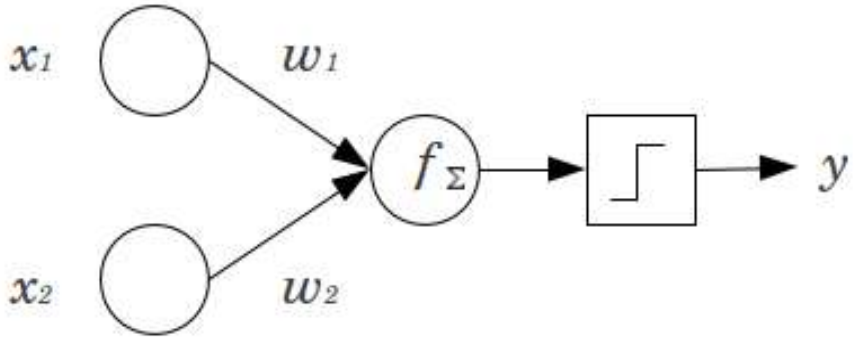


***Figure I.3**. Le Perceptron : structure et comportement. (Les connexions des deux entrées x1 et x2 au neurone sont pondérées par les poids w1 et w2)*

8.2. Perceptron multicouche

D'une façon générale, un perceptron multicouche (Multi Layer Perceptron) contient une couche d'entrée, une ou plusieurs couches cachées, chargées des calculs possédant une fonction d'activation et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système, elle ne contient aucun neurone tandis que les couches

suivantes sont des couches de neurones. La couche de sortie correspond aux sorties du système.[8]

Dans le modèle du Perceptron Multi–Couches, les perceptrons sont organisés en couches. Les perceptrons multi–couches sont capables de traiter des données qui ne sont pas linéairement séparables Avec l'arrivée des algorithmes de rétro propagation, ils deviennent le type de réseaux de neurones le plus utilisé.

L'utilité de plusieurs couches cachées n'a pas été démontrée. La figure I.4 illustre la structure d'un PMC présentant quatre neurones en entrée, trois neurones sur la couche cachée et deux en sortie. [9]

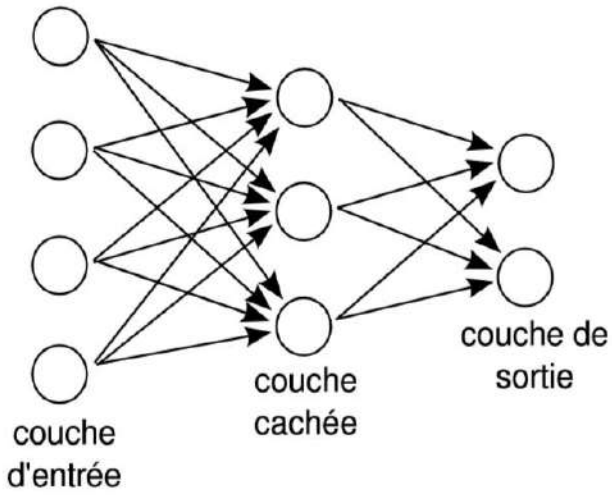


Figure I.4*. Modèle du Perceptron Multi–Couches.*

Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées. Les équations de propagation décrites plus haut s'appliquent à tous les neurones.

Cependant, le passage d'une couche à l'autre peut être formalisé sous forme matricielle. Soit un PMC dont le nombre de neurones sur la couche d'entrée est n0, n1 sur la couche cachée et n2 sur la couche de sortie. Les poids de la couche cachée peuvent s'écrire sous la forme d'une matrice $W \in Rn1 \times n0$.

Pour un vecteur $X \in Rn0$ présenté en entrée, le vecteur potentiel V et le vecteur de sortie Y pour la couche cachée s'écrivent donc :

$$V = WX \tag{1.4}$$

$$Y = \Phi(V) \tag{1.5}$$

Avec Φ une fonction d'activation. La sortie de la couche cachée devient ensuite l'entrée de la dernière couche [9]

8.2.1. Rétro propagation du gradient pour un Perceptron Multi–Couches :

La rétro propagation du gradient s'applique de manière récursive de la couche de sortie à la couche d'entrée. L'initialisation se fait donc par le calcul de l'erreur à la sortie de la dernière couche. Soit l'échantillon d'apprentissage $X \in Rn0$, et le vecteur de sortie désiré $D \in Rn2$. L'erreur sur la couche de sortie peut être définie comme précédemment :

$$L = 1/2 \left\| Y2 - D \right\|^2 \tag{1.6}$$

Où le vecteur Y2 représente la sortie du PMC (i.e. la sortie de la couche 2). De manière immédiate, le gradient de l'erreur en sortie peut être calculé :

$$\partial L/\partial Y2 = Y2 - D \tag{1.7}$$

La rétro propagation du gradient s'applique alors récursivement de la couche de sortie à la couche d'entrée. Pour une couche dont X est le vecteur d'entrée, V le potentiel, W la matrice de poids et Y la sortie, la rétro propagation s'écrit donc :

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial V} \tag{1.8}$$

$$= \phi'(V)\frac{\partial L}{\partial Y} \tag{1.9}$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial V}\frac{\partial V}{\partial X} \tag{1.10}$$

$$= W^T\frac{\partial L}{\partial V} \tag{1.11}$$

8.2.2. Mise à jour des poids :

Une fois la rétro propagation du gradient effectuée pour toutes les couches, les matrices de poids sont mises à jour :

$$W \leftarrow W - \lambda dW \tag{1.12}$$

Où λ est le taux d'apprentissage. [9]

9. Modèle Adaline

Au début des années 60 B. Widrow et M.E. Hoff ont proposé un système adaptatif, qu'ils ont appelé Adaline (de l'anglais Adaptive Linear Element).

La structure de l'Adaline diffère du perceptron par l'utilisation d'une seule cellule d'association et utilisation d'une fonction de seuil différent de celle de Heaviside (-1 et t+1).

De plus, il utilise un algorithme adaptif pour mesurer l'écarte entre la sortie réelle et la sortie du processeur élémentaire. Le schéma de l'Adaline est représenté en figure I.5. Le plus souvent, les entrées sont binaires et la réponse souhaitée est également binaire.

L'Adaline est le seul réseau de neurones artificiels utilisé massivement dans l'industrie, ce circuit est en effet utilisé dans la télécommunication pour améliorer le signal sur bruit en prenant en compte la variation de l'impédance des différentes lignes téléphoniques. [11]

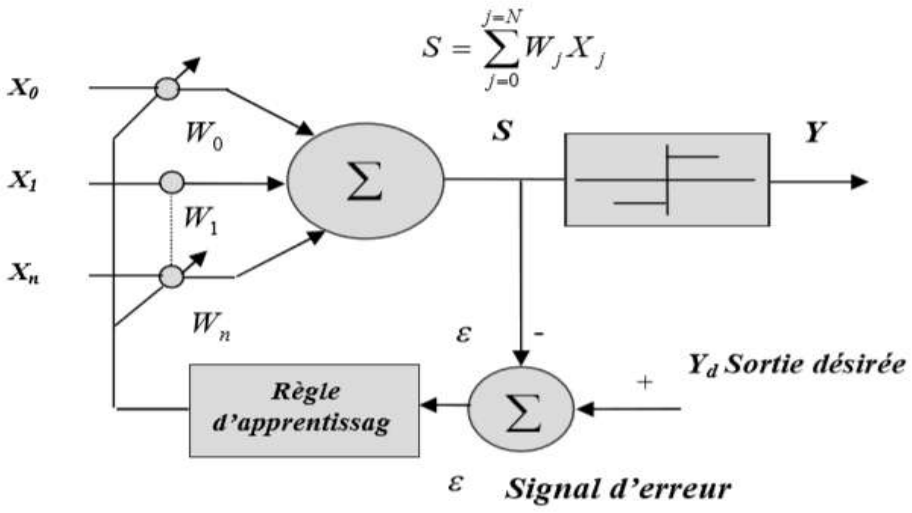


***Figure I.5**. Schéma de principe De l'Adaline*

9.1. Prédiction du signal avec Adaline

Nous allons ici illustrer comment un réseau de type Adaline peut être utilisé. Son point fort est qu'il possède la capacité de s'adapter à tout signal d'entrée. **[10]**

9.1.1. Création d'un réseau Adaline sous Matlab avec toolbox :

Nous allons définir un signal que nous échantillons à une fréquence de 40Hz. Le but de ce réseau est connaissant 5 valeurs successives de notre signal, de prédire la valeur suivante en utilisant le toolbox du Matlab2017b. **[10]**

```
Clc
Clear all ;
Close all ;
time = 0 :0.025 :5 ; Q=5 ;
T=sin (time*4*pi) +sin (time*3*pi) ;
Plot (time, T)
P=zeros (5, Q) ;
P (1,2 : Q)=T (1,1 : (Q-1)) ;
P (2,3 : Q)=T (1,1 : (Q-2)) ;
P (3,4 : Q)=T (1,1 : (Q-3)) ;
P (4,5 : Q)=T (1,1 : (Q-4)) ;
P (5,6 : Q)=T (1,1 : (Q-5)) ;
net=newlind (P, T) ;
a=Sim (net, P) ;
plot (time, a, time, T)
```

Figure I.6. *Script Matlab utilisé 1*

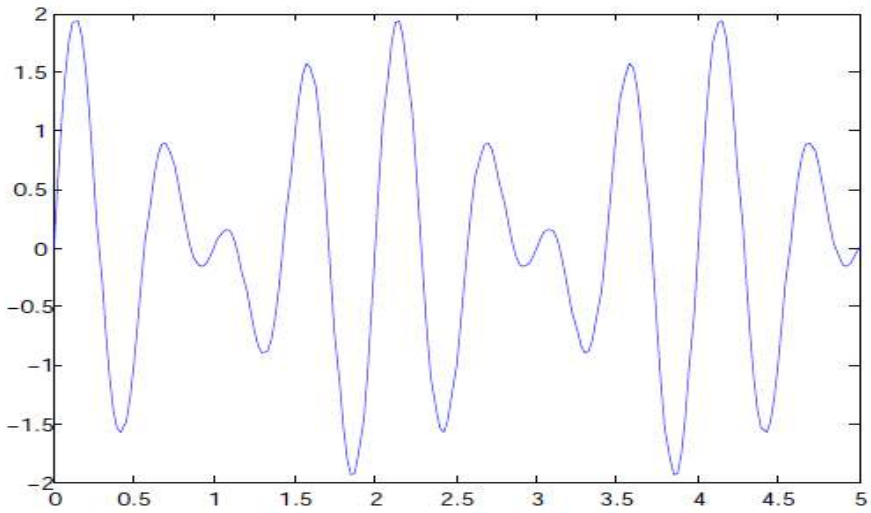


Figure I.7. *Signal fourni en apprentissage au réseau Adaline*

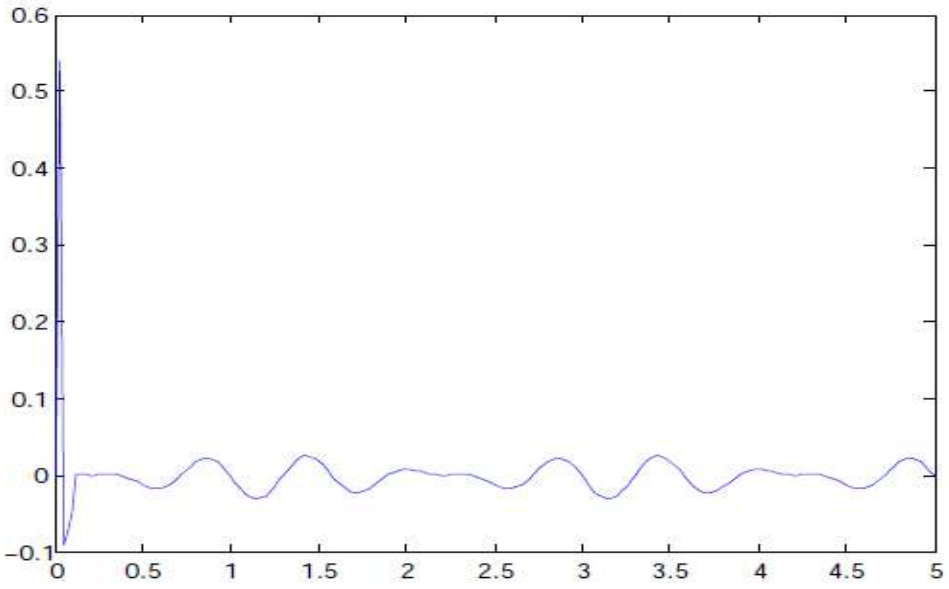


***Figure I.8**. Erreur entre le signal initial et le signal prédit par le réseau de neurones*

9.1.2. Création d'un réseau Adaline sous Matlab sans toolbox :

L'entrainement d'un réseau de neurones Adaline peut être effectué aussi par un programme Matlab sans utiliser le toolbox comme montré par l'exemple suivant.

```
Clc
Clear all ;
Close all ;
p= [0 :0.001 :5] ; T=sin (4*p*pi) ;
N=size (p, 2) ; P=4 ;V=0.004 ; W=rand (1, P)/8 ;
x=convmtx (p, P) ; x=x ( :,1 : N) ;
y=zeros (size(T)), eps=ones (size(T)) ;
For n=1 : N
y(n)=w*x ( : n) ;
Eps(n)=T(n)-y(n) ;
W=w+2*v*eps(n)*x ( : n) ;
end
Figure(1)
Plot (p, T, 'b', p, y,'-r'), grid, title ('target and predicted signals')
xlabel ('time [sec]')
Figure(2)
Plot (p, eps), grid, title ('prédiction error'), xlabel ('time [sec]') ;
```

***Figure I.9**. Script Matlab utilisé 2*

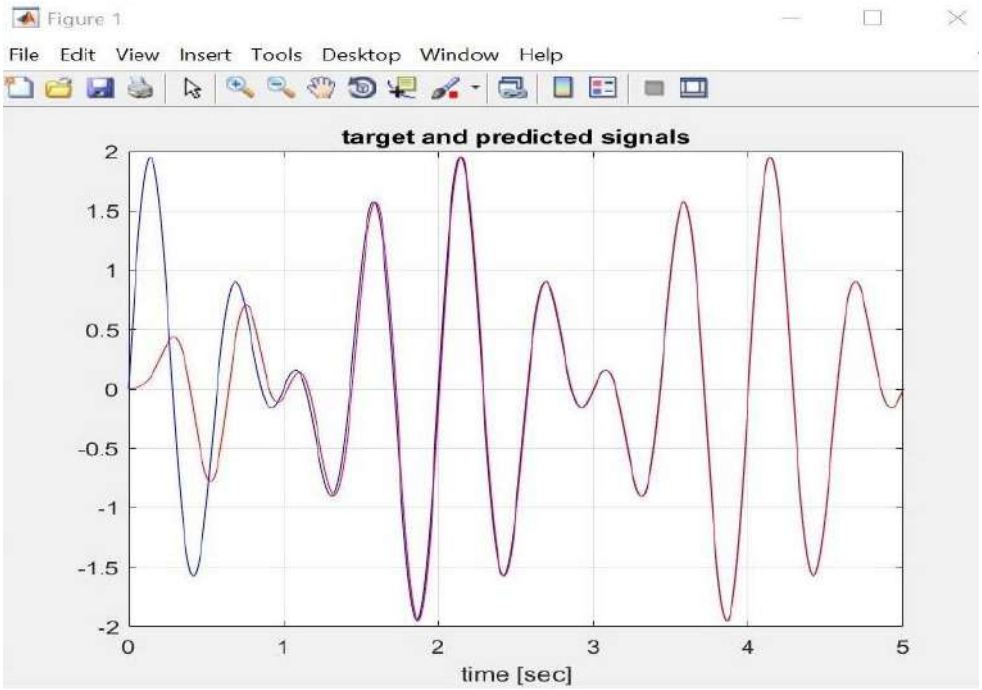


***Figure I.10**. Signal fourni en apprentissage au réseau Adaline*

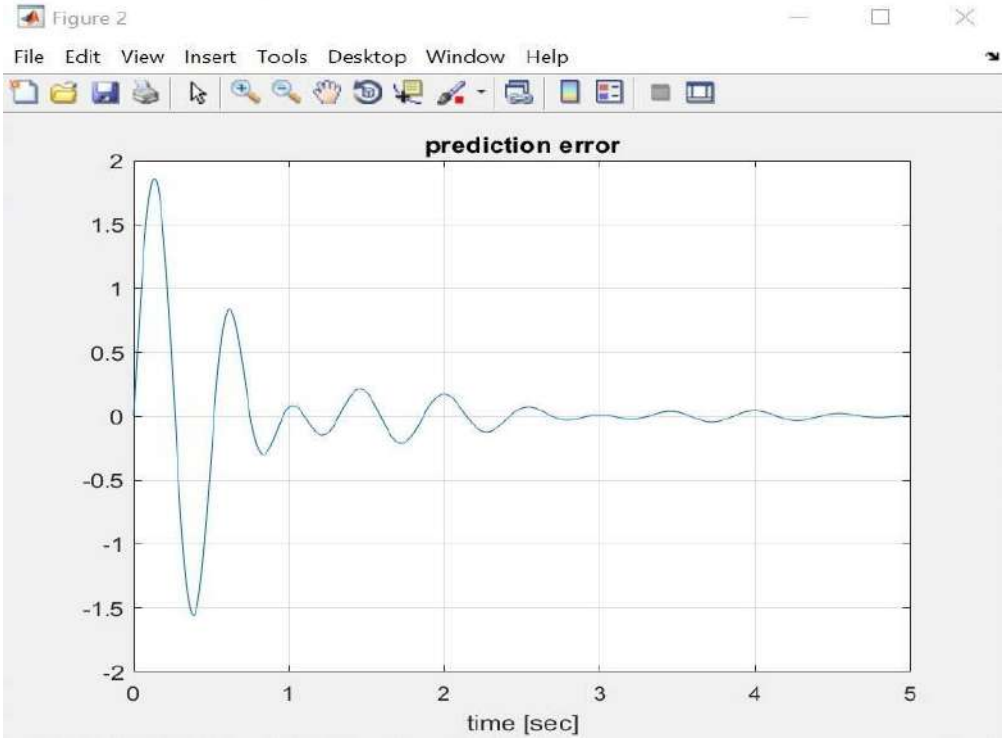


***Figure I.11**. Erreur entre le signal initial et le signal prédit par le réseau de neurones*

10. Conclusion :

A partir du fonctionnement du cerveau humain et du modèle neuronal biologique simple, les chercheurs ont pu développer et mettre aux points des modèles neuronaux artificiels complexes.

Les réseaux de neurones artificiels, sont le cœur de plusieurs applications réelles. C'est un outil indispensable qui a fait ses preuves dans des applications pratiques du fait de leurs caractéristiques de généralisation et de robustesses face au bruit.

Dans ce chapitre nous avons décrit les réseaux de neurones artificiels, les bases biologiques et leurs différentes architectures. Nous avons également exposé les différentes méthodes d'apprentissage des réseaux de neurones. Ces concepts et ces techniques ont été illustrés à travers de quelques exemples de réseaux comme l'Adaline et le perceptron multicouche. Les différents exemples réalisés en Matlab nous ont permis de mettre en exergues leurs capacités à apprendre et à prédire le comportement de signaux déterministes. Ces expérimentations nous ont permis par ailleurs de maitriser ces techniques pour pouvoir les appliquer ultérieurement pour résoudre des problèmes plus complexes tels que l'analyse et le traitement de signaux aléatoires.

Dans la prochaine partie, nous allons présenter l'architecture matérielle retenue pour exécuter les différents algorithmes neuronaux que nous avons mis au point.

CHAPITRE II : les circuits programmables FPGA

1. Introduction

L'idée de base du matériel programmable est d'avoir un circuit générique où la fonctionnalité peut être programmée pour une application particulière. Il représente la fonctionnalité en tant que circuit, où le particulier circuit peut être programmé pour répondre aux exigences d'une application. La distinction clé est que la fonctionnalité est implémentée comme un système parallèle plutôt que séquentiel. Toute fonction logique peut être mise en œuvre à l'aide d'une représentation à deux niveaux (OR ou AND). Par conséquent, il était logique que les anciens dispositifs logiques programmables consistent en une architecture matricielle programmable à deux niveaux, Field Programmable Gate Array ' FPGA ' est une percée technologie actuelle dans le domaine de la conception de système embarqué. Grâce à ses caractéristiques remarquables telles que vitesse, parallélisme, capacités, coût, l'efficacité et les capacités reconfigurables dans notre projet nous préférions de l'utiliser car L'utilisation du FPGA pour la mise en œuvre de réseaux neuronaux offre une flexibilité dans les systèmes programmables.

2. Les circuits programmables FPGA

2.1. Définition

Un FPGA est un appareil contenant une matrice de circuits logiques reconfigurables en réseau de portes. Lorsqu'un FPGA est configuré, le circuit interne est connecté de manière à créer une implémentation matérielle de l'application logicielle. Contrairement aux processeurs, les FPGA utilisent un matériel dédié pour la logique de traitement et ne disposent pas de système d'exploitation. Les FPGA sont véritablement parallèles par nature, de sorte que différents traitements ne doivent pas concurrencer pour les mêmes ressources. En conséquence, les performances d'une partie de l'application ne sont pas affectées par l'ajout de traitements. En outre, plusieurs boucles de contrôle peuvent être exécutées sur un même périphérique FPGA à des vitesses différentes. Les systèmes de contrôle basés sur FPGA peuvent appliquer une logique de verrouillage critique et peuvent être conçus pour empêcher un opérateur de forcer les E / S. Cependant, contrairement aux conceptions de carte de circuit imprimé (PCB) câblées qui ont des ressources matérielles fixes, les systèmes basés sur FPGA peuvent littéralement recâbler leurs circuits internes pour permettre la reconfiguration après le déploiement du

système de contrôle sur le terrain. Les dispositifs FPGA offrent les performances et la fiabilité d'un circuit matériel dédié.

Un seul FPGA peut remplacer des milliers de composants discrets en incorporant des millions de portes logiques dans une seule puce de circuit intégré (IC). Les ressources internes d'une puce FPGA sont constituées d'une matrice de blocs logiques configurables (CLB) entourés d'une périphérie de blocs d'E / S illustrée à la Figure II.1 Les signaux sont acheminés au sein de la matrice FPGA par des commutateurs d'interconnexion programmables et des routes filaires. [12]

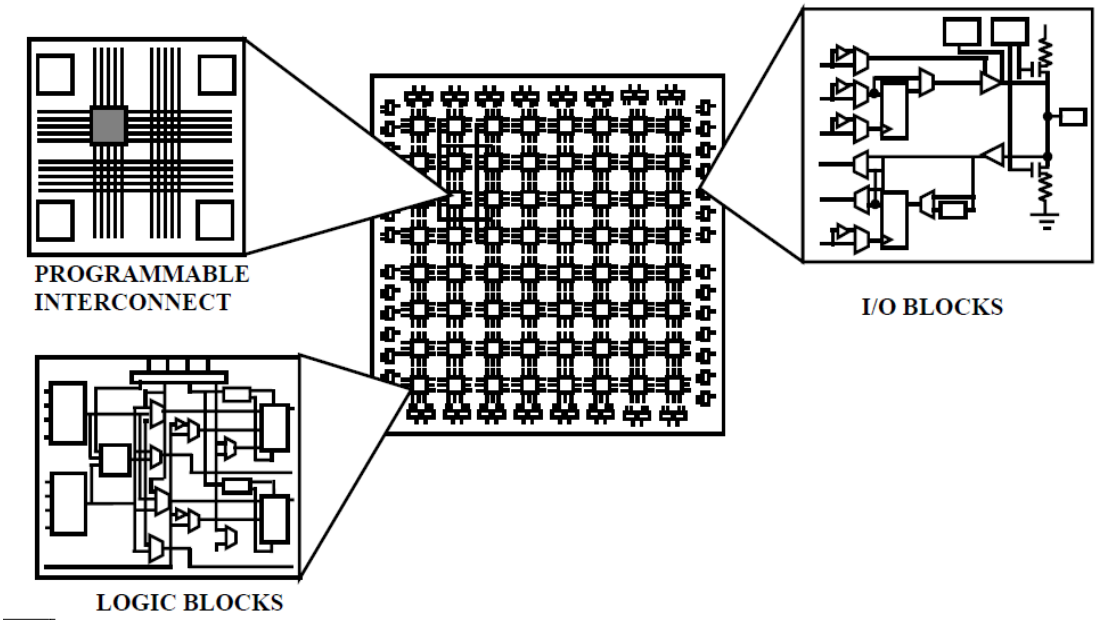


Figure II.1 *Structure interne du FPGA [12]*

Dans un FPGA, les blocs logiques sont implémentés à l'aide de portes d'entrée multiples, ce qui lui confère une conception plus compacte par rapport à une implémentation avec une logique AND-OR à deux niveaux. FPGA fournit à son utilisateur un moyen de configurer : L'intersection entre les blocs logiques AND et La fonction de chaque bloc logique. [12]

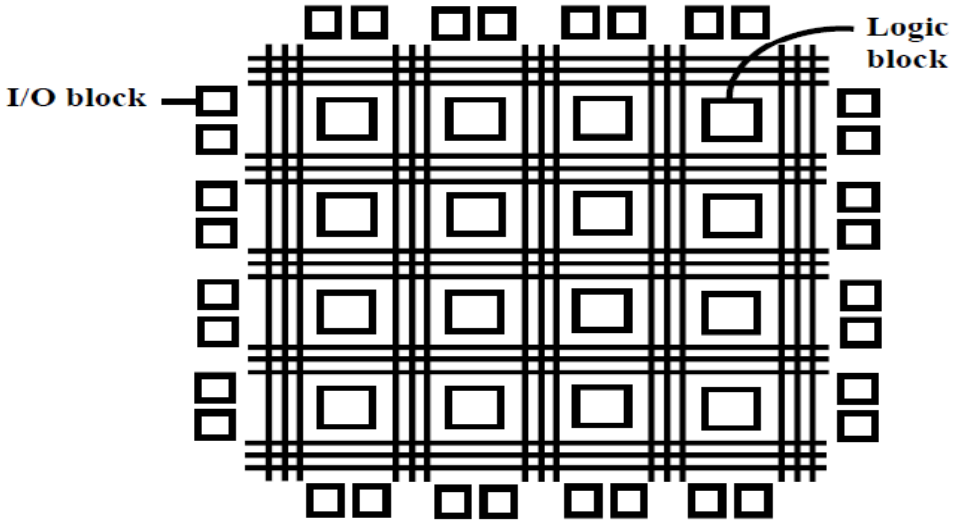


Figure II.2 *Structure interne simplifiée du FPGA [12]*

Le bloc logique d'un FPGA peut être configuré de manière à fournir une fonctionnalité aussi simple que celle d'un transistor ou aussi complexe que celle d'un microprocesseur. Il peut être utilisé pour mettre en œuvre différentes combinaisons de fonctions logiques combinatoire et séquentielle. [13]

3. Structure interne

La figure II.2illustre la structure de base et les composants essentiels d'un FPGA générique. La logique programmable consiste en un ensemble de blocs à grain fin utilisés pour implémenter la logique de l'application. Ceci est parfois appelé la structure du FPGA. Les blocs logiques sont généralement basés sur une architecture de table de consultation, leur permettant de mettre en œuvre toute fonction arbitraire des entrées. Les blocs logiques sont généralement disposés en mosaïque dans une structure en grille et interconnectés via une matrice de routage programmable qui permet aux blocs d'être connectés dans des configurations arbitraires. Les blocs d'entrée et de sortie (E / S) interface entre les composants internes ou le cœur du FPGA et les périphériques externes. Le routage signifie que pratiquement n'importe quel signal peut être acheminé vers n'importe quelle broche d'E/S du périphérique. [12]

Outre ces fonctionnalités de base, la plupart des FPGA fournissent une forme de synchronisation d'horloge permettant de contrôler la synchronisation d'un signal d'horloge par rapport à une source externe. Un réseau de distribution d'horloge fournit des signaux d'horloge à toutes les parties du FPGA tout en limitant le décalage d'horloge entre les différentes sections

d'une conception. Il existe également une logique dédiée pour charger la configuration dans le FPGA. Cette logique ne forme pas directement une partie dans la conception de l'utilisateur, mais c'est la charge supplémentaire nécessaire pour que les FPGA soient programmables et compatibles fournit les mécanismes permettant de charger la conception de l'utilisateur sur un FPGA.
Chacune de ces composantes est examinée un peu plus en détail dans les sections suivantes.

3.1. Logique des FPGA

La plus petite unité logique du FPGA est la cellule logique. C'est l'unité logique à grains la plus fine produisant un seul bit en sortie. Il se compose généralement d'une petite LUT (look up table) table de conversion et d'un verrou sur la sortie, comme indiqué dans Figure II.3. Le loquet est utilisé pour enregistrer la sortie. Sur certains FPGA, les deux sorties peuvent être fournies, comme dans Figure II.3; sur d'autres, un multiplexeur peut être utilisé pour choisir si le signal enregistré ou non enregistré est sorti. Les registres peuvent être utilisés pour construire des machines à états finis, des compteurs et pour enregistrer des données dans les systèmes en pipeline.
La LUT peut implémenter toute fonction arbitraire de ses entrées. Des fonctions plus complexes peuvent être mis en œuvre en cascadant plusieurs niveaux de LUT en connectant la sortie, X, à l'entrée de la couche suivante. Les fabricants de FPGA doivent faire un compromis en choisissant la taille des LUT. Envisager de mettre en œuvre une simple porte AND à deux entrées. Avec une grande LUT, une fraction importante de la logique serait inutilisée de plus petites TAL permettraient une utilisation plus importante des ressources. Cependant, lors de la mise en œuvre complexe fonctions avec de petites LUT, plusieurs cellules logiques devraient être mises en cascade, ce qui peut considérablement augmenter le délai de propagation. Pour des fonctions complexes, il serait préférable d'utiliser une table de correspondance plus grande. Pour équilibrer cela, la surface de silicium de la table croît de manière exponentielle avec le nombre d'entrées est plus efficace pour scinder une fonction complexe en plusieurs LUT. [12]

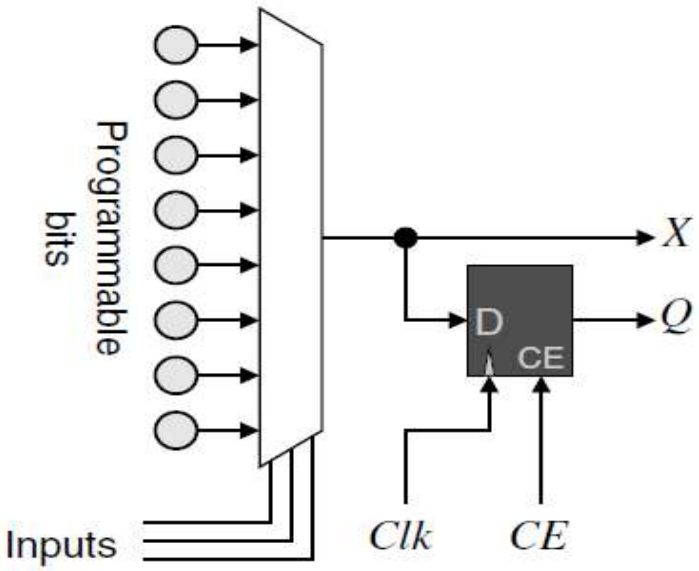


Figure II.3. *Une cellule logique est la pierre de fondation d'un FPGA [12]*

Les FPGA homogènes à grain fin consistent en une mer de blocs logiques identiques. Cependant, la plupart des FPGA actuels se sont éloignés de cette structure homogène, avec un mélange hétérogène de à la fois des blocs logiques à grain fin et des blocs plus gros à grain grossier destinés à l'accélération couramment utilisés opérations. Cela présente à la fois des avantages et des inconvénients en termes de conception. Les systèmes hétérogènes seront généralement plus rapides, car les blocs à grains grossiers n'auront pas le même niveau de frais généraux que ceux Cependant, les dispositifs hétérogènes sont également plus difficiles à programmer, et si le mélange des blocs spécialisés n'est pas correct, une grande partie des fonctionnalités peut rester sous-utilisée. [12]

3.2. Interconnexion

L'interconnexion programmable a pour but de permettre aux différents blocs logiques d'être utilisés de manière flexible connecté pour mettre en œuvre la fonctionnalité souhaitée. Il est impossible de fournir un dédié connexion directe de toutes les sorties possibles à toutes les entrées possibles. Dans une application donnée, seul une très petite fraction de ces connexions serait nécessaire ; cependant, aucune des connexions potentiellement peut être requis. La solution consiste à disposer d'un plus petit ensemble de ressources de connexion (lignes de

routage) pouvant être utilisées. Partagés de manière flexible pour créer les connexions requises pour une application donnée.

Puisque l'interconnexion prend place sur le silicium, il y a un compromis. Si seulement quelques lignes de routage sont fournies, plus d'espace est disponible pour la logique, mais des circuits complexes ne peuvent pas être construits en raison des ressources d'interconnexion insuffisantes. Alternativement, si trop de lignes de routage sont fournies, beaucoup ne seront pas utilisés, ce qui réduit la zone disponible pour la logique. Il y a un optimum quelque part entre ces extrêmes, où des lignes de routage suffisantes sont disponibles pour la plupart des applications, sans utiliser trop d'espace.

Le réseau de connexions pour la plupart des FPGA sont basés sur une structure de grille, semblable à celle montré par la Figure II.2. À chaque intersection de grille, une forme de commutateur de traverse permet une connexion programmable entre les lignes de routage horizontale et verticale.

Un autre facteur qui doit être considéré est le retard de propagation par le réseau d'interconnexion. Chaque commutateur que le signal passe à travers ajoutera au retard. Cela a conduit à une structure segmentée, où pas toutes les lignes de routage est commuté à chaque jonction. En outre, des connexions directes spécifiques peuvent être fournies entre les blocs logiques adjacents pour réduire le nombre de signaux qui doivent passer par la matrice interconnecte. Tout en ayant une gamme de différentes ressources d'interconnexion améliore la flexibilité et la vitesse, il peut rendre le processus de trouver de bonnes routes nettement plus complexes, en particulier pour les dessins qui utilisent la plupart de la logique disponible sur la FPGA.

Il est important qu'une seule sortie pilote une ligne de routage. Si deux sorties sont connectées ensemble avec un haut et l'autre bas, l'alimentation sera effectivement court-circuitée, endommageant la puce. L'un des rôles des outils de programmation FPGA est de veiller à ce que cela ne se produise jamais. Périphériques utilisant le bus les structures de routage doivent limiter le courant des transistors de commande où utiliser un drain ouvert et un pull-up passif. Les deux approches affectent la vitesse et la dissipation de puissance. [12]

3.3. Entrée et sortie

Les blocs d'entrée et de sortie permettent la connexion de signaux entre le FPGA et des périphériques externes. La flexibilité des FPGA exige qu'ils soient capables de se connecter à un large éventail d'autres composants. La plupart des broches d'entrée / sortie peuvent être programmées comme entrées, sorties ou les deux et être connectées directement à la mémoire interne logique du FPGA, ou passer par les registres. Cette gamme de fonctionnalités peut être fournie par un périphérique d'E / S générique. Tel que celui illustré à la figure II.4, chaque multiplexeur étant contrôlé par un programmable bit de configuration. La principale complexité des E / S provient du grand nombre de signaux standards différents utilisées par différents appareils et en assurant une synchronisation appropriée entre le FPGA et les composants hors puce.

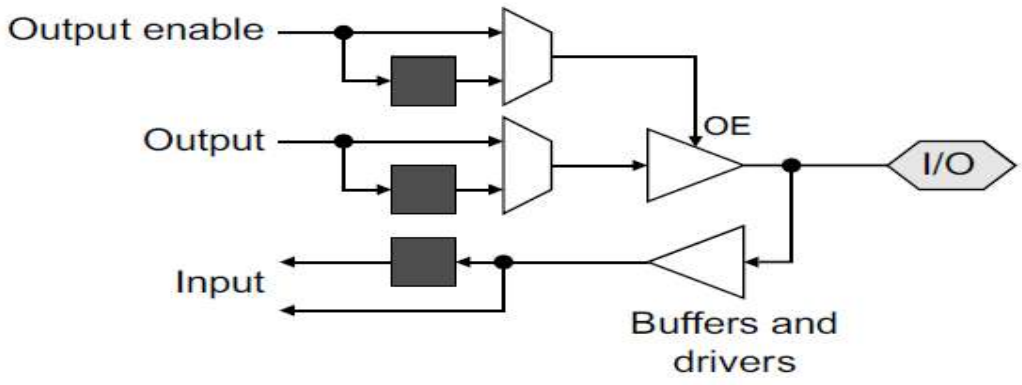


Figure II.4. *Bloc d'entrée / sortie générique [12]*

Lors de l'interfaçage de signaux haute vitesse, il est important de prendre en compte les effets sur la ligne de transmission. En règle générale, toute connexion plus longue qu'un huitième environ du temps de montée doit avoir la terminaison appropriée pour minimiser les réflexions du signal. Les FPGA étant devenus plus grands et plus rapides, il a devenu pratique courante de fournir une terminaison programmable sur puce. Si non fourni, il est nécessaire d'assurez-vous que la terminaison appropriée est construite hors puce pour des signaux de commutation rapides. De nombreux protocoles de communication de haut niveau nécessiteraient d'importantes ressources FPGA simplement pour gérer le protocole. Bien que cela puisse être mis en œuvre dans la structure du FPGA, la conception est souvent assez complexe et est mieux implémenté en utilisant une puce d'interface dédiée. Ces puces gèrent généralement l'accès aux médias (MAC) et la signalisation physique (PHY) ne nécessitant que des signaux logiques standard dans leur interface avec le FPGA. Certains FPGA intègrent des circuits MAC et PHY

dédiés aux communications couramment utilisées protocoles, tels que PCI Express et Gigabit Ethernet. La plupart des FPGA offrent une gamme de connexions d'E / S générales et dédiées. [12]

3.4. Horloge

Les FPGA sont conçus principalement comme des dispositifs synchrones, ils nécessitent donc un signal d'horloge. Bien que multiples horloges peuvent être utilisées dans une conception, les registres, les mémoires synchrones et les blocs d'E / S ne peuvent être contrôlés que par une horloge. Un domaine d'horloge comprend tous les circuits contrôlés par une horloge particulière. Étant donné qu'une horloge pilote potentiellement de nombreuses entrées sur l'ensemble du FPGA, il existe deux options spéciales. Caractéristiques des lignes d'horloge qui les distinguent des autres interconnexions d'un FPGA. Tout d'abord, c'est important que tous les registres d'un domaine d'horloge soient synchronisés simultanément. Le décalage d'horloge est la différence heure d'arrivée du signal d'horloge aux registres successifs d'un dessin (Friedman, 2001). Tout biais réduira la marge de synchronisation et impact sur la vitesse d'horloge maximale d'une conception. Bien que l'asymétrie puisse être minimisée en utilisant des réseaux de distribution d'horloge H-tree ou X-tree, ceux-ci ne correspondent pas bien à la structure en grille de FPGA (Lamoureux etWilton, 2006). Au lieu de cela, une structure de colonne vertébrale et de côtes est généralement utilisée (Figure II.5), où l'horloge est distribuée aux rangées à l'aide d'une épine puis aux éléments de la rangée à l'aide des nervures (Lamoureux etWilton, 2006). Une série de mémoires tampons est utilisée dans le réseau de distribution pour gérer les grande fan-out. En pratique, il est un peu plus complexe que cela, car le réseau doit prendre en charge plusieurs domaines d'horloges, et de nombreux FPGA ont également un mélange d'horloges locales et globales. [12]

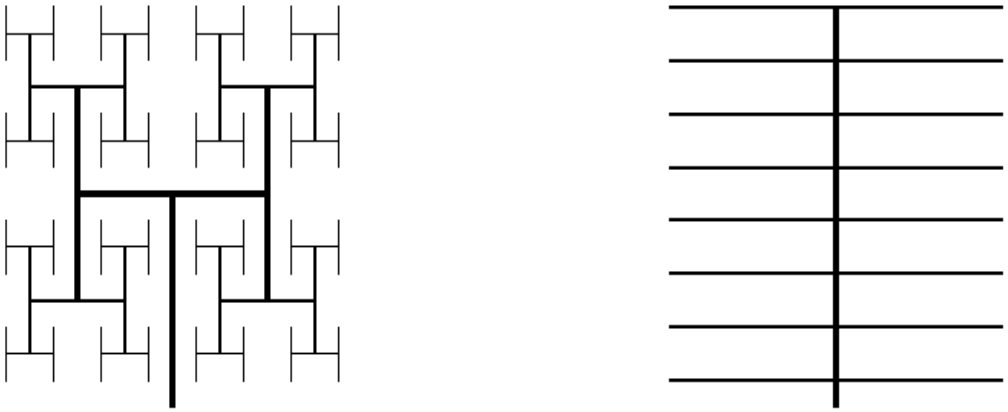


Figure II.5. *Réseaux de distribution d'horloge. À gauche : structure en H ; à droite : colonne vertébrale et structure des côtes. [12]*

3.5. Configuration

Tous les commutateurs, structures et options du FPGA sont contrôlés par des bits de. Plus généralement, les données de configuration sont contenues dans le FPGA dans des cellules de mémoire statique (SRAM), bien que les appareils à programmation flash sont disponibles chez certains fabricants. Les FPGA programmés en flash sont non volatiles, conserver leur configuration après la mise hors tension, afin qu'ils puissent immédiatement commencer à travailler après mise sous tension. Les FPGA à mémoire statique perdent leur configuration à la mise hors tension, ce qui nécessite la configuration fichier à recharger chaque fois que le système est allumé. La mémoire statique avait l'avantage d'être infiniment reprogrammable ; La mémoire flash a généralement un nombre limité de cycles d'écriture.

Un FPGA est configuré en chargeant le fichier de configuration dans la mémoire interne. Ceci est généralement accompli en transmettant le fichier de configuration sur le FPGA soit en série, soit en parallèle 8, 16 ou en certains cas 32 bits à la fois. Dans un système autonome, le fichier de configuration est généralement contenu dans un fichier. Mémoire flash externe, telle que celle illustrée à la figure II.6, avec la carte configurée pour configurer automatiquement le FPGA au démarrage. Les FPGA peuvent également être programmés en série via un JTAG (Joint Test Action Group) interface, et cette interface est couramment utilisée lors de la programmation de FPGA à partir d'un système informatique hôte. La nécessité de transférer la configuration dans le FPGA chaque fois que le système est alimenté expose le fichier de configuration rendant la propriété intellectuelle contenue dans vulnérable au piratage ou au

retournement ingénierie. Les FPGA haut de gamme surmontent cette limitation en chiffrant le fichier de configuration et déchiffrer la configuration une fois qu'elle est dans le FPGA. Certains FPGA prennent également en charge la relecture de la configuration à partir du FPGA. Cette fonctionnalité peut être utile pour déboguer une conception avec les outils appropriés, mais n'est généralement pas utilisée en fonctionnement normal. [12]

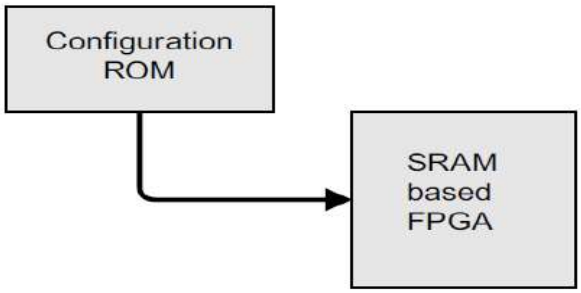


Figure II.6. *Configuration de FPGA basée sur la RAM statique [12]*

3.6. Consommation D'énergie

Dans les applications intégrées, la consommation électrique du système, et en particulier du FPGA, est essentielle. La puissance dissipée par un FPGA peut être divisée en deux composantes : la puissance statique et la puissance dynamique. La puissance dynamique est la puissance nécessaire pour commuter les signaux de 0 à 1 et vice versa. Inversement, l'énergie statique est celle qui est consommée sans commutation, simplement en raison du périphérique allumé. La plupart des FPGA sont basés sur la technologie CMOS. Un inverseur CMOS, le circuit CMOS le plus élémentaire, est présenté à la figure II.7. Il se compose d'un transistor de tirage à canal P et d'un transistor à tirage bas à canal N transistor. Un seul des transistors est allumé à la fois et, comme les transistors MOS ont une très haute résistante aux chocs, les circuits CMOS ont généralement une très faible dissipation de puissance statique. [12]

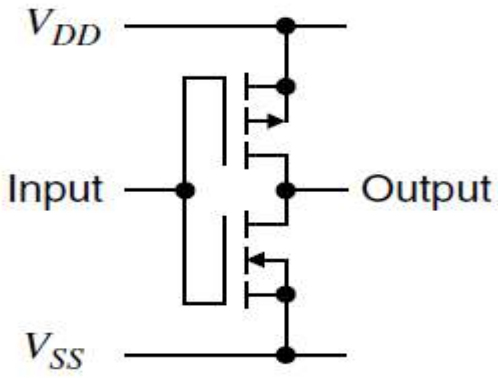


Figure II.7 *Inverseur CMOS.*

4. Les Familles et caractéristiques des FPGA

Dans la section précédente, les caractéristiques architecturales des FPGA ont été examinés d'une manière générale. Dans cette section, les caractéristiques particulières d'appareils sont examinées plus en détail. Les deux principaux fabricants de FPGA en termes de part de marché sont Xilinx et Altera, bien que plusieurs d'autres qui fournissent des FPGA. Les offres actuelles de chacun de ces éléments sont décrites à leur tour. [12]

4.1. Xilinx

Xilinx est le premier fournisseur mondial de technologies et d'appareils programmables. La société va au-delà de la logique programmable traditionnelle pour permettre la programmation matérielle et logicielle, intégrer des fonctions de signaux mixtes analogiques et numériques et permettre de nouveaux niveaux d'interconnexion programmable dans les circuits intégrés 3D monolithiques et multiples. Les produits Xilinx sont associés à un environnement de conception de nouvelle génération et à l'IP pour répondre à un large éventail de besoins des clients, de la logique programmable à l'intégration de systèmes programmables.

5. MicroBlaze

5.1. Définition

Le processeur MicroBlaze est un processeur Soft Core RISC à 3 étages avec une architecture Harvard et des registres internes de 32 bits. Il dispose d'un bus d'instructions et de données internes et externes séparés (ILMB, DLMB, IOPB et DOPB). Il est entièrement intégré dans l'environnement de conception des FPGA's de Xilinx. Il peut être facilement configure pour de nombreuses applications différentes. Il possède plus de 70 options de configuration, permettant à l'utilisateur de sélectionner ou de modifier les composants internes selon les besoins de l'application.

La figure II.8 représente le diagramme en blocs du MicroBlaze. Les parties en couleur grise sont optionnelles et montrent à quel point le MicroBlaze est configurable. Les caches permettent au processeur un gain en temps d'exécution en récupérant des données directement à l'intérieur sans aller les chercher dans la mémoire. Le « Barrel Shift », le « Multiplieur », le « Diviseur » et le « FPU : Floating Point Unit » permettent d'accélérer les traitements de données.

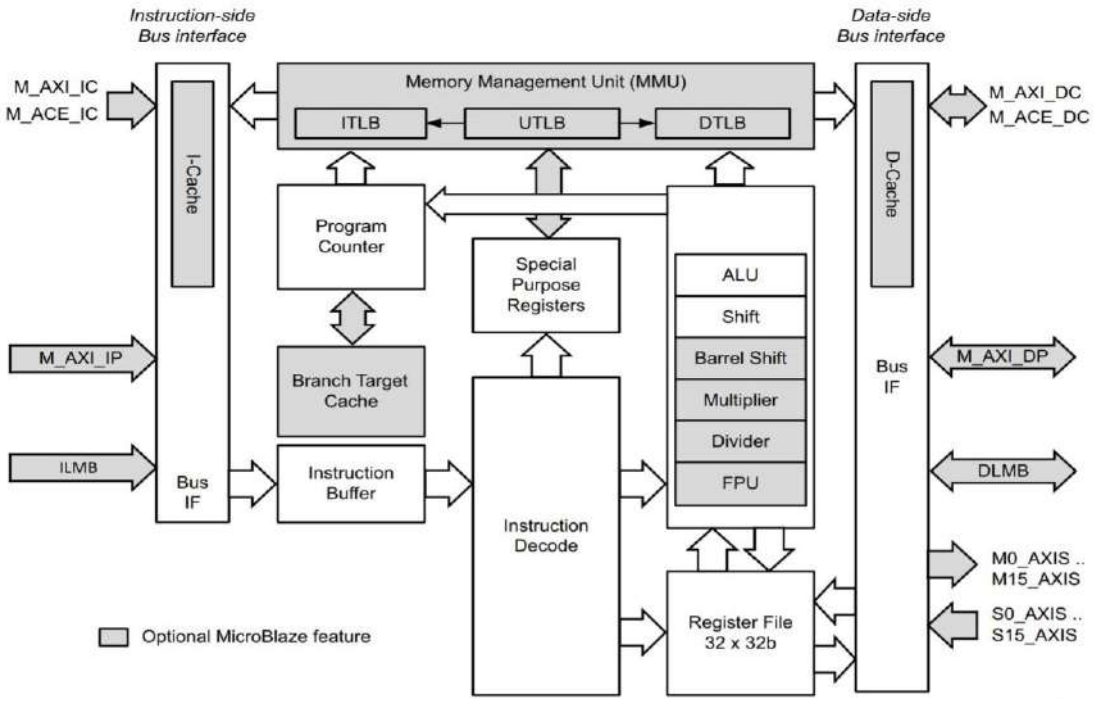


Figure II.8. *Schéma fonctionnel de base de MicroBlaze [14]*

5.2. Caractéristiques

Le processeur Soft Core MicroBlaze est hautement configurable, nous permettant de sélectionner un ensemble spécifique de caractéristiques requises par notre conception.
Les fonctionnalités fixes du processeur comprennent :

- Trente-deux registres à usage général 32 bits
- Mot d'instruction 32 bits avec trois opérandes et deux modes d'adressage
- Bus d'adresses 32 bits
- Pipeline à numéro unique

En outre ces fonctionnalités fixes, le processeur MicroBlaze est paramétré pour permettre la sélection et l'activation de fonctionnalités supplémentaires.
Xilinx recommande que toutes les nouvelles conceptions utilisent la dernière version préférée de MicroBlaze processeur.[14]

5.3. Bus et protocole AXI

Le MicroBlaze peut être connecte à n'importe quel composant dans la zone logique programmable à travers l'ensemble des bus AXI (Advanced eXtensible Interface).
Le protocole AXI, est une norme pour la connexion, la gestion et la communication des blocs dans les modèles SoC. L'objectif principal est de faciliter la gestion et le développement des systèmes multi-processeurs, de ses contrôleurs et ses périphériques. Ce protocole est très robuste et garantit une communication sure et rapide entre les différents modules liés à un système d'interface maitre-esclave. Du point de vue pratique, le protocole AXI est utilisé par la plupart des IPs de Xilinx.
Chaque module basé sur l'AXI possède son propre pilote, qui peut être initialise dans le code source en exécutant des fonctions spécifiques. Par conséquent, ce protocole est fondamental pour la configuration des paramètres matériels, la gestion interne des données et le contrôle complet du matériel. Il existe trois types d'interface AXI :

- AXI-Full : Pour les exigences des mémoires mappées de haute performance.
- AXI-Lite : pour les simples accès mémoire, à faible débit.
- AXI-Stream : pour des flux de données rapides en continu.[15]

6. Conclusion

L’implémentation d’un réseau de neurones sur FPGA est possible mais des exigences doivent être réfléchies en avance, telle que la précision à utiliser (virgule flottante / point fixe / entier), nombre maximum de neurones dans chaque couche. Si ce nombre n'est pas choisi avec soin, un débordement de mémoire tampon peut survenir ce qui sera néfaste au bon fonctionnement du réseau de neurones. Il est un peu difficile d'implémenter des fonctions non linéaires dans le matériel sans un support de bibliothèques appropriées. Des solutions plus simples consistent à utiliser des tables de correspondance ou à implémenter une fonction linéaire par morceaux. Cette approche nous permettra d’implémenter le réseau de neurones de manière plus efficace.

Les différentes étapes nécessaires à la conception et à l’implémentation de notre algorithme neuronal sont présentées et détaillées dans le chapitre suivant.

Chapitre III : Conception et implémentation FPGA

1. Introduction

L'objectif de ce chapitre est d'implémenter un réseau de neurones artificiel de type ADALINE sur une carte FPGA. Cette implémentation a pour but de détection des harmoniques.

Dans cette étude, nous commençons par la simulation de RNA sur Matlab, pour assurer la possibilité d'implémentation de ce réseau avec les fonctions de base (sans toolbox).

Ensuite nous concevons ce réseau sur System Generator. Ce dernier permet de générer un IP-Core. Cet IP-Core sera utilisé avec un processeur MicroBlaze dans une architecture embarquée qui sera implémenté sur une carte FPGA de type Nexys A7-100T. Afin d'avoir une implémentation optimale, nous devons assurer une efficacité, une rapidité d'exécution, et une faible consommation des ressources sur le circuit FPGA.

2. Les phases d'étude et d'implémentation

La figure III.1 montre les étapes suivis au niveau logiciel et matériel pour l'implémentation.

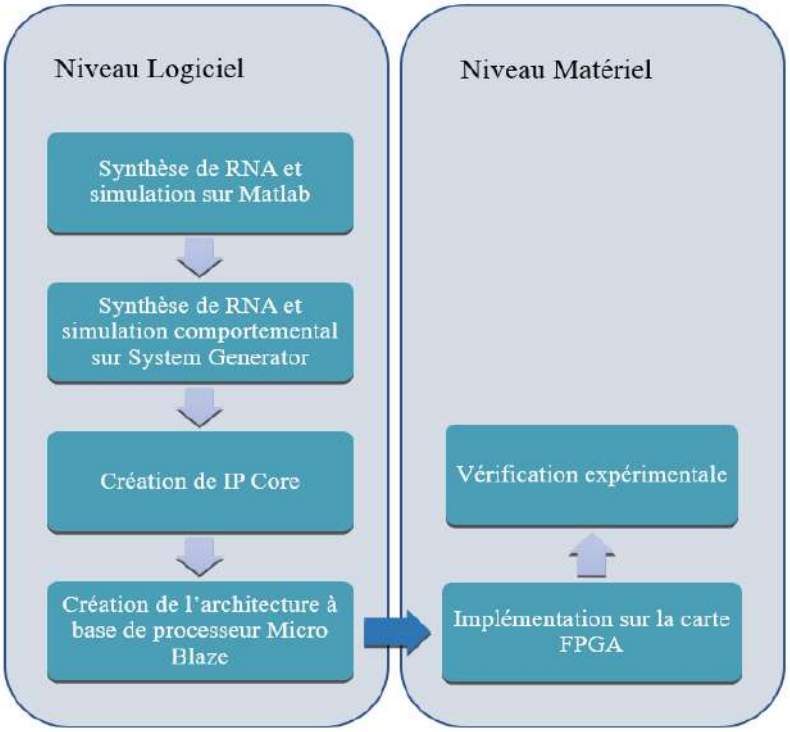


Figure III.1. *Phases d'implémentation sur FPGA*

3. Synthèse de RNA et simulation sur Matlab

Pour assurer une efficacité, et une possibilité d'exécution sur la carte FPGA. On se propose d'étudier un réseau de neurones artificiel de type ADALINE (Adaptative Linear Element) sur Matlab, avec seulement les fonctions de base utilisées par Matlab, et sans utiliser le Toolbox concerné par les réseaux de neurones. Ce réseau est développé afin d'estimer les fréquences et les amplitudes des harmoniques dans un signal électrique. On a choisi le type ADALINE à cause de sa rapidité et sa précision. La fonction d'activation dans ce cas est lineaire f(x)=x. La figure III.2, montre la structure de notre réseau.

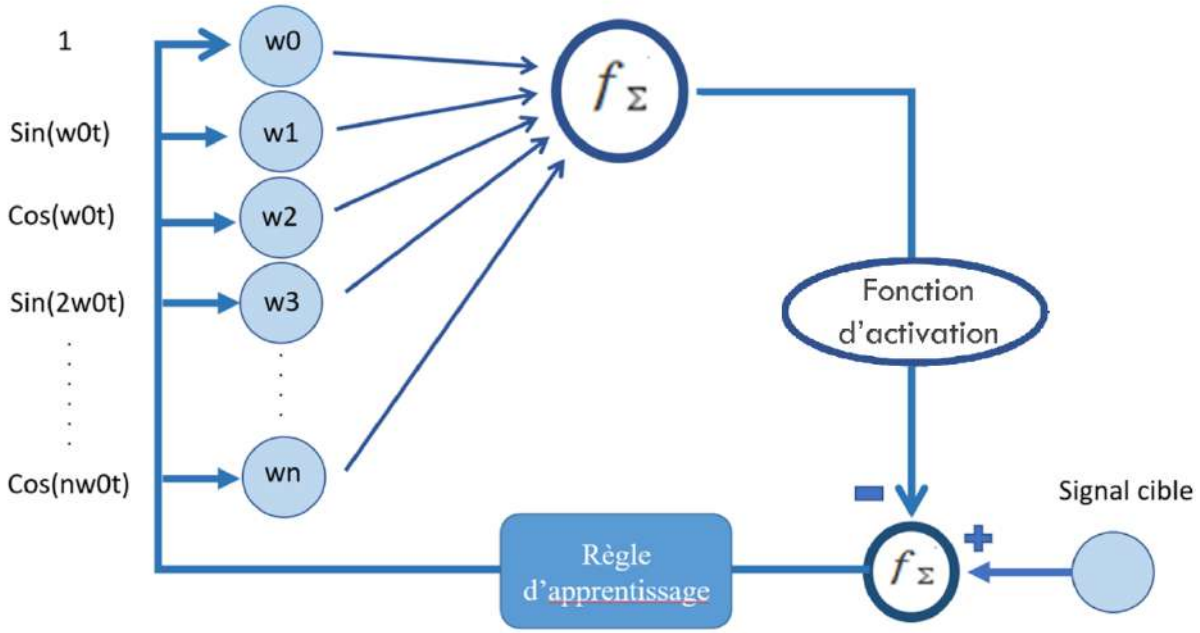


Figure III.2. *Structure du réseau*

3.1. Organigramme d'apprentissage

La figure suivante représente l'organigramme qui montre le principe d'adaptation des signaux d'entrée au signal cible (apprentissage).

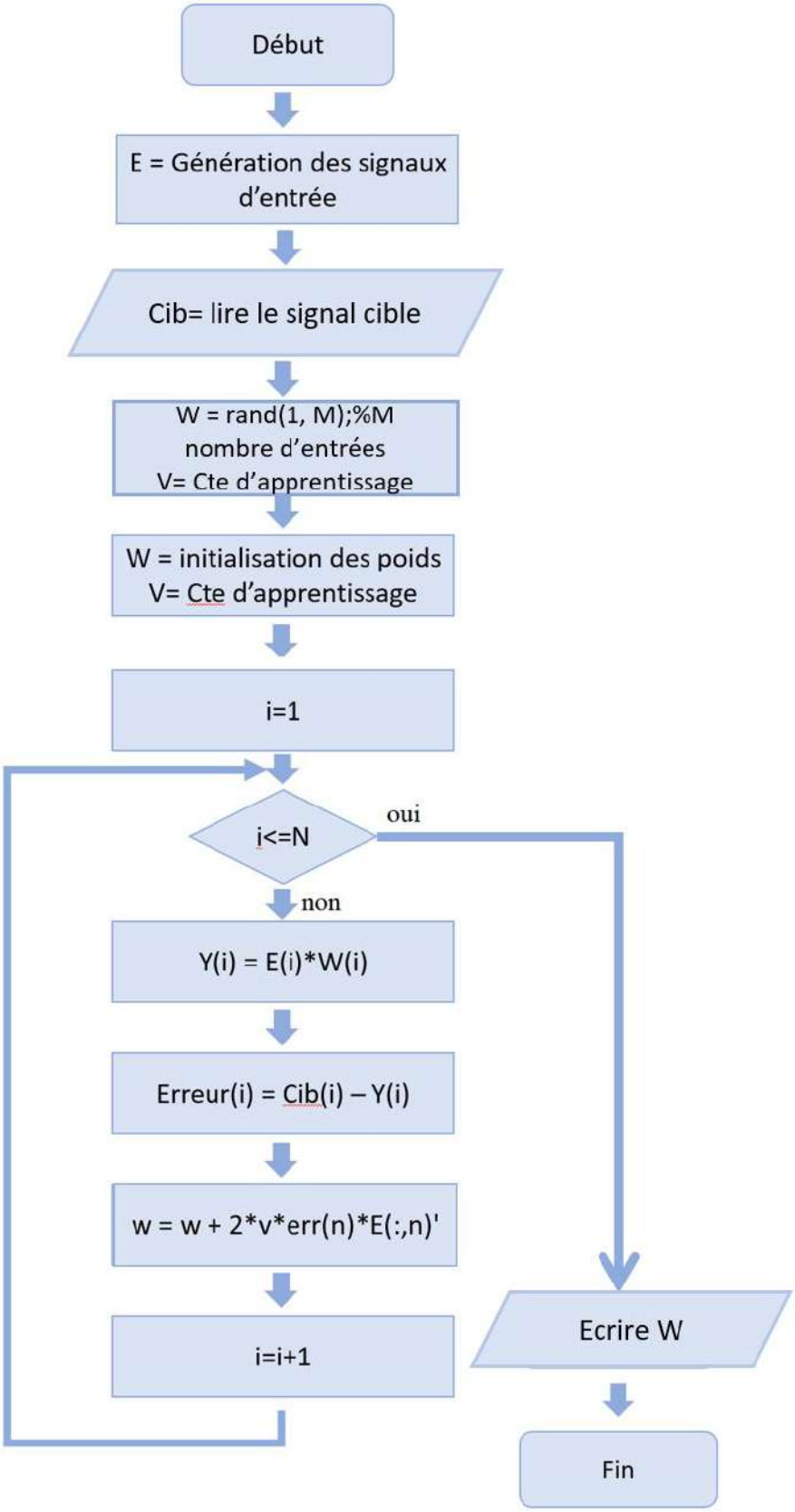


Figure III.3. *L'organigramme d'apprentissage*

3.2. Résultats d'essai de réseau

On a effectué un apprentissage sur un signal composé de plusieurs signaux sinus et cosinus. La figure suivante montre la courbe d'adaptation des signaux d'entrée au signal cible.

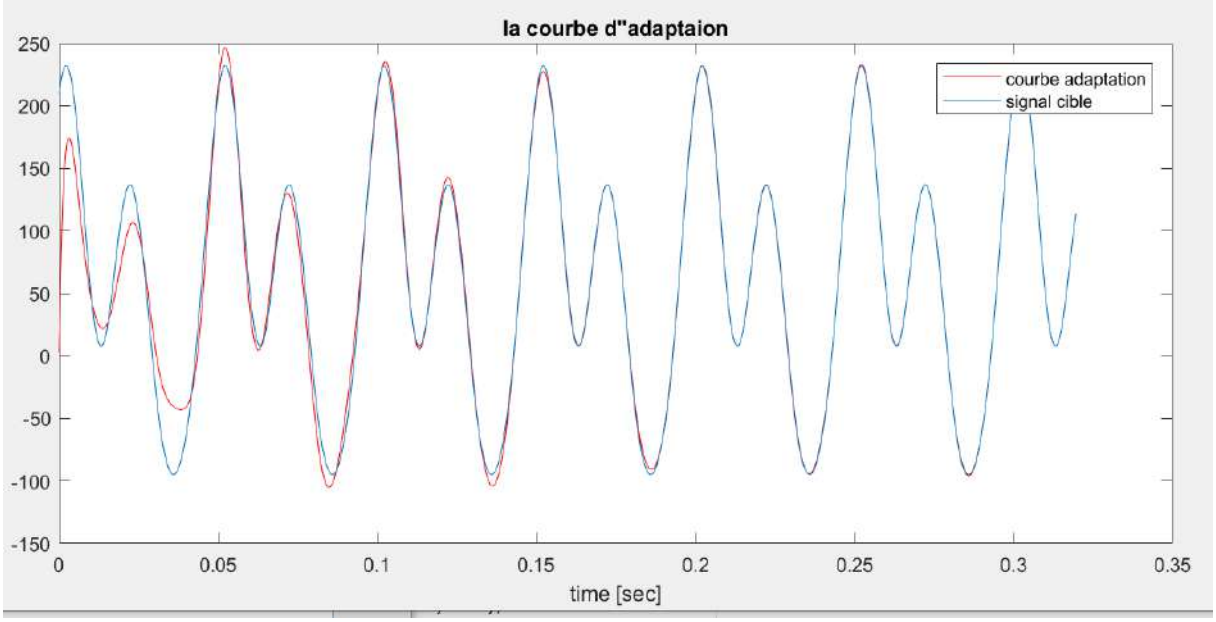


Figure III.4. *La courbe d'apprentissage*

Le signal d'adaptation rapproche au signal cible jusqu'ils seront quasiment adaptées, donc l'erreur va tend vers zéro comme illustre la figure III.5.

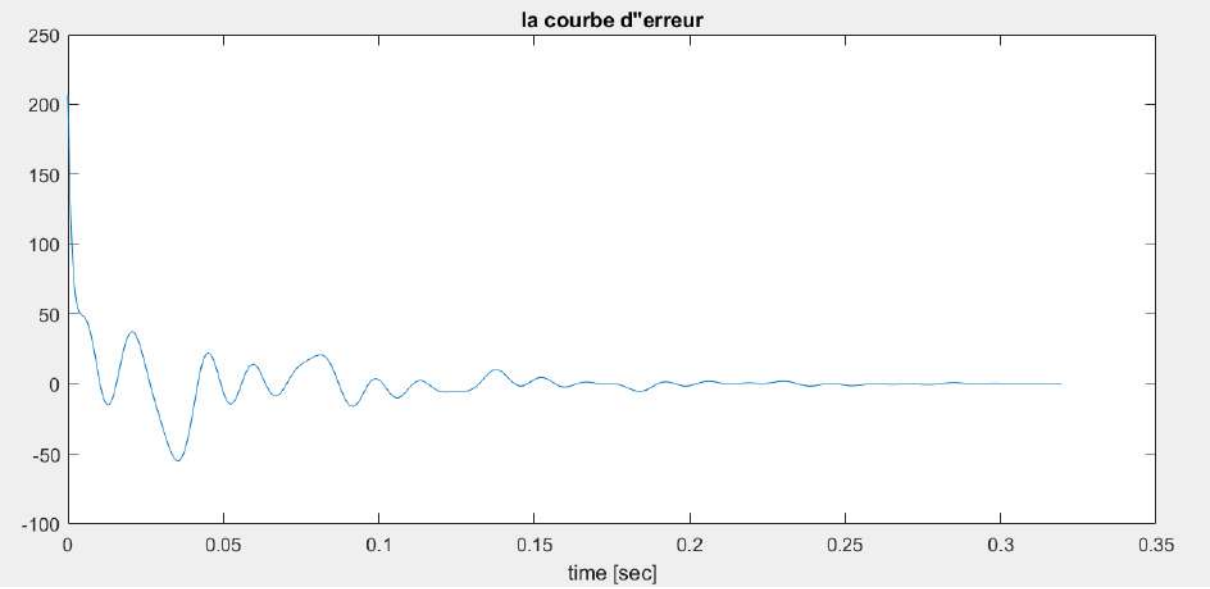


Figure III.5. *La courbe d'erreur*

4. System Generator

Xilinx System Generator permet la conception de systèmes matériels à partir d'un environnement graphique de haut niveau Simulink. System Generator étend la conception traditionnelle du langage de description de matériel (HDL) en fournissant des modules graphiques. Il ne nécessite donc pas une connaissance détaillée de ce langage complexe. Le langage graphique Simulink permet une abstraction de la conception grâce à l'utilisation des blocs et sous-systèmes disponibles dans le System Generator. Cela réduit le temps nécessaire entre les dérivations de conception, de contrôle et la mise en œuvre matérielle. En outre, le logiciel fournit la simulation matérielle et la vérification matérielle dans la boucle, appelée Co-Simulation matérielle, à partir de cet environnement. Cette méthodologie facilite la vérification et la mise en œuvre du matériel par rapport à une approche basée sur le HDL.

La simulation Simulink et l'approche matérielle en boucle constituent une solution beaucoup plus économique que d'autres méthodes. La possibilité de réaliser rapidement et directement une conception de système de contrôle en tant que système intégré en temps réel facilite grandement le processus de conception. [16]

4.1. Méthodologie de conception pour la mise en œuvre sur FPGA avec Xilinx System Generator

Un système de prototypage rapide efficace nécessite un environnement de développement ciblant la plate-forme de conception matérielle. Les outils utilisés sont MATLAB R2017b avec Simulink de MathWorks, System Generator 2018.2 pour DSP et Vivado de Xilinx présentent de telles capacités, la figure III.6 présente conception de Flux de XSG. System Generator automatise le processus de conception, corrige, implémente et vérifie les FPGA basés sur Xilinx. Il fournit une interface de Co-Simulation HDL haute vitesse, une estimation des ressources au niveau du système et une simulation accélérée via du matériel dans les interfaces de Co-Simulation en boucle, qui peut augmenter les performances de la simulation jusqu'à 1000x. Il fournit également une plate-forme d'intégration système pour la conception des FPGA DSP, qui permet aux composants RTL, Simulink, MATLAB et C / C ++ d'un système DSP de s'unir dans un environnement de simulation et de mise en œuvre unique.
System Generator prend en charge un bloc de boîte noire qui permet l'importation de RTL dans Simulink et la Co-Simulation avec ModelSim ou Xilinx ISE Simulator Figure III.6**.** [17]

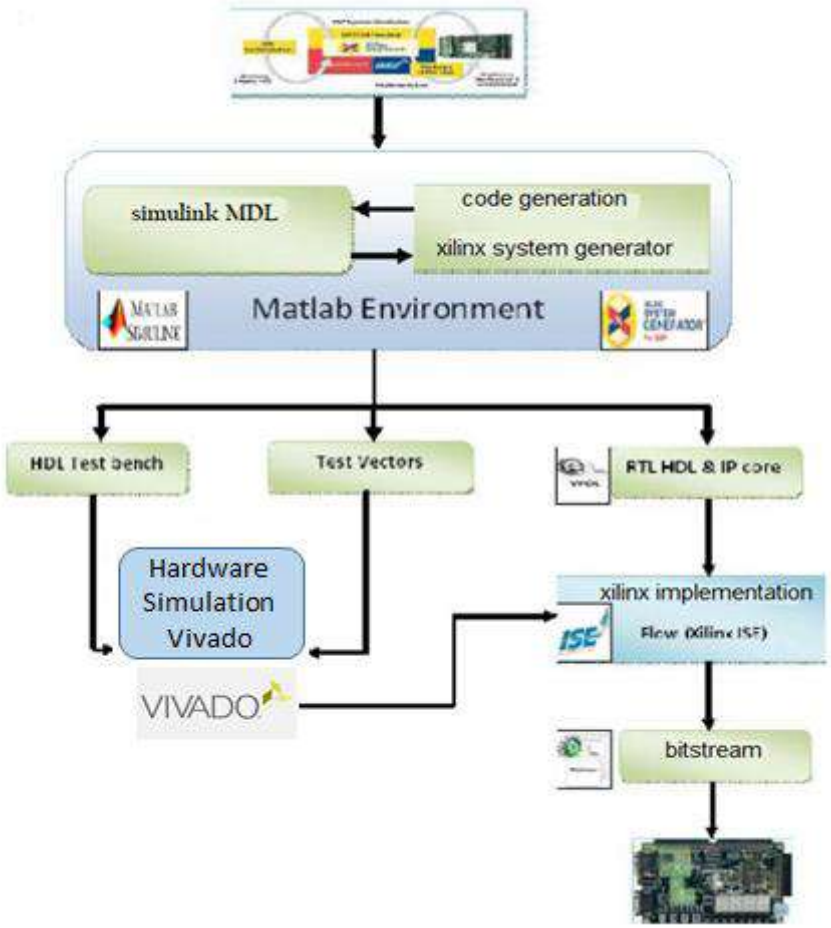


Figure III.6. *Conceptions de Flux du System Generator [18]*

System Generator simule les boîtes noires en lançant automatiquement un simulateur HDL, en générant du HDL supplémentaire selon les besoins (analogue à un banc d'essai HDL), en compilant le HDL, en programmant des événements de simulation et en gérant l'échange de données entre Simulink et le simulateur HDL. C'est ce qu'on appelle la Co-Simulation HDL.

System Generator fournit une Co-Simulation matérielle, permettant d'incorporer une conception s'exécutant directement dans un FPGA dans une simulation Simulink.

Les cibles de compilation "Hardware Co-Simulation" créent automatiquement un train de bits et l'associent à un bloc. (Figure III.7).

Lorsque la conception du système est simulée dans Simulink, les résultats de la partie compilée sont calculés dans du matériel FPGA réel, ce qui entraîne souvent des temps de simulation nettement plus rapides, tout en vérifiant l'exactitude fonctionnelle du matériel. System Generator for DSP prend en charge Ethernet ainsi que la communication JTAG entre une plate-forme matérielle et Simulink.19]]

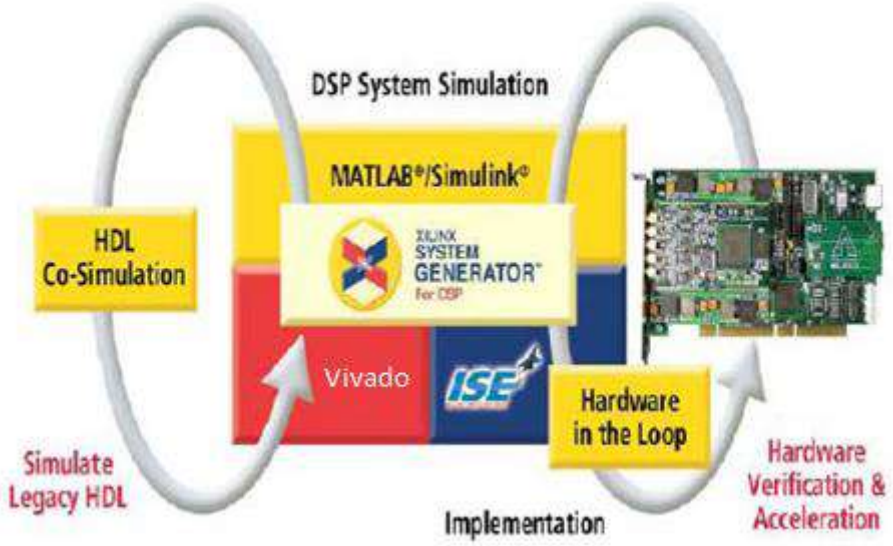


Figure III.7. *Environnement de Co-simulation matériel-logiciel basé sur FPGA [20]*

4.2. Blocksets de System Generator

Les Blocksets de Systeme generator est une bibliothèque de blocs pouvant être connectés dans l'éditeur de blocs Simulink pour créer des modèles fonctionnels d'un système dynamique. Pour la modélisation du système, ces blocs sont utilisés comme les autres blocs de Simulink. Les blocs fournissent des abstractions de fonctions mathématiques, logiques, de mémoire et DSP pouvant être utilisées pour créer des systèmes sophistiqués de traitement du signal (et autres). Il y a aussi des blocs qui fournissent interfaces avec d'autres outils logiciels (par exemple, FDATool, ModelSim) ainsi que le logiciel de génération de code System Generator (Figure III.8) [10]

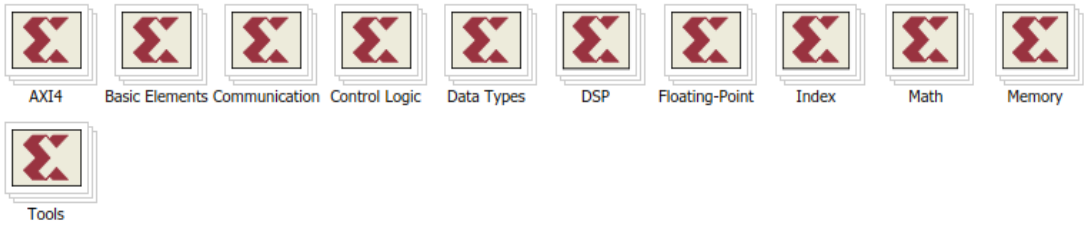


Figure III.8. *La liste des blocksets de System Generator*

Les blocs de System Generator sont bit-précis et cycle-précis. Les blocs bit-précis produisent des valeurs dans le Simulink qui correspondent aux valeurs produites dans le matériel ; cycle-précis produisent des valeurs à des temps correspondants. [21]

La figure III.9. represente quelques blocs des elements de base.

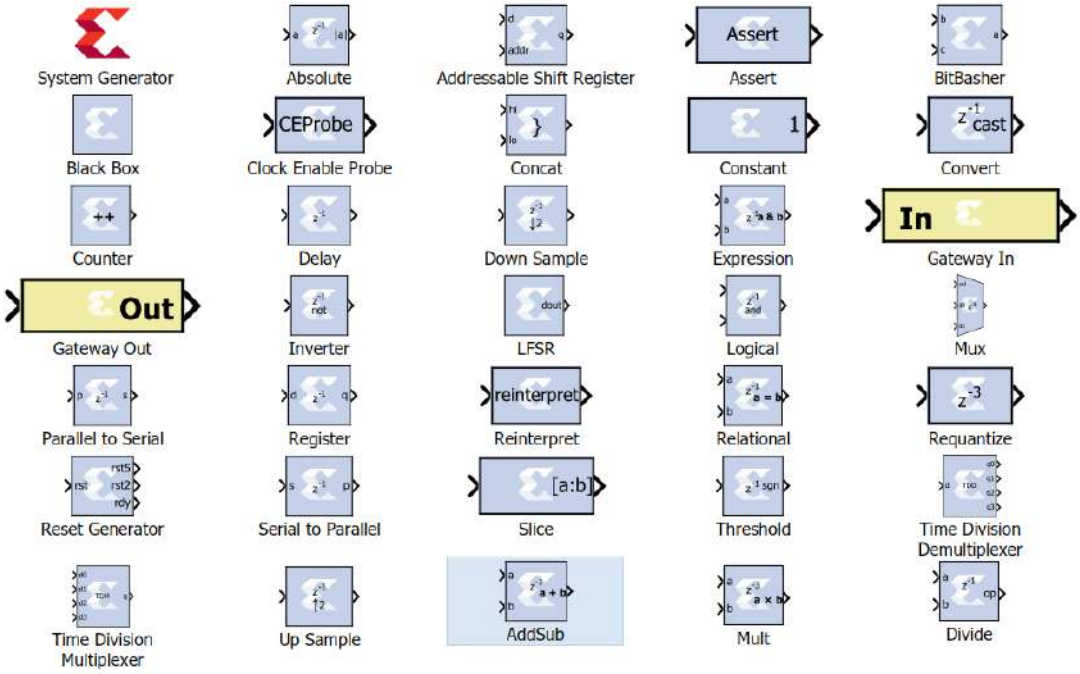


Figure III.9. *La liste des blocs : éléments de base*

Quelques définitions des blocs utilisés :

- System Generator « jeton » : Le bloc System Generator permet de contrôler le système et les paramètres de simulation et est utilisé pour appeler le générateur de code. Chaque modèle Simulink contenant n'importe quel élément du Xilinx Blockset doit contenir au moins un bloc System Generator (jeton). Une fois Le bloc System Generator est ajouté à un modèle, il est possible de spécifier comment la génération de code et la simulation doivent être traitées.
- Gateway In : Convertit les entrées de type simulink entier, simple, double et à virgule fixe en type de données Xilinx à virgule fixe ou à virgule flottante.
- Gateway Out : Les blocs Xilinx Gateway Out sont les sorties de la partie Xilinx de votre conception Simulink. Ce bloc convertit le type de données à virgule fixe System Generator en Simulink Double.

- Mult : Le bloc Xilinx Mult implémente un multiplicateur. Il calcule le produit des données sur ses deux ports d'entrée, produisant le résultat sur son port de sortie.
- AddSub : Le bloc Xilinx AddSub implémente un additionneur / soustracteur. L'opération peut être fixe (addition ou soustraction) ou modifiée dynamiquement sous le contrôle du signal du sous-mode.
- Delay : Le bloc Xilinx Delay implémente un retard fixe de N cycles

5. Synthèse de RNA et simulation sur System Generator

5.1. La modélisation de RNA sous Simulink

La conception de RNA sous Simulink est représentée sur la figure III.10

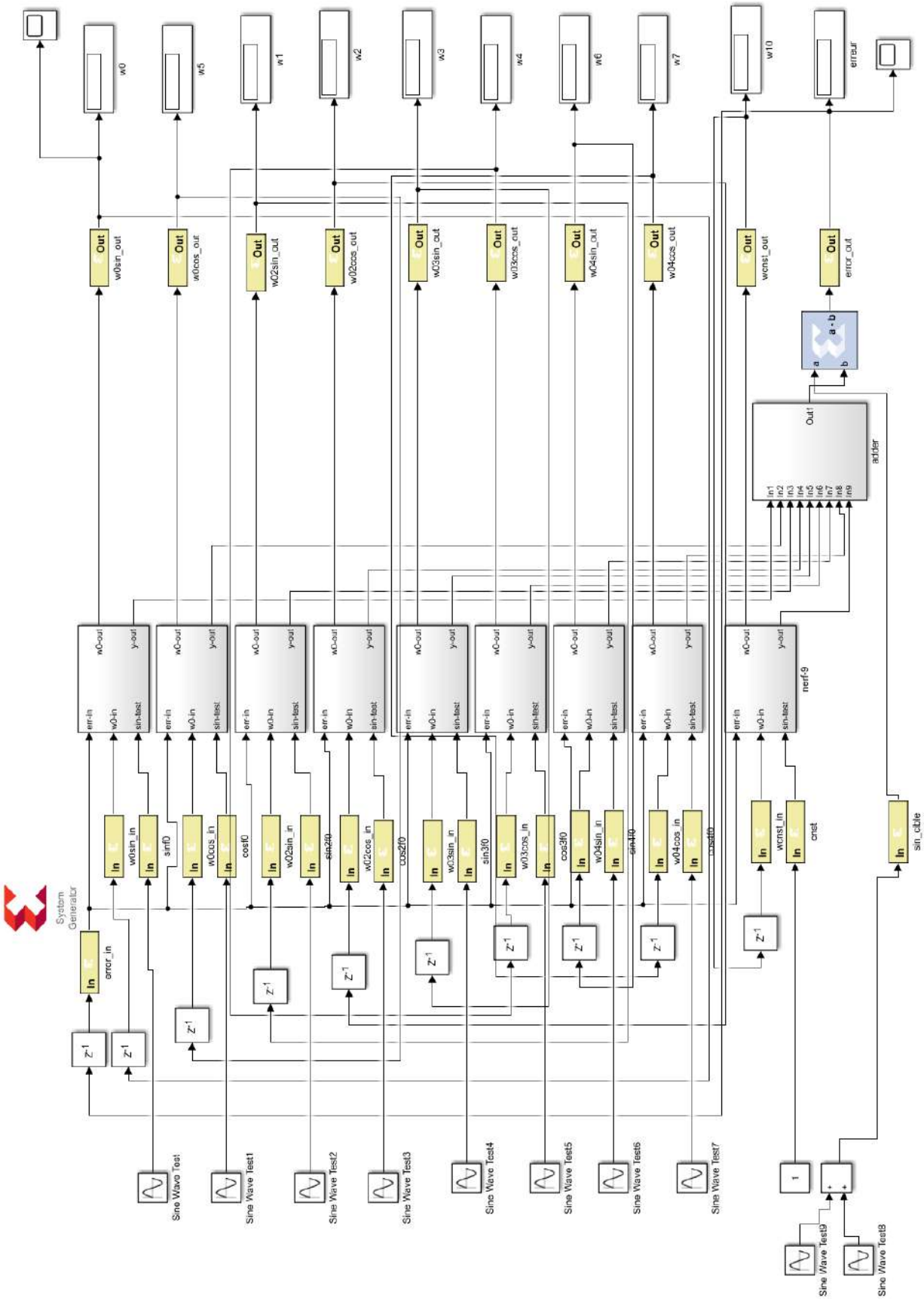


Figure III.10. *Réseau de neurones artificiels sous Xilinx System Generator*

Le contenu des sous-systèmes est donné comme suit :

- Le sous-système ‘Nerf’ : il sert à injecter l’erreur pour corriger les poids dans chaque cycle. Ce sous-système est représenté sur la figure III.11

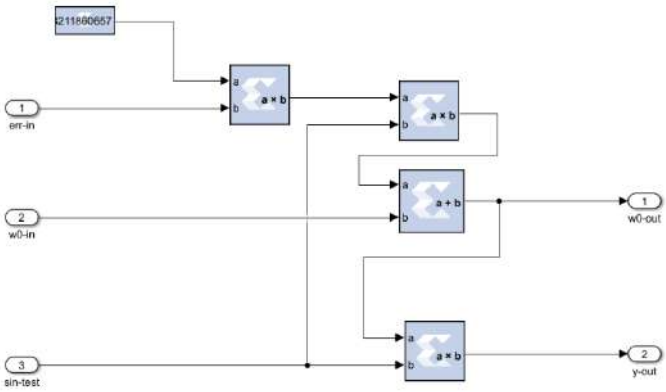


Figure III.11. *Sous-système ‘Nerf’*

- Le sous-système ‘Adder’ a un rôle d’un additionneur à neuf entrés. Ce sous-système est représenté sur la figure III.12

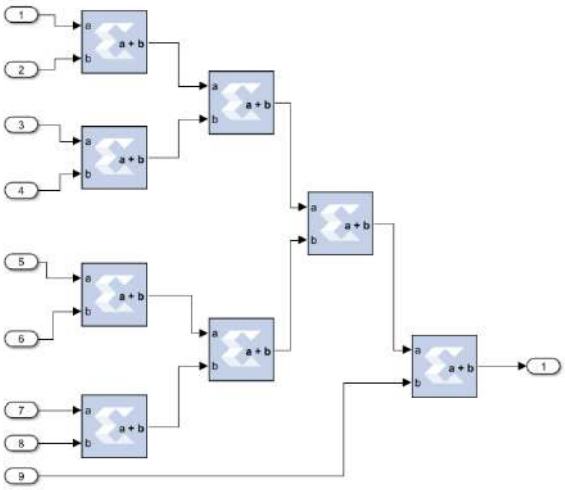


Figure III.12. *Sous-système ‘Adder’*

5.2. Génération du bloc de Co-Simulation

Après la conception de RNA sous Xilinx System Generator on peut également générer un bloc de Co-Simulation en utilisant les caractéristiques suivantes :

- Board: Nexys A7-100T D.0
- Compilation: Hardware Co-Simulation (JTAG)
- FPGA clock period (ns): 10
- Simulink system period (sec): 1/400

On ouvre le jeton System Generator après on met les caractéristiques de bloc, et notre répertoire de travail désiré puis on clique sur 'Generate', voir la figure III.13

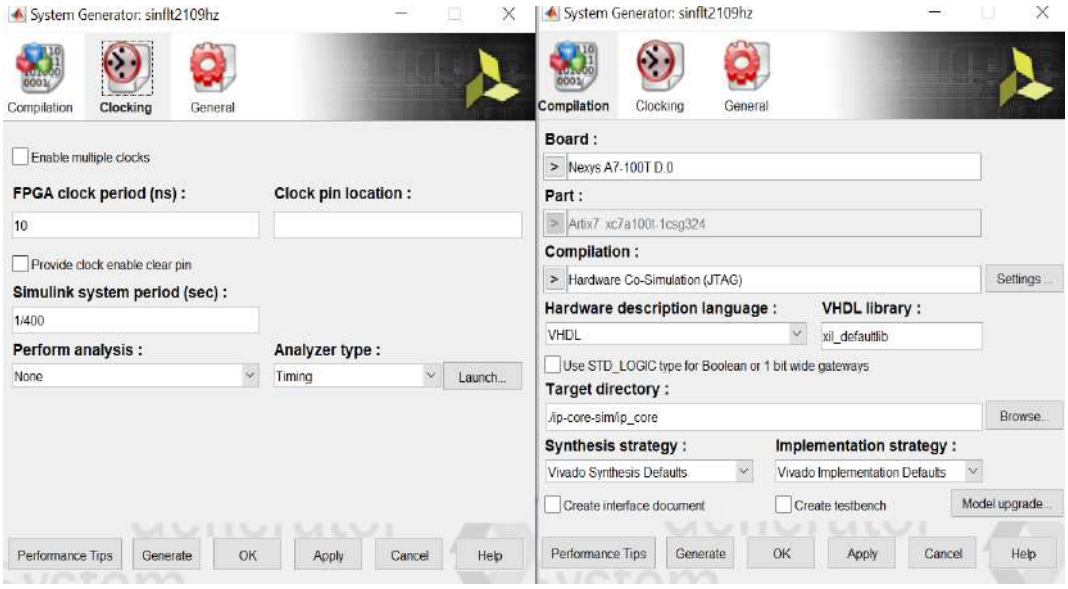


***Figure III.13**. Configuration de Co-Simulation.*

Cette opération regroupe ce projet dans un seul bloc. Ses entrées et ses sorties sont respectivement les entrées et les sorties de notre projet System Generator. La figure III.14 représente le bloc Co-Simulation généré.

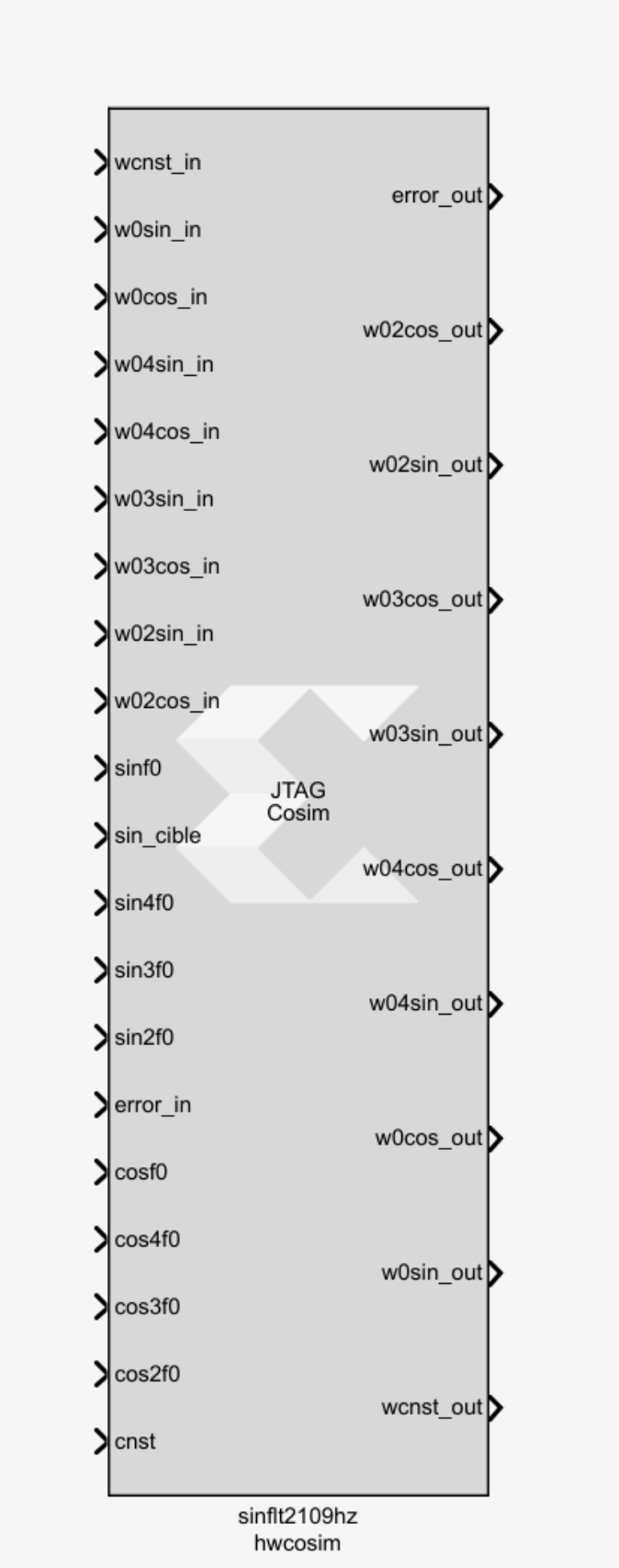
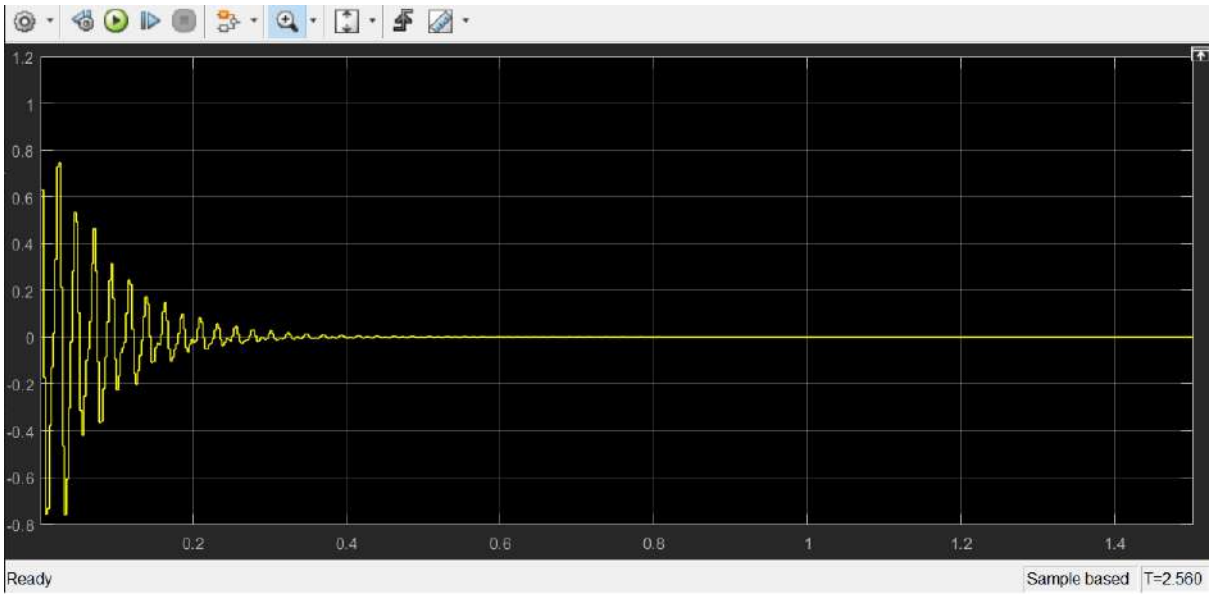
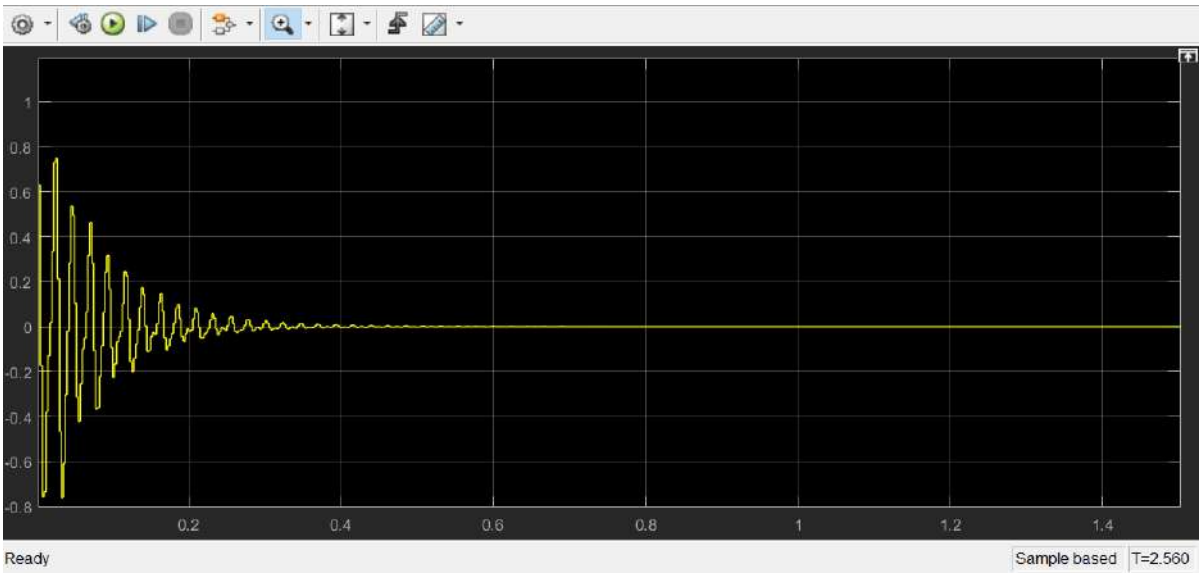


Figure III.14. *Bloc de Co-Simulation*

Après la comparaison des résultats obtenus par le bloc de Co-Simulation et le projet System Generator. Nous avons trouvé presque les mêmes résultats. Les figures III.15. (a et b) présentent respectivement les courbes d'erreur obtenus par le projet System Generator et par le bloc de Co-Simulation avec les mêmes signaux d'entrée.



(a)



(b)

Figure III.15. *Courbe d'erreur: (a) System Generator (b) Co-Simulation.*

5.3. Création de l'IP-Core

Les IP-Cores sont des cœurs de propriétés intellectuelle. Ce sont des blocs ou des modules conçus et testés pour une fonction spécifique. Ces blocs et notre processeur Micro Blaze doivent être interconnectés par un bus AXI, donc nous devons modifier l'interface des entrées (Gateway IN) en AXI4-Lite. (Figure III.16.)

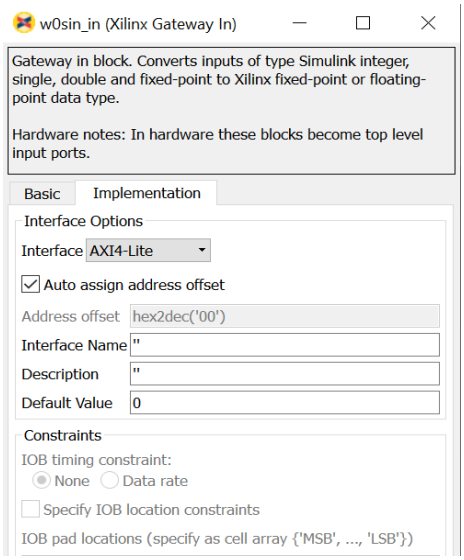


***Figure III.16**. Configuration d'interface AXI*

Pour créer l'IP-Core nous ouvrons le jeton System Generator et nous choisissons les caractéristiques suivantes (voir la figure III.17.) :

- Board: Nexys A7-100T D.0
- Compilation: IP Catalog
- FPGA clock period (ns): 10
- Simulink system period (sec): 1/400

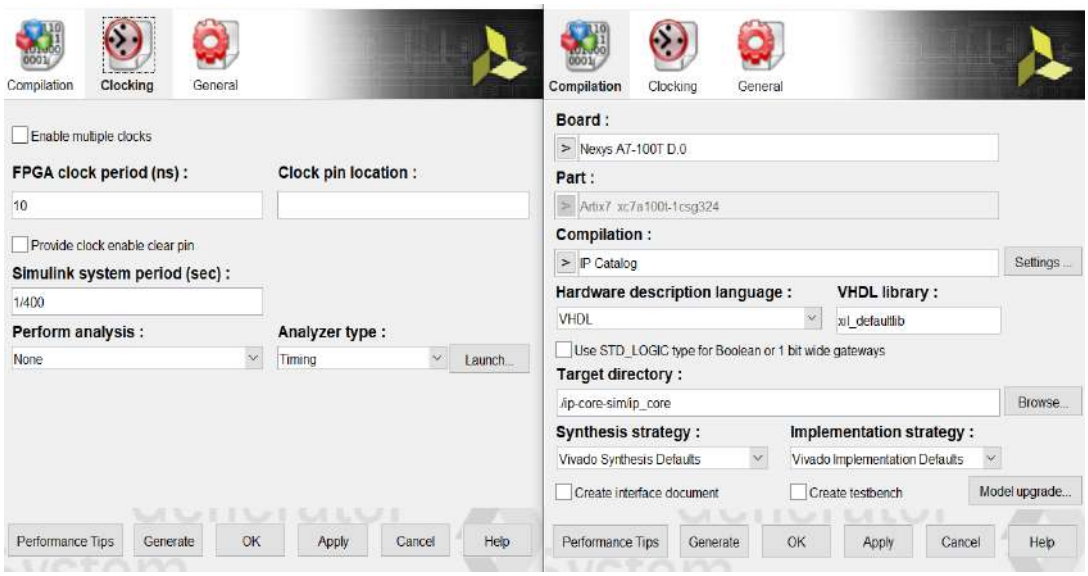


Figure III.17. *Configuration d'IP-Core.*

6. La carte Nexys A7-100T

6.1. Définition

La carte Nexys A7 est une plate-forme de développement de circuits numériques complète et prête à l'emploi, basée sur le dernier Field Programmable Gate Array (FPGA) Artix-7 ™ de Xilinx®. Avec son grand capacité FPGA, ses mémoires externes généreuses et sa collection de ports USB, Ethernet et autres, le Nexys A7 peut héberger des conceptions allant des circuits combinatoires d'introduction aux processeurs embarqués puissants. Plusieurs périphériques intégrés, dont un accéléromètre, un capteur de température, un microphone numérique MEMs, un amplificateur de haut-parleur et plusieurs périphériques d'E / S permettent au Nexys A7 d'être utilisé dans une large gamme de conceptions sans nécessiter d'autres composants.[22]

6.2. Caractéristiques

- Artix-7 FPGA
  - 15 850 Slices de logique programmable, chacune avec quatre LUT à 6 entrées et 8 bascules (* 8 150 slices)
  - 4,860 Kbits de RAM de bloc rapide

- Six tuiles de gestion d'horloge, chacune avec une boucle à verrouillage de phase (PLL)
- 240 slices DSP
- Vitesse d'horloge interne supérieure à 450 MHz
- Convertisseur analogique-numérique interne (XADC) à deux canaux, 1 MSPS

➢ Mémoire

- 128 Mo DDR2
- Flash série
- Fente pour carte microSD

➢ Puissance

- Alimenté par USB ou par n'importe quelle source d'alimentation externe 4.5V-5.5V

➢ USB et Ethernet

- PHY Ethernet 10/100
- Circuit de programmation USB-JTAG
- Pont USB-UART
- Hôte USB HID pour souris, claviers et clés USB

➢ Entrée / sortie utilisateur simple

- 16 commutateurs
- 16 LED
- Deux LED RVB
- Deux afficheurs à 7 segments à 4 chiffres

➢ Audio et vidéo

- Sortie VGA 12 bits
- Sortie audio PWM
- Microphone PDM

➢ Capteurs supplémentaires

- Accéléromètre 3 axes
- Capteur de température

➢ Connecteurs d'expansion

- Connecteur Pmod pour les signaux XADC
- Quatre connecteurs Pmod fournissant un total de 32 entrées / sorties FPGA

Le tableau III.1 et la figure III.18 représentent les composants de la carte Nexys A7-100T.

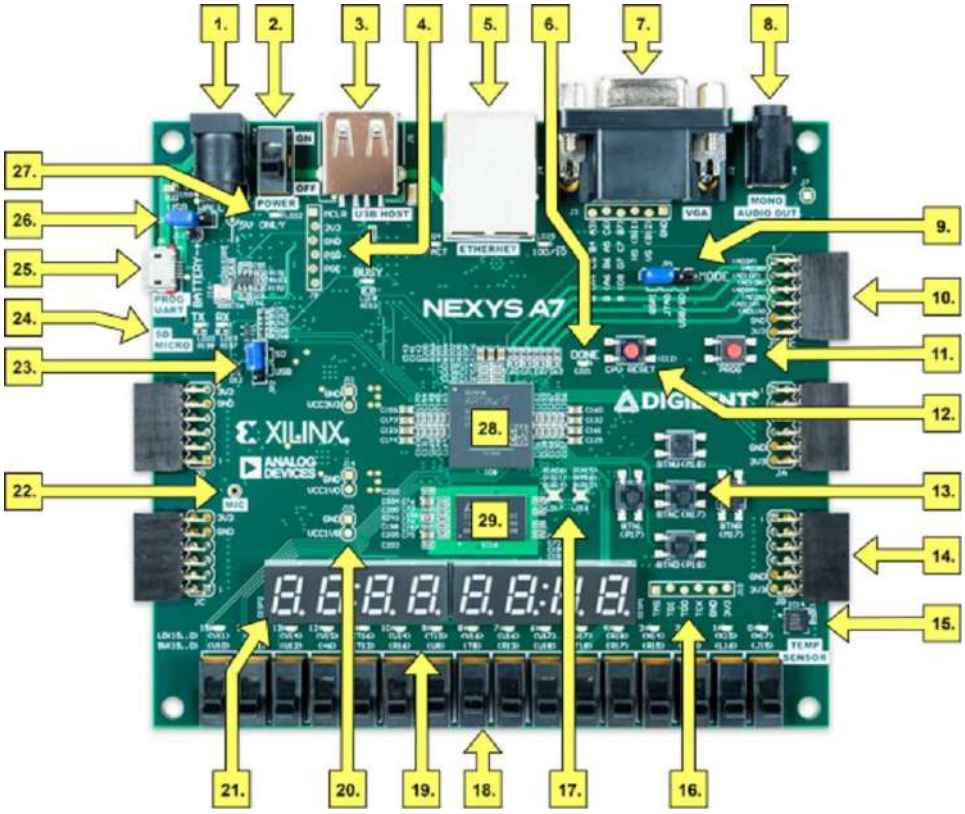


***Figure III.18**. Légende de la fonction Nexys A7-100T*

Tableau III.1. Les composants de la carte Nexys A7-100T

Composant	Description de composant	Composant	Description de composant
1	Prise d'alimentation	16	Port JTAG pour câble externe (en option)
2	Interrupteur d'alimentation	17	LED tricolores (RVB)
3	Connecteur hôte USB	18	Commutateurs à glissière (16)
4	Port de programmation PIC24	19	LED (16)

	(Utilisation d'usine)		
5	Connecteur Ethernet	20	Point (s) de test d'alimentation
6	Programmation FPGA terminée LED	21	Affichage à sept segments de huit chiffres
7	Connecteur VGA	22	Microphone
8	Connecteur audio	23	Cavalier de configuration externe (SD / USB)
9	Cavalier de mode de programmation	24	Logement de carte MicroSD
10	Port Pmod de signal analogique (XADC)	25	Port USB partagé UART / JTAG
11	Bouton de réinitialisation de la configuration FPGA	26	Cavalier de sélection de l'alimentation et tête de la batterie
12	Bouton de réinitialisation de CPU (pour les noyaux souples)	27	Voyant Power-Good
13	Cinq boutons poussoir	28	FPGA Xilinx Artix-7
14	Port (s) Pmod	29	Mémoire DDR2
15	Capteur de température		

7. Vivado Design Suite

Vivado Design Suite est une suite logicielle produite par Xilinx pour la synthèse et l'analyse de conceptions HDL, remplaçant Xilinx ISE avec des fonctionnalités supplémentaires pour le développement de système sur puce et la synthèse de haut niveau. Vivado représente une réécriture et une refonte complète du flux de conception (comparé à ISE) et a été décrit par les critiques comme "bien conçu, étroitement intégré, rapide, évolutif, facile à gérer et intuitif "

Contrairement à ISE qui utilisait ModelSim pour la simulation, Vivado System Edition inclut un simulateur logique intégré. Vivado introduit également la synthèse de haut niveau, avec une chaîne d'outils qui convertit le code C en logique programmable. Vivado a été décrit comme un

« outil EDA complet et à la pointe de la technologie, doté de toutes les dernières nouveautés en termes de modèle de données, d'intégration, d'algorithmes et de performances ». [23]

8. L'implémentation sur la carte FPGA

8.1. Diagramme des blocs

La figure III.19, illustre l'architecture de blocs de réseau de neurones implémenté sur la carte Artix7-100T. Nous ajoutons le bloc de notre IP-Core et le bloc micro blaze, le logiciel Vivado ajoute automatiquement des blocs supplémentaires tel que le bloc d'horloge et le bloc Timer, Ces blocs sont interconnectés automatiquement par Vivado en utilisant des bus AXI.

Après nous générons un bitstream qui sera chargé dans la mémoire BRAM de la carte.

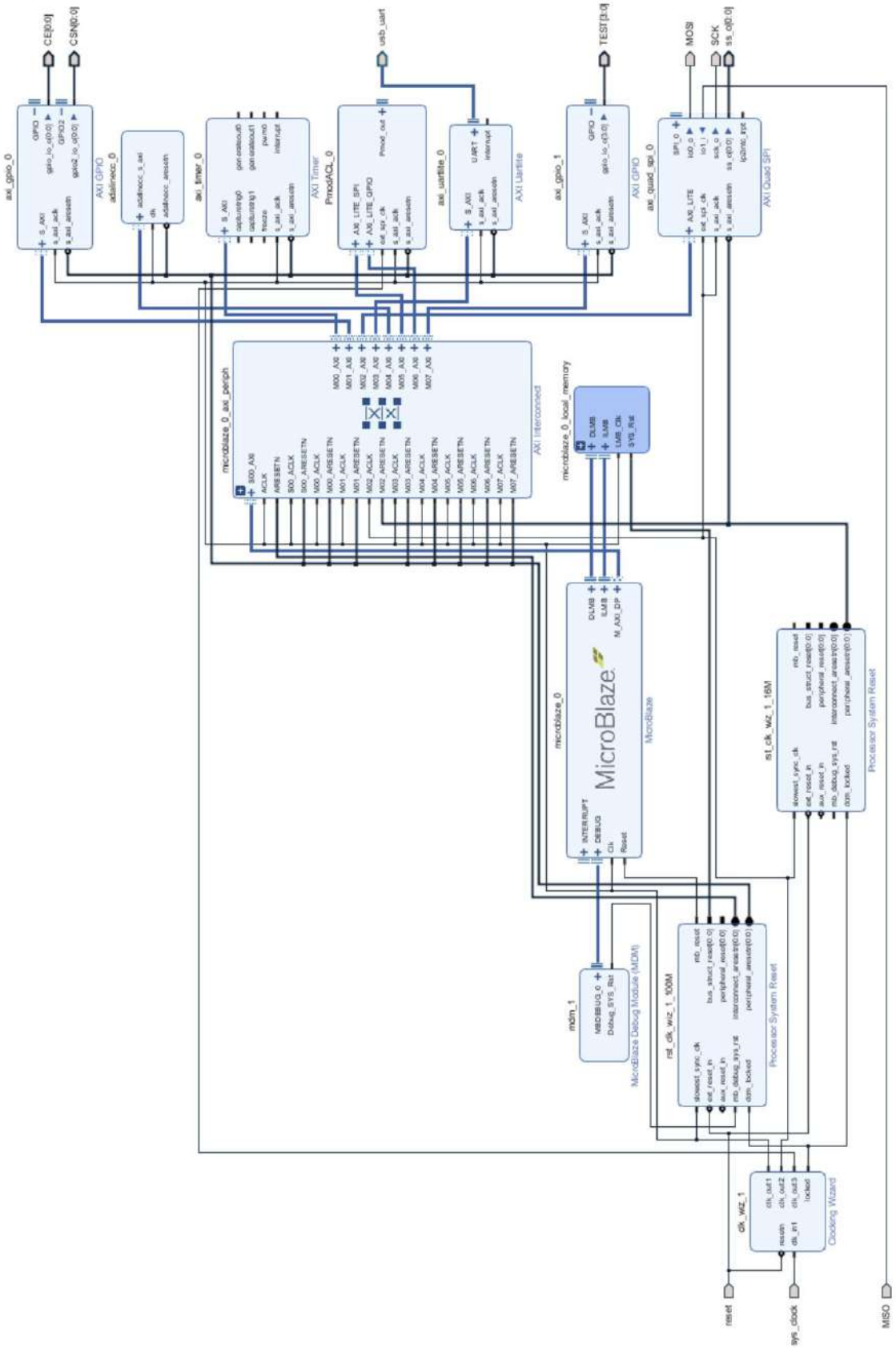


***Figure III.19**. Diagramme des blocs de réseau de neurones*

8.2. Résultats d'implémentation matérielle de réseau de neurones

Le circuit FPGA choisi doit disposer de suffisamment de ressources logiques de blocs mémoires et de slices DSP. Le tableau III.2, résume les résultats de synthèse et d'implémentation d'architecture du réseau de neurones sur la plateforme Nexys A7-100T.

D'après ces résultats, nous pouvons voir que le réseau de neurones a été besoin de presque la moitié (47.41%) des blocs BRAM et moins de 30% des autres ressources. D'un autre coté nous avons pu arriver à faire fonctionner le réseau de neurones à un niveau acceptable de puissance.

Tableau III.2. Résultats de synthèse et d'implémentation de réseau de neurones

Ressources	**Disponible**	**Utilisé**	**Utilisation (%)**
Slice LUT	63400	13733	21.66
Bascules (FlipFlop)	126800	4136	3.26
Bloc RAM	128	64	47.41
Slices DSP	240	72	30
Blocs IO	210	22	10.476
Puissance totale estimée sur puce		301 mW	
Fréquence max du MicroBlaze	100 MHz		

9. Configuration FPGA

Après la mise sous tension, le FPGA Artix-7 doit être configuré (ou programmé) avant de pouvoir exécuter toute fonction. Nous pouvons configurer le FPGA de l'une des quatre manières suivantes :

- Un PC peut utiliser le circuit Digilent USB-JTAG (port J6, intitulé « PROG ») pour programmer le FPGA à tout moment de la mise sous tension.
- Un fichier stocké dans le périphérique flash série non volatile (SPI) peut être transféré vers le FPGA à l'aide du port SPI.
- Un fichier de programmation peut être transféré sur le FPGA à partir d'une carte micro SD.

- Un fichier de programmation peut être transféré depuis une clé USB connectée au port USB HID.

La figure III.20, Représente les différentes options disponibles pour la configuration du FPGA. Un cavalier « mode » intégré (JP1) et un cavalier de sélection de support (JP2) permettent de sélectionner les modes de programmation.

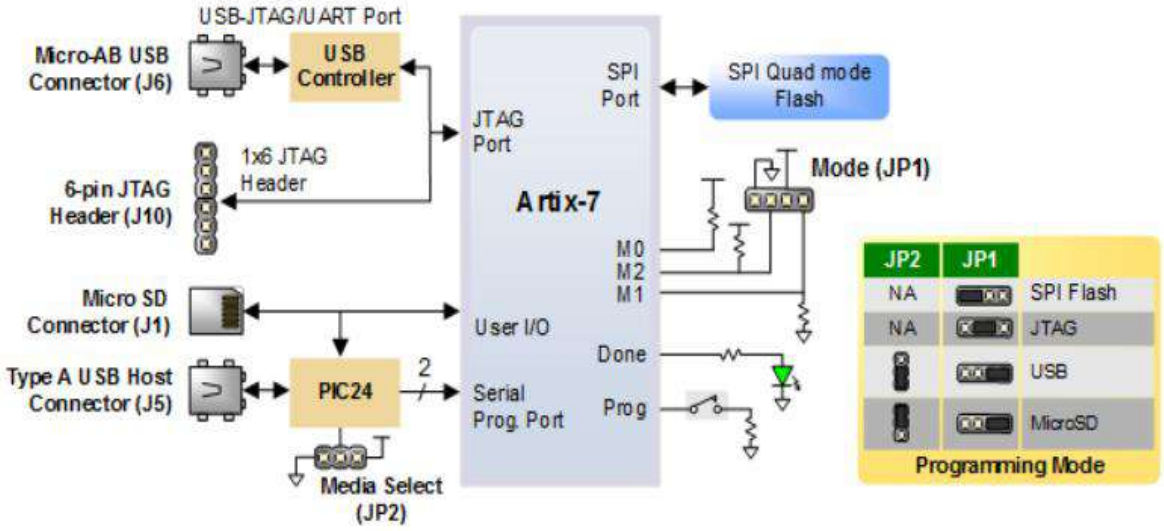


Figure III.20. *Options de configuration du Nexys A7 DDR*

9.1. Bitstream

Les données de configuration FPGA sont stockées dans des fichiers appelés bitstreams d'extension de fichier : «.bit ». Les logiciels ISE ou Vivado de Xilinx peuvent créer des Bitstreams à partir de fichiers sources VHDL, Verilog ou schématiques (dans le jeu d'outils ISE, EDK est utilisé pour les conceptions à base de processeur intégré MicroBlaze).

Les bitsreams sont stockés dans des cellules de mémoire basées sur la mémoire SRAM au sein du FPGA. Ces données définissent les fonctions logiques et les connexions de circuit du FPGA. Elles restent valables jusqu'à ce qu'elles soient effacées en coupant l'alimentation de la carte, en appuyant sur le bouton de réinitialisation attaché à l'entrée PROG ou en écrivant un nouveau fichier de configuration à l'aide du port JTAG.

Le temps nécessaire à la programmation du Nexys A7 peut être réduit en compressant le bitstream avant la programmation, puis en permettant au FPGA de décompresser le bitstream lui-même lors de la configuration. En fonction de la complexité de la conception, des taux de

compression de 10x peuvent être atteints. La compression de bitstream peut être activée dans les outils Xilinx (ISE ou Vivado) pour se produire lors de la génération. Après avoir été programmé avec succès, le FPGA provoquera le voyant “DONE” à éclairer. Nous appuyons sur le bouton “PROG” à tout moment pour réinitialiser la mémoire de configuration du FPGA. Après avoir été réinitialisé, le FPGA tentera immédiatement de se reprogrammer à partir de la méthode sélectionnée par les cavaliers en mode de programmation.

9.2. Le logiciel SDK

Le kit de développement logiciel (SDK) Xilinx® est un environnement de développement intégré (IDE) pour le développement d'applications logicielles intégrées destinées aux processeurs intégrés de Xilinx. Le SDK fonctionne avec les conceptions matérielles créées avec Vivado Design Suite. Le SDK est basé sur le standard open source Eclipse. Les fonctionnalités du SDK incluent :

- Editeur de code C / C ++ riche en fonctionnalités et environnement de compilation
- Gestion de projet
- Configuration de construction de l'application et génération automatique de Makefile
- Erreur de navigation
- Environnement bien intégré pour le débogage et le profilage en continu des cibles intégrées
- Contrôle de version du code source
- Analyse de la performance au niveau du système
- Outils spéciaux focalisés pour configurer le FPGA.
- Programmation Flash
- Outil de ligne de commande scriptable [24]

9.3. Configuration JTAG

Les outils Xilinx communiquent généralement avec les FPGA à l'aide de l'architecture Test Access Port et Boundary-Scan, généralement appelée JTAG. Pendant la programmation JTAG, un fichier « .bit » est transféré du PC vers le FPGA à l'aide du circuit Digilent USB-JTAG intégré (port J6) ou d'un programmeur JTAG externe, tel que le Digilent JTAG-HS2, connecté au port J10. Nous pouvons exécuter la programmation JTAG à tout moment après la mise en

sous tension du Nexys A7, quel que soit le type de cavalier défini pour le cavalier de mode (JP1). Si le FPGA est déjà configuré, la configuration existante est remplacée par le bitstream transmis via JTAG. Le paramétrage du cavalier de mode sur le paramètre JTAG (illustré à la figure III.20) est utile pour empêcher la configuration du FPGA à partir d'une autre source de bitstream jusqu'à ce qu'une programmation JTAG ait lieu.
La programmation du Nexys A7 avec un bitsrteam non compressé à l'aide du circuit USB-JTAG intégré prend généralement environ cinq secondes. La programmation JTAG peut être effectuée à l'aide du serveur matériel de Vivado ou de l'outil iMPACT inclus avec ISE et de la version Lab Tools de Vivado.

9.4. L'organigramme d'exécution sur la carte

La figure III.21 représente l'organigramme d'exécution sur la carte Nexys A7-100T en séparant les instructions qui s'exécutent au niveau d'IP-Core et au niveau de Micro Blaze.

Tout d'abord le Micro Blaze initialise les signaux d'entrée, l'erreur et les poids (w), après il les injecte dans l'entrée de l'IP Core ainsi que le premier élément de vecteur cible. L'IP Core calcule les nouveaux poids (w) et l'erreur à partir des données injectées, les nouveaux poids et l'erreur calculés seront réinjectés en prochain cycle dans l'entrée de l'IP Core ainsi que le prochain élément de vecteur cible.

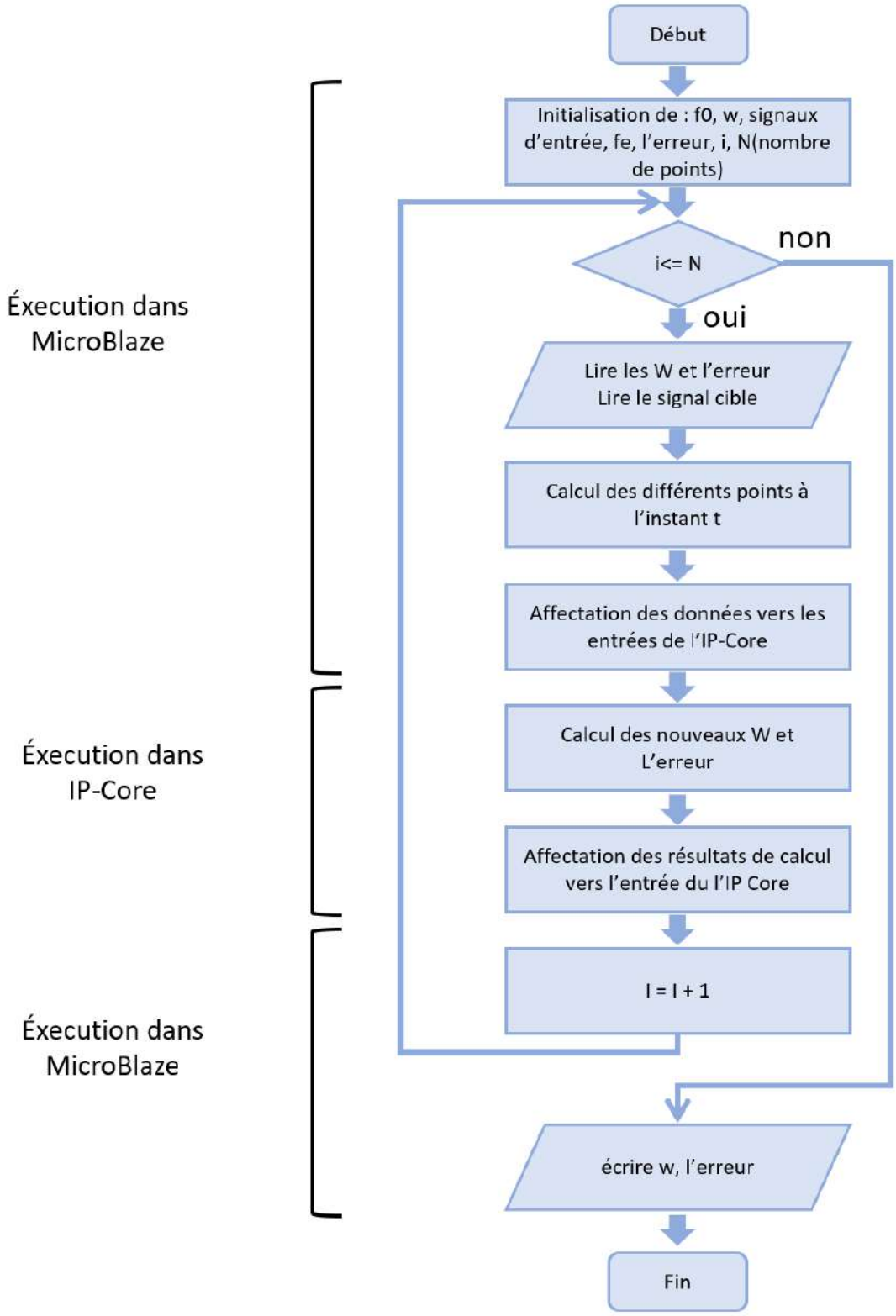


Figure III.21. *Organigramme d'exécution sur IP-Core et MicroBlaze*

10. Test de résultats dans la carte

Pour exécuter ce projet sur la carte Nexys A7-100T. nous devons utiliser un logiciel pour charger le bitstream sur la carte et tester le projet, dans notre cas nous avons choisi SDK de Xilinx.

10.1. La configuration sur SDK

Premièrement nous connectons la carte avec le PC ; ensuite nous chargeons le bitstream sur la mémoire BRAM de la carte (figure III.22.)

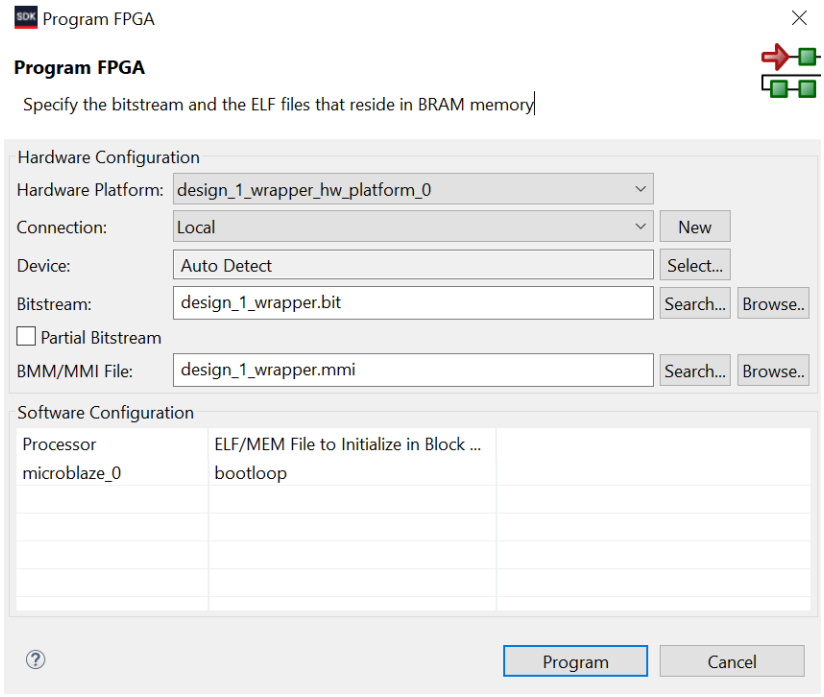


Figure III.22. *Chargement de bitstream sur la carte*

10.2. Résultats d'exécution sur la carte

Dans l'exécution nous devons prendre un signal cible pour assurer le bon fonctionnement de notre réseau de neurones, dans notre cas nous avons importé un vecteur depuis Matlab, ce vecteur est généré à partir d'une somme de cinq signaux cosinus et sinus de différentes amplitudes, et de différentes fréquences, ainsi que la composante continue ; en ajoutant à ce signal un bruit aléatoire. En prendre en considération la fréquence d'échantillonnage (Fs) et la

fréquence fondamentale (F0) pour les utiliser dans la génération des signaux d'entrées de réseau de neurones. La formule du signal bruit et du signal cible est donne comme suit :

$$Bruit=(2*rand(1,nfft)-1)*10 \quad (3.1)$$

$$SignalCible=60+75*sin(2*pi*f0*t)+50*cos(2*pi*f0*t)+100*cos(2*pi*2*f0*t)+25*sin(2*3*pi*f0*t) \quad (3.2)$$

Où 'nfft' est le nombre d'échantillons.

La (figure III.23.) et la (figure III.24.) représentent respectivement la forme du signal cible ainsi que son spectre (sa transformée de Fourier rapide).

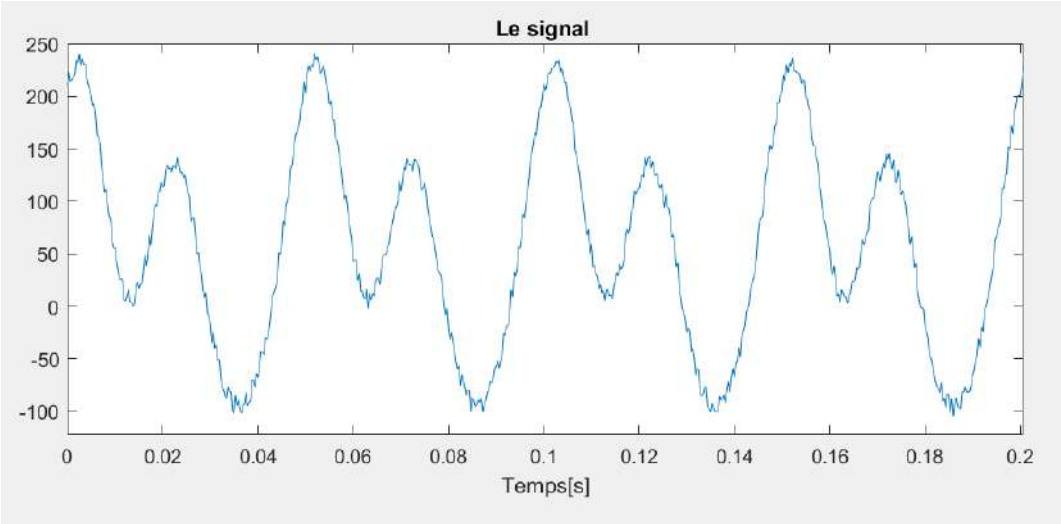


Figure III.23. *Le graphe du signal cible*

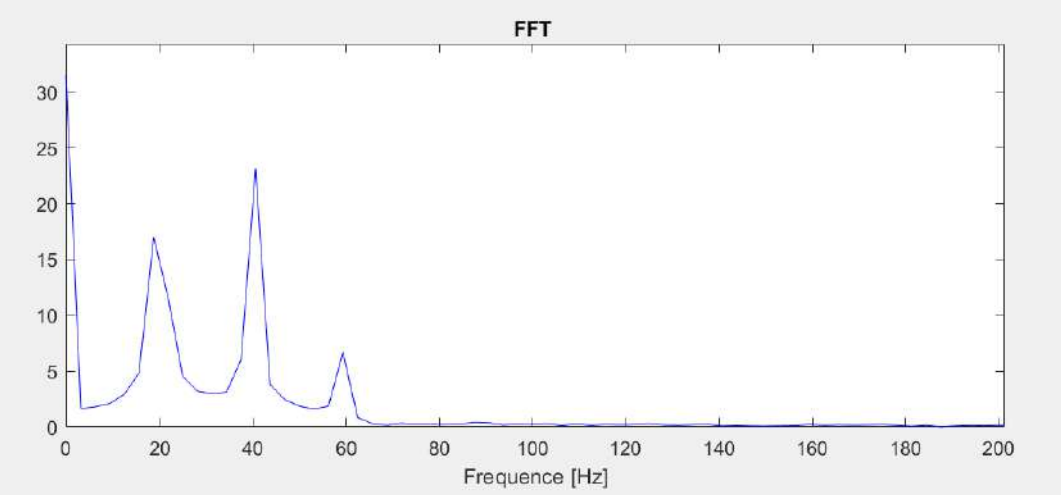


Figure III.24. *Le spectre du signal cible*

Nous connectons la carte à un serial port de notre PC et nous sélectionnons ‘Exécuter en tant que : Lancer sur matériel’.

10.2.1. Discutions de résultats obtenus

Après l’exécution nous remarquons que les valeurs obtenues des signaux composants sont plus proches aux amplitudes des signaux composants du signal cible avec une faible valeur d’erreur (-0.310471).

Nous voyons que les résultats des fréquences composantes du signal cible sont aussi presque les mêmes valeurs pris du spectre du signal cible avec Matlab.

Nous pouvons dire aussi que le temps d’apprentissage était court (7.478 ms). (Figure III.25)

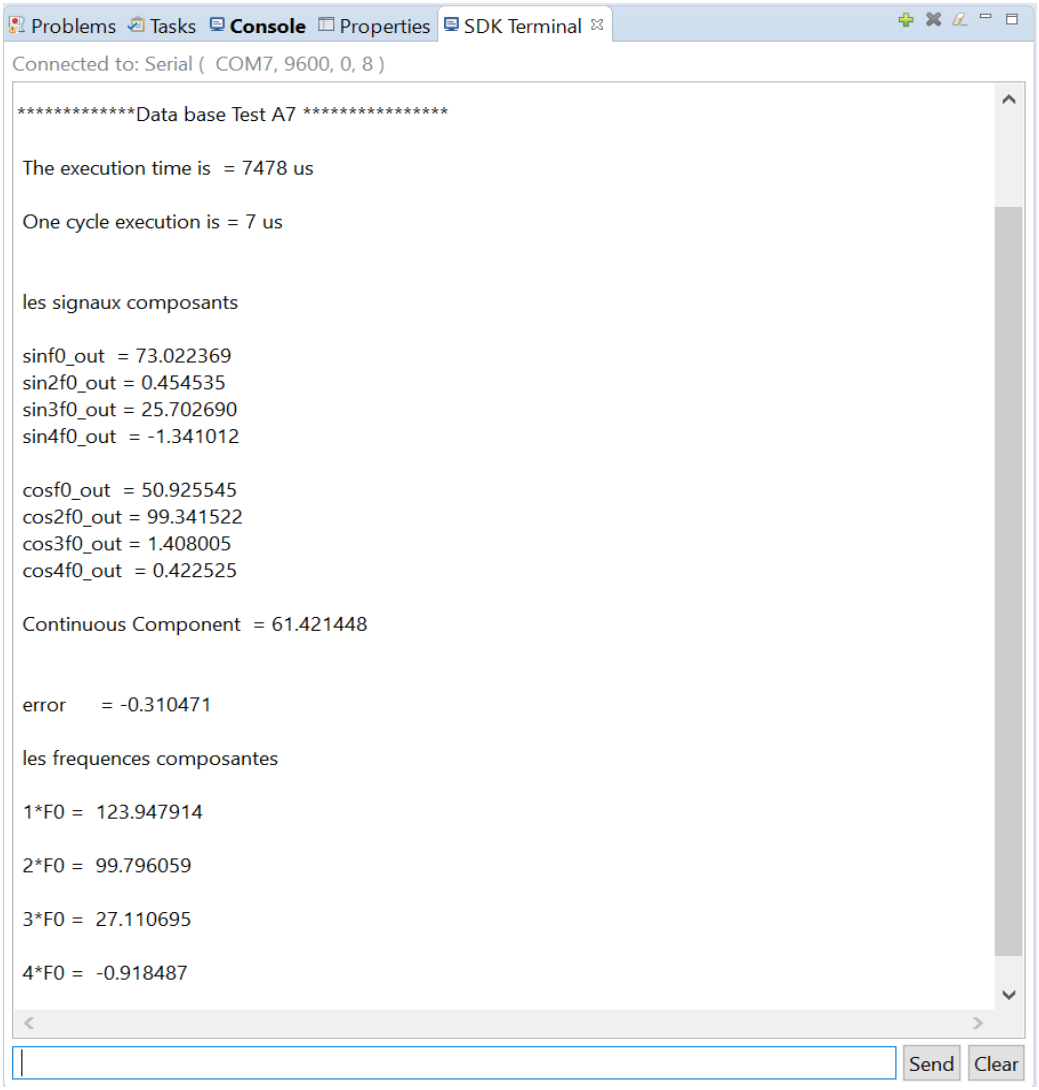


Figure III.25. *Résultats d’exécution sur la carte*

11. Conclusion

Dans ce chapitre, nous avons présenté une conception matérielle d'un réseau de neurones artificiel de type ADALINE qui nous a permis de mettre en œuvre un système qui détecte les fréquences des harmoniques et leurs amplitudes.

Au cours de ce chapitre, nous avons simulé un réseau de neurones artificiel sur Matlab, puis nous avons conçu ce réseau sur System Generator de Xilinx afin de générer un IP-Core, ensuite nous avons connecté ce dernier avec un processeur embarqué Micro Blaze dans une architecture matérielle, après nous avons implémenté cette architecture sur une carte FPGA de type Nexys A7-100T, enfin nous avons vérifié nos résultats expérimentaux sur la carte.

Conclusion générale

Les réseaux de neurones ont prouvé leurs efficacité grâce à leur puissance et à leur rapidité d'apprentissage et ceci en donnant des meilleurs résultats de fonctionnement. Ce sont en effet des mécanismes d'autoapprentissage qui ne nécessitent pas les compétences traditionnelles d'un programmeur.

L'implémentation sur circuits reconfigurables FPGA est un sujet d'actualité, mais d'une très grande complexité dans le domaine de recherche scientifique, car il nécessite non seulement une maitrise des technologies relatives aux FPGA mais aussi une très bonne connaissance des applications et leurs environnements.

L'objectif de ce projet était la conception et l'implémentation d'un IP-Core pour un réseau de neurones de type ADALINE qui sert à la détection des harmoniques des signaux à base d'une architecture embarquée sur un circuit FPGA. Nous avons choisi la carte Nexys A7-100T de Xilinx, qui comprend une quantité suffisante des ressources logiques qui sont généralement requises dans notre application.

Cette étude a été structuré autour de trois chapitres essentiels, le premier consacré au domaine de réseaux de neurones, cette partie a traité quelques définitions de réseaux de neurones artificiels, ainsi que leurs différentes architectures et leurs types d'apprentissage, et pour expliquer le fonctionnement de ses algorithmes nous avons traité quelques exemples avec Matlab (avec deux méthodes : toolbox et script). Dans le second, nous avons parlé sur les circuits programmables FPGA, Nous avons commencé par une définition sur les FPGAs, ces familles et ces caractéristiques cette partie a été suivi de quelques descriptions sur le processeur MicroBlaze. Dans le dernier nous avons conçu et implémenté un IP-Core pour un réseau de neurones de type ADALINE sur une plateforme FPGA.

En fin, on peut conclure que l'implémentation des fonctions sur FPGA est la plus simple car elle utilise non seulement un langage évolué tel que le VHDL, mais aussi le logiciel MATLAB nous offre la possibilité de simuler n'importe quelle fonction sur Simulink, en utilisant la libraire de Xilinx (System Generator) pour que cette dernière soit directement transformée en langage VHDL.

Références bibliographiques :

[1] TOUZET**,** Claude. Les réseaux de neurones artificiels, Cours, exercices et travaux pratiques :" Introduction au connexionnisme **"**,129p :1992

[2] MUXIKA OLASAGASTI, Efiaut." Application des Réseaux de Neurones a l'Identification d'un Axe de Machine-Outil ", Thèse de doctorat, INPG : 2002

[3] Bernard GOSSELIN," Application de réseaux de neurones artificiels a la reconnaissance au de caractères manuscrits " 231p. Thèse de doctorat. Faculté Polytechnique de Mons : 95-96

[4] DREYFUS, Gérard.**"** Neural networks, méthodology and applications " .509p.paris :2002

[5] Principe de fonctionnement d'un neurone formel. https://www.editionseni.fr/open/mediabook.aspx?idR=2dac0ec804e7e66899c47f486d228ff0

[6] S.ZEGHLACHE, **"** Commande Intelligente **"**, cours, université Mohamed boudiaf-m'sila.2008

[7] PETITJEAN**,** Gérald. Cours, réseaux de neurones :" Introduction aux réseaux de neurones " 102p

[8] Mme BABA HAMED Amel née BENSEMAIN : "Vers Un Modèle De Classification Neuronale Des Données Médicales A Base De La Technologie FPGA".173p. thèse de doctorat. Université Aboubakr Belkaïd Tlemcen : 2017

[9] BUYSSENS, Pierre. Traitement des images : " Fusion de différents modes de capture pour la reconnaissance du visage appliquée aux e_transactions " 242p. Thèse de doctorat. Université de Caen Basse-Normandie : 2011

[10] Cyrille FAUCHEUX, Alexandre LISSY. Rapport de projet " Réseaux de neurones sous Matlab ", École Polytechnique de l'Université de Tours,

[11] Marc PARIZEAU " Réseaux de neurones ". université Laval 124p :2004

[12]: Donald G. Bailey .Design for embedded image processing on FPGAs. Massey University, New Zealand

[13]: Field Programmable Gate Arrays and Applications

[14]:Xilinx.Vivado Microblaze reference :2018
[https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug984-vivado-microblaze-ref.pdf]
[15] BENGHERBIA, Billel. « Noeud de capteurs sans fil (WSN) dans une architecture embarquée de type FPGA ». 151. Thèse de doctorat. : électronique. Universite de Blida : 2018

[16]: Using Xilinx System Generator for Real Time Hardware Co-simulation of Video Processing System Chapter in Lecture Notes in Electrical Engineering · April 2010

[17]: S. Mittal, S. Gupta, and S. Dasgupta, "System Generator: The State-Of-Art FPGA Design Tool For DSP Applications", Third International Innovative Conference On Embedded Systems, Mobile Communication And Computing (ICEMC2 2008), Global Education Center, Infosys.

[18]: T. Saidani, D. Dia, W. Elhamzi, M. Atri and R. Tourki, "Hardware Co-simulation For Video Processing Using Xilinx System Generator" Proceedings of the World Congress on Engineering 2009, London, U.K.

[19]: Real Time Hardware Co-simulation of Edge Detection for Video Processing System Yahia Said, Taoufik Saidani, Fethi Smach and Mohamed Atri Laboratory of Electronics and Microelectronics Faculty of Sciences

[20]: Xilinx System Generator v2.1 Basic Tutorial, 2015, Xilinx

[21]: System Generator for DSP User Guide UG640 (v11.4), Decembre 2009

[22]: Programmable-Logic: Nexys-A7: Reference-Manual
https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual]

[23]: Vivado Design Suite and User Guide, Release Notes, Installation, and Licensing: 2018, Xilinx

[24]: SDK Getting Started, Xilinx, 2018