République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université ABDERRAHMANE MIRA Bejaia

Faculté de Technologie

Département de Génie Electrique

Mémoire de Master

Filière : Électronique
Option Automatique

Thème : Implementation matérielle de réseau de neurones probabiliste pour une interface myoélectrique embarquée

Présenter par : BOUFALA Nabil

Jury :

Président :	Mohamed SABI
Promotrice :	Samia MEZZAH
Examinateurs :	Mohamed ABDI

Année : 2012/2013

Dédicace

A mes très chère parents.
A mon petit frère et mes soeurs.
A toute personne chère a mes yeux en particulier Mounira.
Je dédie ce modeste travail.

Nabil

Remerciements

Je remercie DIEU tout puissant, pour nous avoir donné le courage et la force pour terminer ce travail.

Le plus grand remerciement est réservé à ma promotrice M^{elle} MEZZAH Samia Pour l'aide et le soutien qu'elle ma apporté tout au long de ce travail.

J'adresse mes chaleureux remerciements aux membres du jury qui ont accepté d'évaluer ce travail.

Je tiens a remercie tout particulièrement le professeur Adrian Chan pour nous avoir fournie la base de données des signaux Myo-électriques.

Pour terminer je tiens à remercier tous mes collègues et mes amis qui m'ont aidé et qui m'ont apporté leur soutien moral ainsi que tous ceux qui ont participé de près ou de loin à l'élaboration de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Résumé

Les muscles squeletique génère une activité electrique pendant leurs contractions. Pouvoir reconnaître un signal EMG d'un autre peut être utile à la détection de pathologies, de mouvements et servir à un diagnostic clinique ou à la commande d'une prothèse.

C'est pourquoi nous nous sommes intéressés au développement d'un classifieur à base de réseaux de neurones probabiliste destiné à la classification de ces signaux et ainsi reconnaitre 7 mouvement different de la main. Pour obtenir un classifieur matériles performant et portable, integrable à une interface embarquée pour la commande myoélectrique d'une prothèse de la main, nous avons choisi d'implementer le classifieur sur un circuit logique programmable de type FPGA (field-programmable gate array).

Mots clefs
Signaux SEMG
Reseaux de neurones
Classification
FPGA
Implementation

Introduction Générale

Les muscles squelettiques génèrent une activité électrique pour réaliser les mouvements liés aux activités humaines. Cette activité électrique peut être détectée et observée par électromyographie de surface. L'analyse du signal életromyographique (SEMG) obtenu peut conduire à de nombreuses applications, parmi les quelles figure le contrôle de prothèses et la détection de pathologies musculaires. Les signaux SEMG d'un bras semble être un vecteur informatif sur les mouvements de celui-ci et ceux de la main[1][2]. Notre travail c'est orienté vers la classification des signaux SEMG collectés à la surface de l'avant bras afin d'identifier sept mouvements différents de la main.

En général, nous pouvons décomposer l'ensemble de la procédure de classification en trois phases : La 1^{ere} phase de traitement concerne le choix des paramètres de caractérisation. La caractérisation en vue d'une classification est menée de telle sorte qu'elle dispose au final d'un fort potentiel de discrimination de classes.

La 2^{eme} partie du traitement est la réduction de l'espace de représentation. En effet, plusieurs études ont montré la nécessite de la réduction de la dimension de l'espace de représentation pour une meilleure classification (les données les plus discriminants seront sélectionnées)[1][3].

La dernière étape est la classification proprement dite. La classification peut être réalisée en utilisant plusieurs méthodes qui sont regroupées en deux catégories : les méthodes supervisées et les méthodes non supervisées.

Dans le cadre de notre travail, nous nous intéressons au dimensionnement d'un classifieur neuronal supervisé et à son implémentation sur un circuit logique programmable de type FPGA (field-programmable gate array). L'objectif est d'obtenir un classifieur matériel performant et portable, intégrable à une interface embarquée pour la commande myoélectrique d'une prothèse de la main.

Ce présent mémoire est organisé en 4 chapitres :

Le premier chapitre est consacré aux réseaux de neurones. Dans ce contexte, nous présenterons les différents types de réseaux de neurones et nous nous pencherons sur deux architectures utilisées pour la classification. Enfin de ce chapitre nous pré-

senterons quelques outils permettant de créer, d'entraîner et de simuler les réseaux de neurones.

Le chapitre 2 est consacré à la classification des signaux SEMG. Nous y discuterons l'origine et les caractéristiques de ces signaux, ainsi que les divers paramètres permettant de les caractériser dans les deux domaines fréquentiel et temporel. La dernière partie de ce chapitre est consacrée à la présentation de différentes méthodes supervisées et non supervisées utilisées en classification.

Le chapitre 3 est consacré à la réalisation de différentes études qui nous permettrons de choisir les paramètres qui caractérise le mieux les signaux SEMG, l'architecture du réseau neuronal ainsi que la dimension de l'espace des paramètres à utiliser.

Le chapitre 4 est dédié à l'implémentation matérielle sur FPGA de la structure neuronal choisi dans le chapitre 3.

CHAPITRE 1

Réseaux de neurones et classification

1.1 Introduction

Le développement des réseaux de neurones artificiels est issu d'une volonté de l'homme de comprendre et d'imiter les capacités du cerveau. Intelligence, apprentissage, mémoirisation, traitement parallèle massif d'informations et flexibilité sont, autant de qualités attribuées au cerveau, recherchées pour la synthèse des différents systèmes artificiels intéligents et complexes[4].

Les réseaux de neurones sont des outils très performants, qui peuvent être définis par leurs structures, leurs variables descriptives , et les interactions des composants.

1.2 Neurone formel

Dans un reseau de neurones artificiel, chaque neurone est un processeur élémentaire (figure 1.1). Il reçoit un nombre variable d'entrées en provenance de neurones amonts. A chacune de ces entrées est associée un poids (weight) représentatif de la force de la connexion. Le neurone est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals[5].

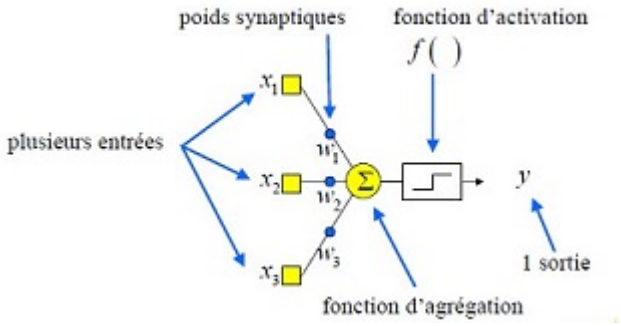

Figure 1.1 : *Neurone formel*

Le traitement réalisé par un neurone comprend deux opérations simples : le calcul de la somme pondérée de ses entrées et l'application d'une fonction d'activation sur le résultat de cette somme.

$$S = \sum_{i=1}^{n} w_i.x_i + b_i$$

$$y = f(S)$$

avec :

- x_i : composantes du vecteur d'entrée.
- w_i : composantes du vecteur poids synaptiques.
- b_i : biais.
- S : somme pondérée appeler potentiels.
- f(.) : fonction d'activation.
- y : sortie du neurone.

Différentes fonctions peuvent être utilisées comme fonction d'activation d'un neurone comme le montre la figure 1.2. Les plus utilisées sont les fonctions «seuil» , «linéaire», «sigmoïde» et «tangente hyperbolique».[6]

Non de la fonction	Relation entrée/sortie	Nom sous Matlab
Seuil	Y=0 si S<0 Y=1 si S>=0	hardlim
Seuil symétrique	Y=-1 si S<0 Y=1 si S>=0	hardlims
Linéaire	Y=S	purelin
Linéaire saturée	Y=0 si S<0 Y=S si 0<=S<=1 Y=1 si S>1	satlin
Linéaire saturée symétrique	Y=-1 si S<0 Y=S si -1<=S<=1 Y=1 si S>1	satlins
Linéaire positive	Y=0 si S<0 Y=S si S>=0	poslin
Sigmoïde	$y = \frac{1}{1 + exp^{-S}}$	logsig
Tangente hyperbolique	$y = \frac{e^{S} - e^{-S}}{e^{S} + e^{-S}}$	tansig
compétitive	Y=1 si S maximum Y=0 autrement	compet

Figure 1.2 : *Fonction de transfert*

1.3 Réseaux de neurones

Les réseaux de neurones artificiels sont constitués de plusieurs couches (d'entrée, de sortie et cachées) de neurones fortement connectés et fonctionnant en parallel. En général, On distingue deux structures de réseau :

- Les réseaux non bouclés (*Statique*).
- Les réseaux bouclés (*Dynamique*).

1.3.1 Les réseaux de neurones non bouclés

Un réseau de neurone non bouclé réalise une (ou plusieurs) fonction algébrique de ses entrées par composition des fonctions réalisées par chacun de ses neurones. Dans un tel réseau, le flux d'information circule des entrées vers les sorties sans retour en arrière[7][8].
Il existe deux types de réseaux de neurones : les réseaux complètement connectés et les réseaux à couches.

les réseaux complètement connecté Dans ce tye de réseaux, les entrées puis les neurones (cachés et de sortie) sont numérotés et pour chaque neurone :

- Ses entrées sont toutes les entrées du réseau ainsi que les sorties des neurones de numéro inférieur.
- Sa sortie est connectée aux entrées de tous les neurones de numéro supérieur.

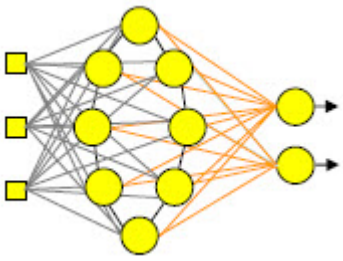

Figure 1.3 : *Réseau de neurones complètement connecté*

les réseaux de neurones à couches Dans une architecture de réseaux à couches, les neurones cachés sont organisés en couches, les neurones d'une même couche n'étant pas connectés entre eux. De plus les connexions entre deux couches de neurones non consécutives sont éliminées. Une telle architecture est historiquement très utilisée, surtout en raison de sa pertinence en classification[3].

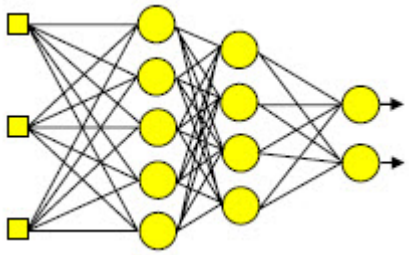

Figure 1.4 : *Réseau de neurones à couche*

Dans un réseau de neurones non bouclé, le temps ne joue aucun rôle fonctionnel : si les entrées sont constantes, les sorties le sont également. Le temps nécessaire pour le calcul de la fonction réalisée par chaque neurone est négligeable et on peut considérer ce calcul comme instantané. Pour cette raison, les réseaux non bouclés sont souvent appelés « réseaux statiques », par opposition aux réseaux bouclés ou « dynamiques ». Ils sont utilisés en classification, reconnaissance des formes (caractères, parole, ...) et en prédiction[9].

1.3.2 Les réseaux de neurones bouclés

Contrairement aux réseaux de neurones non bouclés dont le graphe de connexions est acyclique, les réseaux de neurones bouclés peuvent avoir une topologie de connexions quelconque, comprenant notamment des boucles qui ramènent aux entrées la valeur d'une ou plusieurs sorties[10][8].

Pour qu'un tel système soit causal, il faut évidemment qu'à toute boucle soit associé un retard : un réseau de neurones bouclé est donc un système dynamique, régi par des équations différentielles

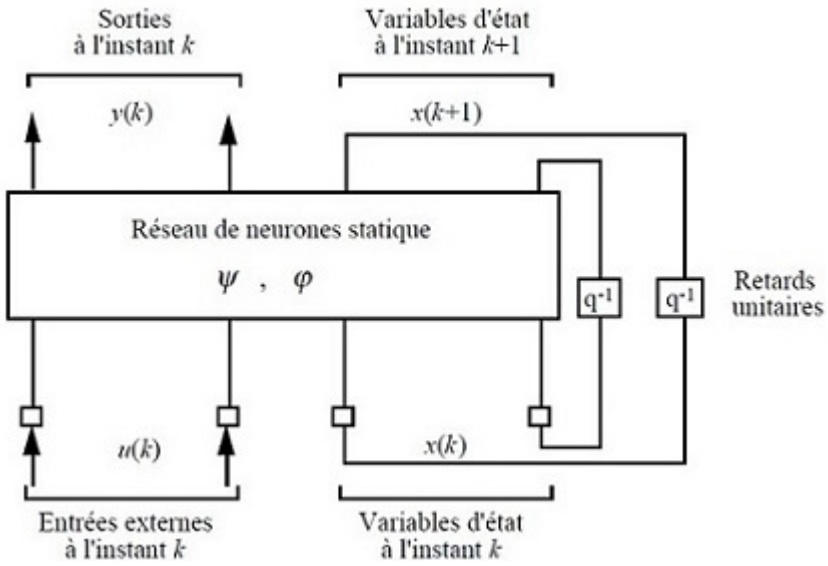

Figure 1.5 : *Forme canonique d'un réseau de neurones bouclé*

Les réseaux de neurones bouclés sont utilisés pour effectuer des tâches de modélisation de systèmes dynamiques, de commande de processus, ou de filtrage[10].

1.4 Apprentissage des réseaux de neurones

Le point crucial du développement d'un réseau de neurone est son apprentissage. Il s'agit d'une procédure adaptative par laquelle les connexions des neurones sont ajustées face à une source d'information. Dans la majorité des algorithmes actuels les variables modifiées pendant l'apprentissage sont les poids des connexions. L'apprentissage est la modification des poids du réseau dans l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Les poids sont initialisés avec des valeurs aléatoires, puis des données expérimentaux représentatifs du fonctionnement du procédé dans un domaine donné, sont appliquées au réseau de neurones. Il existe de nombreux types de règles d'apprentissages qui peuvent être regroupées en trois catégories. Mais l'objectif fondamental de l'apprentissage reste le même. [7][4][6]

1.4.1 Apprentissage supervisé

Un apprentissage est dit supervisé lorsque l'on force le réseau à converger vers un état final précis, en même temps qu'on lui présente un motif. Ce genre d'apprentissage est réalisé à l'aide d'une base d'apprentissage constituée de plusieurs échantillons(données) sous forme de couple (entrées,sorties).

La modification des poids s'effectue progressivement jusqu'à ce que l'erreur (ou

l'écart) entre les sorties du réseau (ou résultats calculés) et les résultats désirés soient minimisés.

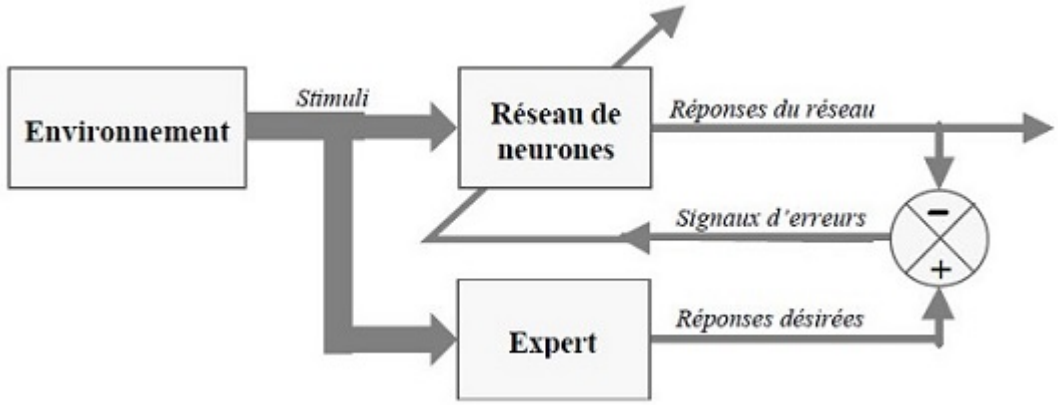

Figure 1.6 : *Illustration de l'apprentissage supervisé*

1.4.2 Apprentissage par renforcement

L'apprentissage par renforcement est une technique similaire à l'apprentissage supervisé à la différence qu'au lieu de fournir des résultats désirés au réseau, on lui accorde plutôt un grade (ou score) qui est une mesure du degré de performance du réseau après quelques itérations.

1.4.3 Apprentissage non supervisé

L'apprentissage non supervisé consiste à ajuster les poids à partir d'un seul ensemble d'apprentissage formé uniquement de données. Aucun résultat désiré n'est fourni au réseau. L'avantage de ce type d'apprentissage réside dans sa grande capacité d'adaptation reconnue comme une auto-organisation, « self-organizing».

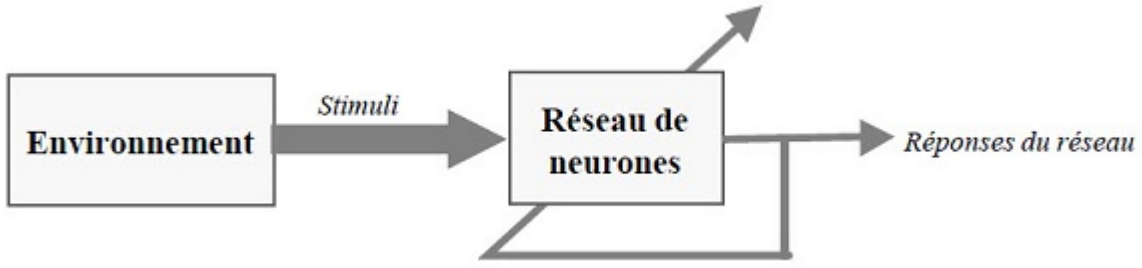

Figure 1.7 : *Illustration de l'apprentissage non supervisé*

1.5 réseaux de neurones et classification

Lorsque l'on cherche à résoudre un problème de classification, on a affaire à des données de différentes natures (par exemples des chiffres manuscrits, des signaux, des images, des phonèmes, des pièces mécaniques, etc...) susceptibles d'appartenir à des catégories, ou classes différentes. les réseaux de neurones offrent des performances équivalentes à celles des meilleurs classifieurs en termes de taux de reconnaissance, mais ils se distinguent de ceux-ci par une plus grande facilité d'implantation sous forme de circuits spécialisés très rapides[3][6][4].

1.5.1 Perceptron Multicouche (*Multi Layer Perceptron : MLP*)

1.5.1.1 Présentation

Un classifieur à K classes peut être réalisé avec un perceptron multicouches. Ce réseau se présente comme la succession de plusieurs couches comprenant une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque neurone est connecté à l'ensemble des neurones de la couche qui suit celle à laquelle il appartient. Dans un problème de classification, le nombre de neurones en sortie est fonction du codage adopté. Le plus souvent, chaque sortie est dédiée à une classe donnée (codage 1 parmi K).[3][6][4]

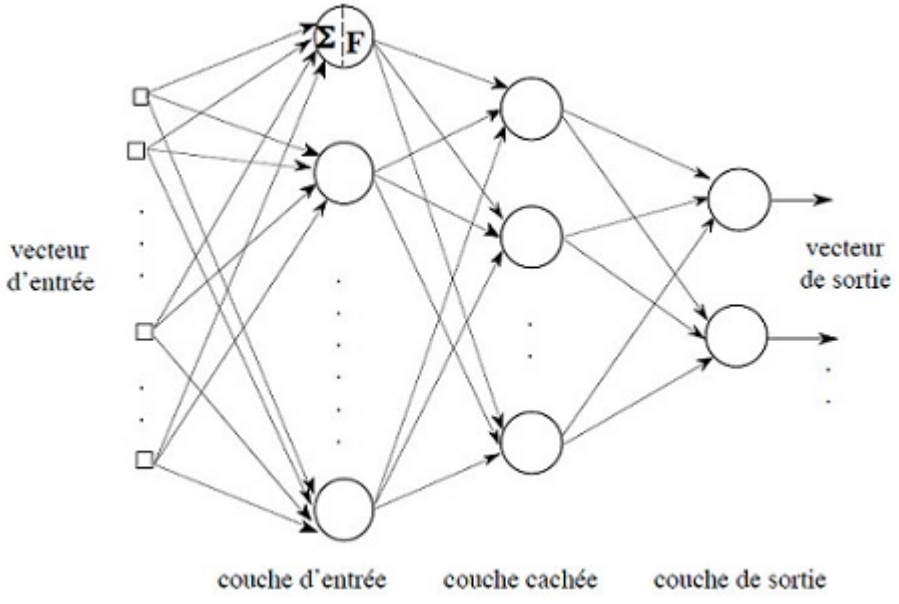

Figure 1.8 : *Structure d'un réseau MLP*

1.5.1.2 Apprentissage

L'entrainement d'un MLP se fait avec une méthode supervisé à l'aide de l'algorithme de rétro-propagation du gradient. L'algorithme de rétro-propagation du gradient altère les coéficients synaptiques du réseau dans le sens inverse du gradient du critère d'erreur J_N, en utilisant seulement les données d'entrées/sorties. En effet, l'erreur à la sortie du réseau et le résultat de fausse valeurs de plusieurs poids synaptiques. Ainsi, l'objectif principal d'un algorithme d'apprentissage est d'assigner le crédit pour chaque poids synaptique dans le réseau et de corriger sa valeur. L'algorithme de rétro-propagation effectue ceci en propageant les erreurs de la sortie vers l'entrée à travers le réseau.[3][6][11]

1.5.2 Réseaux de neurone probabiliste(*Probabilistic Neural Network : PNN*)

1.5.2.1 Présentation

Un réseau de neurones probabiliste (PNN) est basé sur le neurone à fonction radiale de base (appeler aussi noyau). ce neurone calcule la distance entre l'entrée et son centre qu'il fait passé ensuite dans une non linéarité (figure 1.9). Dans ce modèle l'opération distance remplace l'opération produit scalaire du neurones formel.

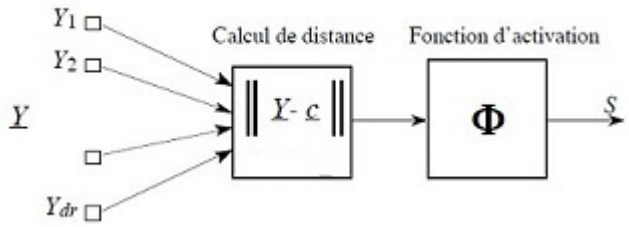

Figure 1.9 : *Neurone élémentaire à noyau*

la sortie S du neurone s'écrit sous la forme :

$$S = \phi(\| Y - c \|)$$

Les neurones à fonction radiale de base (RBF) utilise des fonctions d'activation définies de R vers R^+ qui sont symétrique radialement par rapport à un point (d'ou la dénomination de neurone à fonctions radiales de base) parmi lesquelles on peut citer :

– fonction gaussien

$$\phi(v) = \exp(-\frac{v^2}{2\beta^2})$$

– fonction thin plate

$$\phi(v) = v^2 \log(v)$$

– fonction multiquadratique

$$\phi(v) = \sqrt{(v^2 + \beta^2)}$$

Le fonction d'activation *gaussien* est le plus utilisé. La valeur que prend sa sortie d'autant plus importante que l'entrée est plus proche de son centre et elle tend vers zéro lorsque la distance entrée-centre devient importante. Le paramètre β permet de contrôler la vitesse de décroissance de la fonction ϕ.

L'architecture d'un réseau *PNN* s'organise en deux couches seulement : une couche cachée et une couche de sortie. La première couche, constituée de N_c noyaux élémentaires effectue une transformation non linéaire de l'espace d'entrée. la seconde couche de sortie calcul une combinaison linéaire des sorties des noyaux élémentaires.[3]

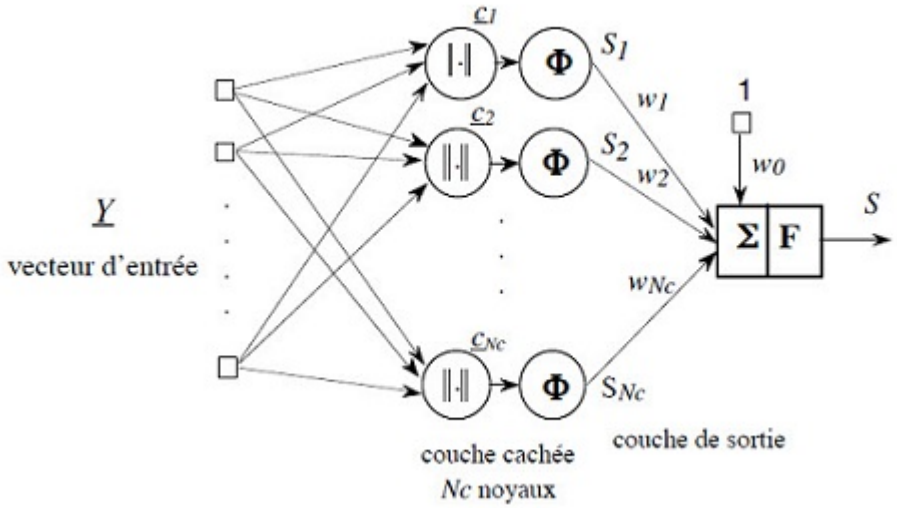

Figure 1.10 : *Architecture d'un réseau PNN*

1.5.2.2 Apprentissage

Les paramètres ajustable dans un réseaux *PNN* sont :
– la position des centres c_i.
– l'optimisation du nombre N_c de noyaux.
– valeur de l'écart-type β_i associés à chaque noyau (neurone)
– les poids de la couche de sortie w.

Différentes stratégies d'apprentissage sont alors possibles [3] :

Apprentissage global pour un nombre de noyaux fixé a priori, cette approche consiste à ajuster simultanément à l'aide d'un apprentissage supervisé la position des noyaux, l'écart-type relatif à chaque noyau et les poids en sortie. Ces

paramètres sont modifié itérativement a l'aide d'un algorithme de gradient pour minimiser une fonction de coût de type moindre carrés.

Apprentissage hybride ce dernier possède plusieurs variantes. Soit en commence par le positionnement des noyaux et l'optimisation de leur nombre et on détermine ensuite les poids et les matrices de normalisation par apprentissage à l'aide d'un algorithme. Soit l'apprentissage des deux couches s'éffectue séparément et dans un premier temps, on optimise les paramètres de la couche cachée (position des noyaux, nombre de noyaux, écart-types et matrice de normalisation) puis on procède au calcul des poids de la couche de sortie.

1.5.2.3 Positionnement des noyaux

Le positionnement des noyaux est un problème crucial dans les réseaux *RBF*. Le choix qui consiste à centrer un noyau sur chaque exemple de la base d'apprentissage est peu réaliste et conduit rapidement à une explosion de la taille du réseau si le nombre d'exemple d'apprentissage est important.

Une alternative à cet inconvenient est la sélection de paramètres pertinents. On peut soit se servire de l'ensemble des exemples de la base d'apprentissage pour construire un ensemble réduit de noyaux, soit sélectionner parmi ces exemples un sous-ensemble représentatif de l'espace des observations.

Pour chaque type d'approche, diverses méthodes sont possibles : l'algorithme *K-means* et l'algorithme de selection par *orthogonalisation (OFR)*.[3]

Algorithme K-means il s'agit d'une méthode de classification automatique non supervisée. Elle a pour but de regrouper les N observations de la base d'apprentissage en N_c groupes (ou clusters) de sorte que toute les observations soit plus proches (au sens d'une distance donnée) des exemples appartenant à son groupe qu'à ceux appartenant aux autres groupes. A chaque groupe de la partition obtenue correspondra un noyaux élémentaire du réseau RBF dont le centre sera le centre de gravité du groupe. Cette algorithme suppose néanmoins que l'on a fixé a priori le nombre de noyaux à atteindre.
Les étapes suivantes sont à envisager :
- choix au hasard de N_c exemples comme étant les centres des N_c groupes.
- affecter tous les exemples d'apprentissage au centre le plus proche en utilisant souvent la distance euclidienne.
- calculer les nouveaux centres de gravité des groupes ainsi obtenus.
- refaire les étapes 2 et 3 jusqu'à ce qu'il ait plus de changement des affectations des exemples.

Méthode d'orthogonalisation(*OFR*) Nous nous plaçons dans le cadre de la selection des N_c centres parmi les N exemples d'apprentissage. Comme pour la selection

de paramètres, la méthode d'orthogonalisation va permettre de classer chaque observation parmi la base complète en terme de contribution à la sortie souhaitée. Chaque observation est donc initialement un centre et le réseaux RBF est considéré comme un modèle de regression linéaire ou la sortie souhaitée pour chaque obsérvation de la base d'apprentissage est donné par la relation :

$$S(Y_k) = \sum_{i=1}^{N} W_i \phi(\parallel Y_i - Y_k \parallel) + \epsilon(k)$$

$k = 1, 2, ..., N$ avec :

$$Y_i = [Y_{i1}, Y_{i2}, ..., Y_{id}]$$

Dans la méthode OFR, le classement des noyaux s'effectue séquentiellement selon leur contribution à la prédiction de la sortie mesurée sur la variance.
La méthode d'orthogonalisation offre une alternative interessante pour construire un réseau RBF parcimonieux. Par rapport aux méthode de type clustering, elle permet à la fois de positionner les noyaux et d'en règler le nombre.

1.5.2.4 Choix de la largeur du noyau (écart-type β_i)

Dans le réseau RBF, l'ajustement de l'écart-type β_i associé à chaque noyau permet de contrôler le chevauchement des différentes fonctions élémentaires. S'il est choisi trop petit, la décroissance de la fonction d'activation autour du centre est rapide : des exemples très proches du centre peuvent conduire à des réponses nulles. A l'inverse, si la largeur du noyau est trop grande, des observations même situées très loin du centre contribuent significativement à la sortie. Dans les deux cas, cela se traduit par une mauvaise généralisation du réseau.

1.6 Outils pour les réseaux de neurones

1.6.1 Neural network toolbox de MATLAB

1.6.1.1 Présentation

Neural network toolbox fournit des outils pour concevoir, mettre en application, visualiser, et simuler les réseaux de neurones.

Ce *Toolbox* peut être utilisé de deux manière :
– utilisation de l'interface graphique *nnstart*.
– utilisation des fonctions prédéfinies avec les fichier M-files.

1.6.1.2 Exemple d'utilisation

Nous voulons classer plusieurs points de coordonnée (a,b) dans le plan (X,Y) en 4 classes définies par les centres (-2,-2),(-2,2),(2,-2) et (2,2). Pour créer un tel réseau on utilise la fonction NEWPNN avec la syntaxe suivante :

$$net = newpann(P, T)$$

avec :

- p : données d'entrée, chaque point est organisé en vecteur cologne.
- T : matrice des classes correspondantes aux points de P.

Pour tester notre réseau, on génère 200 points de manière aléatoire.
Les points générés ont été classés en 4 classes comme le montre la figure 1.11.En remarque que seul 3 points sont mal classés.

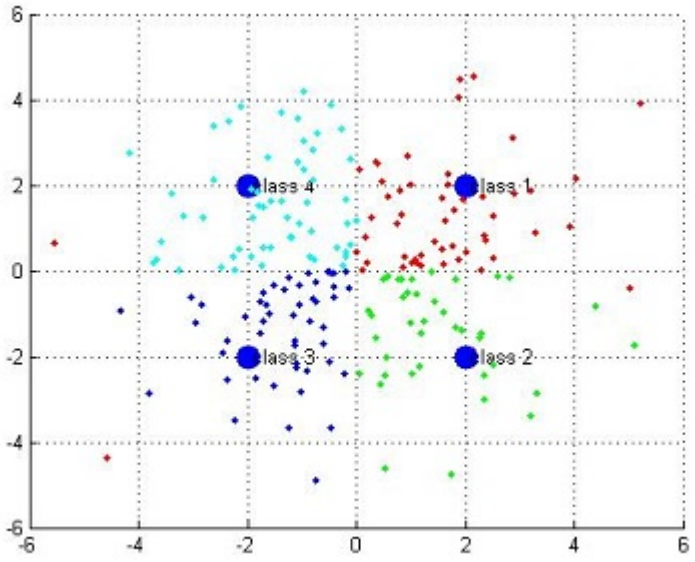

Figure 1.11 : *Résultat de classification*

1.6.2 Bibliothèque pour C++

FANN (pour Fast Artificial Neural Network) est une librairie open-source de réseaux de neurones qui permet à la fois l'entraînement du réseau neuronal et ensuite l'utilisation de celui-ci. d'après les exemples vue dans le guide de cette bibliothèque, elle est relativement simple d'utilisation. Des fonctions de création , de configuration, d'entraînement et de test sont prédéfinit. le FANNTOOL est un logiciel multi plateforme développé en C qui fournit un ensemble d'outils permettant l'utilisation de la librairie FANN. l'interface est relativement simple d'utilisation.Elle propose diffèrent

algorithme d'entraînement, plusieurs fonction d'activation et permet de personnaliser les dimensions du réseau (nombre de couches, nombre de neurone dans chaque couche).

1.7 Conclusion

Dans ce chapitre, nous avons rappelé d'abord des notions générales sur les réseaux de neurones : neurones formel, fonction d'activation, différentes topologies de réseau et les types d'apprentissage.

Par la suite, nous nous somme focalisé sur les réseaux de neurones pour la classification. nous avons présenté deux architectures, à savoir le perceptron multi-couche (MLP) et le reseau probabiliste (PNN) toute en détaillant leurs architectures et leur stratégies d'apprentissage.

Enfin, nous avons terminé par une breve présentation de quelque outils permettant de créer et de simuler les réseaux de neurones.

CHAPITRE 2

Classification des signaux SEMG

2.1 Introduction

Les signaux elecromyographiques de surface (Surface ElectroMyoGraphy signal SEMG), sont des signaux électriques pouvant être enregistrées à la surface des muscles pendant leurs contractions. L'interprétation de ces signaux peut conduire à de nombreuses applications, parmi lesquelles figure le contrôle de prothèses. [9].

A cause de leurs caractéristiques, l'interprétation de ces signaux nécessite l'application de plusieurs traitements spécifiques. L'objectif de ce chapitre est de présenter les principaux méthodes de traitement, de caractérisation et de classification de ces signaux *SEMG*.

2.2 Muscle squeletique

Le corps humain comprend plus de 650 muscles fixés au squelette soit directement, soit par des tendons. Les muscles travaillent conjointement afin de fournir une force pour produire les mouvements du corps : mouvement d'une articulation, stabiliser une position ou prévenir tout mouvement dans la direction opposée à celle souhaitée. Chaque extrémité du muscle est attachée à l'os par des points appelés origine ou insertion (figure 2.1). L'origine correspond au point d'attache du muscle à l'os qui reste fixe et l'insertion est le point d'attache du muscle à l'os qu'il met en mouvement : le raccourcissement du muscle pendant sa contraction rapproche les deux os. En règle générale, seul l'os comprenant l'insertion est mis en action alors que l'os auquel se rattache l'origine du muscle reste fermement en place.[12]

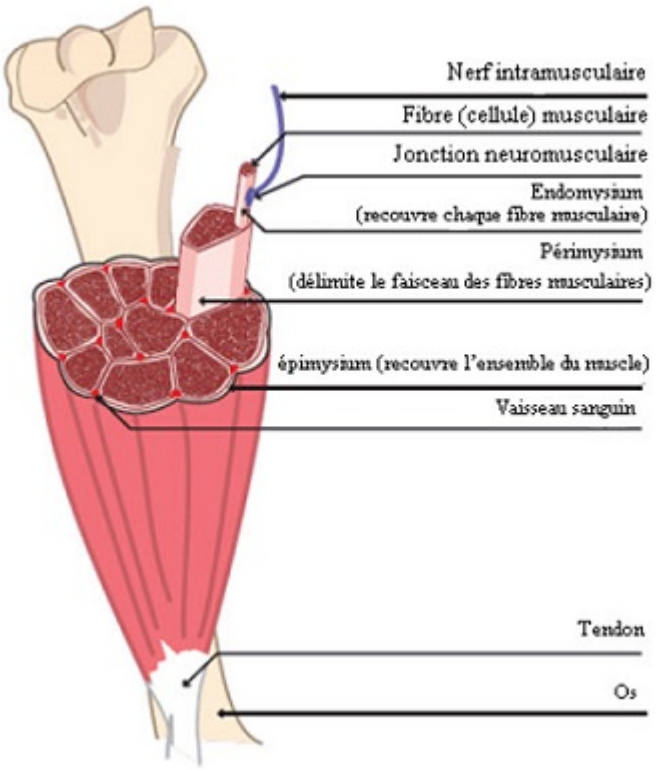

Figure 2.1 : *Anatomie du muscle strié squelettique*

2.2.1 Anatomie

Le muscle squelettique est constitué de faisceaux musculaires formés eux-mêmes d'un ensemble de cellules de forme cylindrique appelées *fibres musculaires*, liées entre elles par du tissu conjonctif. Les fibres musculaires sont innervées par les axones des nerfs moteurs émanant de la moelle épinière. Chaque nerf moteur ou motoneurone innerve ainsi plusieurs fibres musculaires. Par contre, une fibre musculaire n'est innervée que par un seul motoneurone. L'ensemble formé par un motoneurone et les fibres qu'il innerve est appelé unité motrice (UM). C'est la plus petite unité fonctionnelle musculaire, car la plus petite contraction musculaire résulterait en fait de l'activation d'une seule unité motrice. Les connexions entre les terminaisons axonales des motoneurones et les fibres musculaires, appelées les jonctions neuromusculaires (JNM) ou les plaques motrices, se trouvent généralement au milieu du muscle.

2.2.2 Physiologie

L'activité normale d'un muscle squelettique est tributaire de l'innervation de ses unités motrices. Chaque fibre musculaire est en contact avec une terminaison nerveuse qui régit son activité. Ces fibres se contractent par la stimulation exercée par le motoneurone sous la forme d'impulsions nerveuses synchronisées générées par

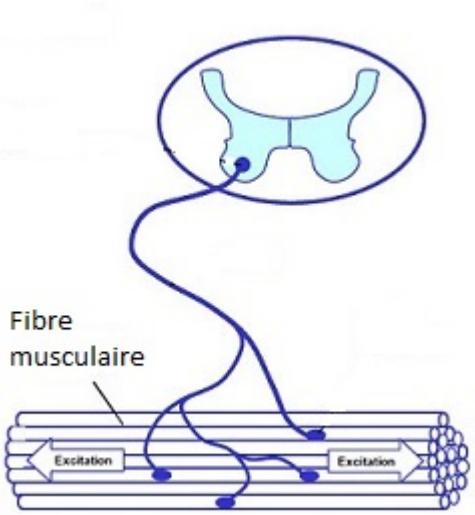

Figure 2.2 : *Unité motrice*

le système nerveux constitué du cerveau et la moelle épinière. Quand un motoneurone est activé, toutes les fibres qu'il innerve répondent aux impulsions du neurone en générant leurs propres signaux électriques qui conduisent à leurs contraction[13] [14]. Pour produire ou maintenir une force désirée pour un muscle, le système nerveux contrôle principalement deux paramètres de l'activation des unités motrices : le recrutement de nouvelles unités motrices (recrutement spatial) et la modulation de la fréquence de décharge des unités motrices déjà activées (recrutement temporel). Quand un muscle est activé, le recrutement d'unité motrice respecte « le principe de la taille »[15]. Les unités motrices possèdent le moins de fibres sont recrutées au début de la contraction musculaire, et plus tard, les unités motrices plus grandes sont activées.

2.3 Electromyographie de surface (SEMG)

2.3.1 Définition

La somme des phénomènes électriques, correspondant à la contraction des différentes fibres musculaires impliquées dans l'activité du muscle considéré, génère un champ électrique suffisamment important pour pouvoir être recueilli, soit localement (dans le muscle à l'aide d'un électrode à aiguille), soit à la surface de la peau, On parle alors d'électromyographie de surface. L'électromyographie de surface est une méthode invasive fréquemment employée dans les domaines de l'étude du mouvement, du diagnostic neuromusculaire et de contrôle de prothèses.

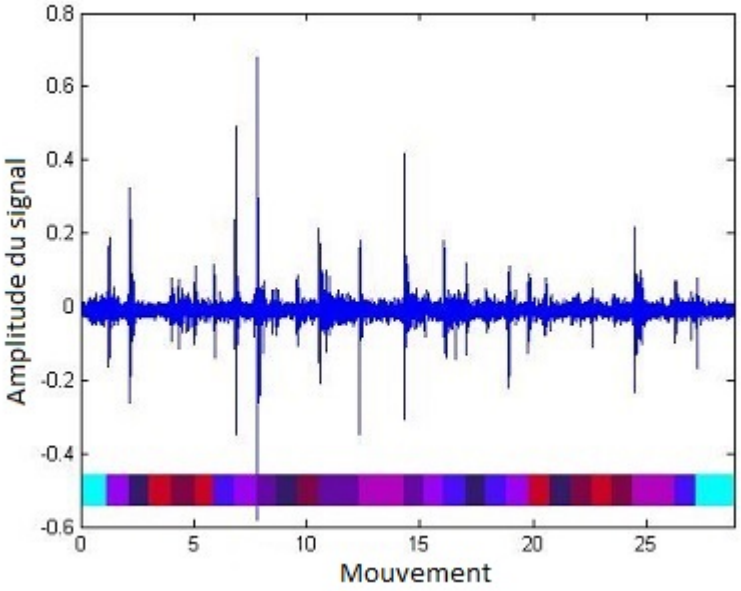

Figure 2.3 : *Exemple de signal SEMG capturé sur l'avant bras pour plusieurs mouvements.*

2.3.2 Génération du signal EMG

Durant la contraction, l'impulsion du motoneurone produit sur chaque fibre à travers la jonction neuromusculaire une zone de dépolarisation se propageant à une vitesse de conduction (VC) vers les extrémités de la fibre. Cette dépolarisation produit un potentiel d'action de fibre (PAF) détectable à la surface de la peau. L'allure de la réponse électrique à la décharge d'une UM est appelée potentiel d'action d'unité motrice (PAUM). Il contient la somme pondérée des PAF de chaque fibre musculaire constituant l'unité motrice[16][12]. Le signal SEMG correspond donc à une sommation des potentiels d'action des unités motrices actives. L'amplitude du signal SEMG varie typiquement entre $+-50$mV et sa bande de fréquence s'étend de 5Hz jusqu'à 500Hz avec une concentration de la puissance dans la bande 20Hz-150Hz[ver].

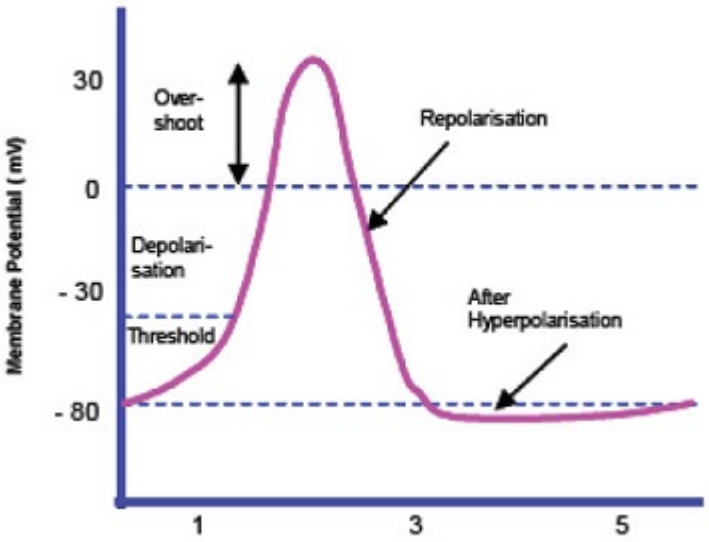

Figure 2.4 : *Potentiel d'action de la fibre*

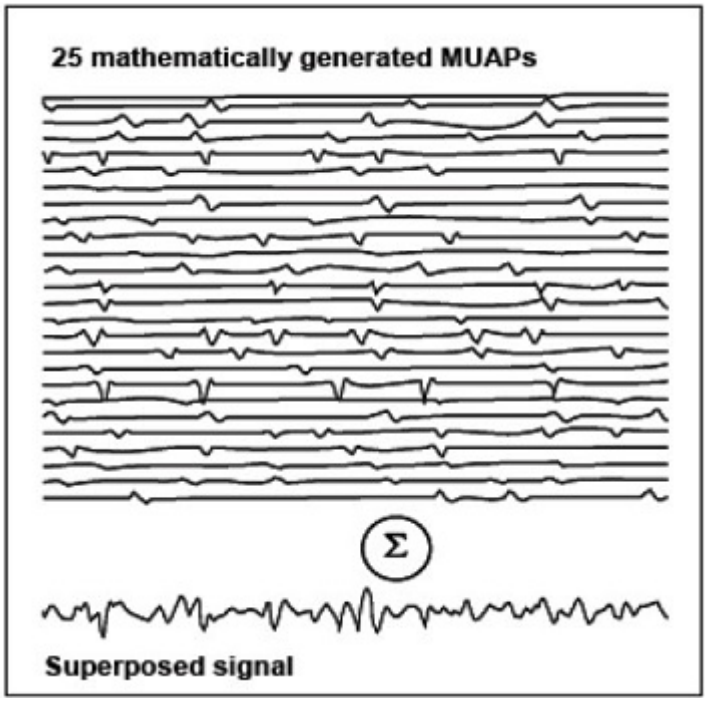

Figure 2.5 : *Superposition des PAUM*

2.4 Aquisition du signal SEMG

Le signal SEMG est recueillie à l'aide des électrodes de surface. Elles sont fixées sur la peau au niveau des muscles cible pour mesurer l'activité musculaire globale sous-jacente correspondant à plusieurs unités motrices. Elles fournissent le signal

émis par un ensemble de fibres musculaires et non plus d'une seule fibre comme les électrodes à aiguille.
Le signal est acquis à une fréquence qui doit respecter la théorie d'échantillonnage de *Nyquist-Shannon*. Cette théorie veut que la fréquence d'acquisition soit au moins deux fois plus grande que la fréquence maximale du signal d'entrée lorsque celui-ci n'est pas affecté par bruit additif. Puisque la bande passante du signal EMG se situe entre 10 et 500 Hz, la fréquence d'acquisition doit être d'au moins 1000 Hz [17][13].

2.4.1 facteur influençant le signal EMG

Le signal EMG est influencé par différents facteurs tels que les caractéristiques du tissu (genre de tissu, épaisseur du tissu, changement physiologiques et la température), la diaphonie (cross-talk), le changement de géométrie du muscle lors de sa contraction, le bruit extérieur (noise), l'emplacement du système de détection (distance inter-électrodes, inclinaison des fibres musculaires en comparaison avec les électrodes) et sa qualité (électrodes et amplificateurs). La diaphonie représente l'influence des muscles environnant au volume de détection et ne dépasse généralement pas 10% à 15% du signal global. Ainsi, le signal provenant d'une électrode placée sur un muscle de surface peut être influencé par les muscles plus profonds faisant partie du volume de détection de l'électrode. Le signal cardiaque peut aussi causer de la diaphonie, surtout pour les signaux détectés au niveau des muscles des membres supérieurs. De plus, le signal EMG varie selon les sujets, les différents emplacements des électrodes et les changements temporels dus à la fatigue musculaire et la sueur [17][14][18][19].

2.5 Caractérisation des signaux SEMG

2.5.1 Domaine temporel

2.5.1.1 Moyenne quadratique (Root Mean Square, RMS)

La plupart du temps, le signal EMG est quantifié dans le domaine temporel au moyen de sa moyenne quadratique (Root Mean Square, RMS), qui représente l'amplitude du signal EMG sur un intervalle de temps donné :

$$RMS = \sqrt{\frac{1}{T}\int_{t-\frac{1}{T}}^{t+\frac{1}{T}} (X(t))^2 dt}$$

ou *X(t)* est le signal a analysé, et T l'intervalle de temps.

2.5.1.2 Integral of absolute value (*IAV*)

L'IAV d'un signal EMG est calculer selon l'equation suivante :

$$IAV = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Avec :

- X_i : i^{me} échantillon.
- N : nombre d'échantillon.

2.5.1.3 Zéro crossing (*ZC*)

ZC est le nombre de fois que le signal passe par l'amplitude zéro. Il est calculer selon l'equation suivante :

$$ZC = \sum_{i=1}^{N} sgn(-X_i X_{i+1})$$

Avec :

$$sgn(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } autre \end{cases}$$

2.5.1.4 Wavelength, WL

Cette fonction évalue la longueur d'onde du signal. Elle est définie selon l'equation suivante :

$$W(n) = \sum_{i=n-N+1}^{n} |\Delta(X_i)|$$

avec : $\Delta(X_i) = w_i - X_{i-1}$

2.5.1.5 Mean of absolue value, MAV

MAV est calculer sur tout la duré du signal selon l'equation :

$$MAV = \frac{1}{N}\sum_{1}^{N} X_i$$

Avec :

- X_i : i^{me} échantillon.
- N : nombre d'échantillon.

2.5.1.6 Coefficient Autoregressif AR

le modèle autorégressif est une représentation pour les processus aléatoire, qui décrit les données temporelles observées par une relation linéaire avec leurs valeurs antérieures. le modèle autorégressif d'ordre p, noté AR(p) est donné par :

$$X_t = C_t + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + ... + \varphi_p X_{t-p} + \epsilon_t$$

avec :
- X_t : processus aléatoire
- φ_i : coefficients du modèle
- C_t : constante
- ϵ_t :bruit blanc

2.5.2 Domaine fréquentiel

2.5.3 Densité spéctrale

Il existe différentes méthodes d'estimation de la densité spectrale de puissance. Le problème des estimateurs de la densité spectrale réside dans leur variance qui est généralement grande et qui ne diminue pas en augmentant le nombre d'échantillons du signal. Les deux méthodes les plus couramment utilisées sont les suivants :

Périodogramme Cette méthode estime la densité spectrale de puissance comme étant le carré du module de la transformée de Fourier du signal. Cette méthode est en fait équivalente à prendre la transformée de Fourier de la fonction d'auto-correlation.

Périodogramme moyenné (Méthode de Welch) Une manière de réduire la variance de l'estimateur est de subdiviser la fenêtre d'observation du signal x(t) en un certain nombre d'intervalles. La moyenne des densités spectrales calculées sur chacun des intervalles donne le périodogramme moyenné.

Puisque les signaux EMG possèdent les caractéristiques pseudo-stationnaires, il vaut mieux estimer le spectre de puissance (DSP) en utilisant la méthode de Welch , qui permet de trouver un compromis entre le biais et la variance

2.5.3.1 Fréquence pic

La Fréquence pic est la fréquence pour laquelle la fonction de densité spectrale atteint un maximum.

2.5.3.2 Fréquence moyenne

Elle représente la moyenne statistique du signal.

$$MPF = \frac{M_1}{M_0}$$

Avec :

$$M_r = 2\int_0^{\inf} f^r S_x(f)df$$

2.5.3.3 Fréquence médiane

La médiane partage la distribution de la densité spectrale en deux parties : 50% des données sont plus petites que la médiane, 50% sont plus grandes. La médiane est calculée par :

$$\int_0^{fmed} S_x(f)df = \int_{fmed}^{fmax} S_x(f)df$$

2.5.3.4 Entropie spectrale

L'entropie mesure la quantité d'information moyenne contenue dans un signal, elle est significative de la variance spectrale :

$$H = -\int_0^{fmax} S_x(f)\ln[S_x(f)]df$$

Avec : $S_x(f)$: représente la densité spectrale de puissance.

2.6 Classification des signaux SEMG

Dans le cas des signaux SEMG, l'intérêt de la classification revient à savoir à quelle classe physiologique appartient un segment donné à partir des traitements effectués sur celui-ci.[20]
. Cela ce fait en 3 étapes :

1. Extraction des paramètres (caractérisation).
2. Réduction de paramètres.
3. Classification.

2.6.1 Réduction de paramètres

Avant d'effectuer la classification proprement dite des signaux SEMG, il peut être nécessaire de faire subir un traitement supplémentaire aux caractéristiques extraites. Cette étape consiste à réduire la dimension de l'espace des caractéristiques de manière à ne retenir que les informations qui sont importantes pour la discrimination entre les classes et à éliminer celles qui sont inutiles. Son principal intérêt est qu'un classifieur avec moins d'entrées aura moins de paramètres adaptatifs à déterminer, ce qui conduit à un classifieur possédant des meilleures propriétés de généralisation. Il existe deux stratégies pour réduire la dimension : la sélection de caractéristiques et la projection de caractéristiques.

2.6.1.1 La sélection de caractéristique

Les méthodes de sélection de caractéristiques tentent de déterminer le meilleur sous-ensemble dans l'ensemble original des caractéristiques. Elles reposent sur une évaluation automatique des caractéristiques extraites. L'auteur dans [1] propose une méthode supervisé *LDA* (Linear Discriminant Analysis) qui utilise les paramètres extraits et leurs classes respectives pour maximiser la distance entre les classes, néanmoins cette méthode soufre de problème de singularité.
la méthode *ULDA*((Uncorrelated Linear Discriminant Analysis) est une variante de LDA avec plus de contrainte pour la réduction de paramètre, ce qui minimise la redondance dans les paramètres. le problème de singularité a été résolue en utilisant la décomposition en valeur singulière (*SVD*).

2.6.1.2 projection de caractéristique

Les méthodes de projection tentent de déterminer la meilleure combinaison des caractéristiques originales. Cette méthode consiste à projeter orthogonalement les vecteurs de caractéristiques dans un sous-espace qui restitue le plus fidèlement les distances entre les vecteurs. Les axes de ce sous-espace aboutissent alors à de nouveaux vecteurs non corrélés, appelés composantes principales (Principal Component Analysis, PCA), qui sont combinaisons linéaires des vecteurs de caractéristiques originaux et qui pourront être utilisés dans l'étape suivante de classification. PCA est une méthode non supervisée puisque la projection des vecteurs de caractéristiques ne nécessite pas d'attribuer une classe d'appartenance aux données.[9][1]

2.6.2 Méthode de classification

A partir d'un ensemble d'individus déjà classés, le système peut apprendre à classer. Après apprentissage, le système est capable de classer de nouveaux individus. Il existe de nombreuses techniques de classification qui peuvent être employées pour développer une règle de la décision. Plusieurs de ces techniques sont décrites dans les paragraphes suivantes.

2.6.2.1 Méthode non supervisée

Méthode de K-moyenne L'algorithme des K-moyennes est un algorithme classique de quantification vectorielle. Son principe est le suivant : on dispose de points de l'espace des observations que l'on souhaite rassembler en classes, sans que l'on dispose de connaissance a priori de propriété particulière sur ces classes ; seul leur nombre p est fixé a priori.[10]

Algorithme :

1. Définir K centroides (vecteurs) initialisés au hasard.
2. Attribuer chaque point au centroide le plus proche.
3. Calculer les coordonnées de chaque centroide comme la moyenne des points qui lui sont attribués.
4. Tant que les centroides évoluent, aller en 2.

Méthode de Linde-Buzo-Gray (LBG) L'algorithme de LBG ne fait pas seulement évoluer les centroides mais aussi leur nombre par duplication. L'intérêt de le faire est d'accroître la precision et aussi d'équilibrer les tailles des classes.[20]

Algorithme LBG :

Entree : K centroides $D = \{y_1, c, y_K\}$, centres de gravite des K ensembles de la partition de $X = \{x_j, j = 1, c, n\}$. Un vecteur z.

Tant que critère d'arret non-satisfait répéter.

Doubler le nombre de centroides en remplaçant chaque centroide y par deux autres y+z et y-z.

Affecter chaque exemple x_j a la classe du nouveau centroide dont il est le plus proche.

Adapter les centroides en les remplaçant par les centres de gravite de leurs classes.

Sortie : un regroupement en Kr classes et les centroides de chaque classe (leurs centres de gravite). L'avantage de cette méthode est que le nombre de classe n'est pas fixe a priori.

Réseaux de neurones compétitifs Le réseau de neurones compétitif (Competitive Neural Network Method, CNNM) permet aux neurones de sortie de se concurrencer afin qu'un seul neurone soit activé à un instant donné. Pour le neurone

gagnant, le niveau de son activité interne pour un signal d'entrée doit être le plus grand par rapport aux autres neurones du réseau. Le signal de sortie du neurone gagnant sera égal à un alors que les signaux de sortie des neurones perdants seront égaux à zéro. Par conséquent, chaque neurone d'un réseau apprend à se spécialiser sur un ensemble de signaux d'entrée et devient un détecteur spécialisé. Dans la forme la plus simple, le réseau compétitif possède un réseau de neurones à une seule couche de neurones de sortie, chacun étant complètement connecté aux signaux d'entrée.

méthode de classification basée sur le test de Fisher Après extraction des paramètres, chaque contraction est caractérisée par le vecteur des paramètres. La méthode de classification statistique non supervisée (USCM, unsupervised statistical classification method) consiste à trouver les vecteurs les plus proches pour les regrouper ensemble dans une même classe. Pour cela, les vecteurs sont comparés en utilisant le Test de Fisher mélangé avec la méthode de K-moyenne.[21]
L'algorithme se résume de la façon suivante :

1. Initialiser le premier vecteur, comme centre de la première classe.
2. Pour chaque nouveau vecteur, calculer la vraisemblance entre ce vecteur et le centre des classes en utilisant le test de Fisher
3. Si les vecteurs (nouveau vecteur et centre de classe) sont identiques avec une probabilité de confiance donnée, adapter les classes. Sinon créer une nouvelle classe.
4. Retour à l'étape 2.

Un seuil est utilisé pour comparer les vecteurs au centre (intervalle de confiance de Fisher). Une grande valeur de seuil aboutit à un nombre minimal de classes. Si la valeur du seuil est petite, alors on obtient un grand nombre des classes.

2.6.2.2 Méthode supervisée

Règle du plus proche voisin La méthode du plus proche voisin se caractérise par sa capacité à traiter localement les informations. Elle consiste à examiner l'élément, dont la classe est connue, le plus proche de l'élément dont on veut déterminer la classe.
Soit $x^n = \{x_1, ..., x_j, ..., x_n\}$ un ensemble d'apprentissage formé de n vecteurs indépendants, où chaque événement est représenté par un vecteur paramétrique, x_j. La classe de chaque élément de l'ensemble d'apprentissage est connue, elle sera désignée pour l'élément x_j par $w(x_j)$. L'événement à classer, représenté par un vecteur paramétrique x, est affecté à la classe correspondant au voisin le plus proche parmi l'ensemble d'apprentissage. La règle dite de 1PPV est simplement la suivante :

$\hat{w}(x) = w(x_{ppv})$ si $d(x, x_{ppv}) = \min_{j=1...n} d(x, x_j =$
où x_{PPV} est l'échantillon le plus proche de x, et $\hat{x}(x)$ est la classe d'affectation estimée de x. La règle du 1PPV classifie x selon sa classe d'affectation estimée $\hat{x}(x)$. La notion du "plus proche", en termes des distances les plus petites, laisse un certain choix pour la détermination de la distance. Dans le cas des K plus proches voisins, pour chaque vecteur x à classer, on recherche les K plus proches voisins dans l'échantillon d'apprentissage, et x est affecté au groupe majoritaire.[20]

Réseaux de neurones L'autre alternative est l'utilisation de réseaux de neurones, vue les différentes architectures existantes et leurs capacité d'adaptation. parmi ces diverses architectures on peut citer :
- perceptron multicouche.
- réseau de neurones probabiliste.

2.7 Conclusion

Dans ce chapitre nous avons présenté les muscles squelettique en donnant leurs principales propriétés et une description de leurs anatomies.
ensuite, nous avons présenté les signaux *SEMG* : définition, acquisition, extraction de caractéristiques, elimination de caractéristiques ne donnant pas d'information pour la classification.
enfin, nous avons rappelé le principe des différentes méthodes de classification : les méthodes de classification supervisées (Règle du plus proche voisin, Réseaux de neurones), et les méthodes de classification non-supervisées(Méthode de K-moyenne, Méthode de Linde-Buzo-Gray, Réseaux de neurones compétitifs, Méthode de classification basée sur le test de Fisher).

CHAPITRE 3

Classification des signaux SEMG par réseaux de neurones

3.1 Introduction

Comme déjà évoqué, le signal SEMG caractérisant l'activité neuromusculaire d'un muscle donné est largement exploité pour diverses applications médicales comme le diagnostic des anomalies dans les fonctions musculaires et le contrôle des prothèses. Ces applications sont basées essentiellement sur la classification dont la méthode définie entièrement les performances.

Pour les mains prothétiques myoélectriques, l'utilisation de plusieurs sites pour l'acquisition des signaux SEMG permettent de classifier et d'identifier plusieurs mouvements d'une main et ainsi mettre en oeuvre une prothèse multifonctionnelle.

Le processus de classification des mouvements, peut être décomposé en 3 phases principales : l'extraction de caractéristiques, la réduction de paramètres et la détermination de la classe du mouvement. L'extraction des caractéristiques consiste en la transformation du signal d'entrée en un ensemble de caractéristiques représentatives. Dans ce chapitre, nous préseterons les différents résultats des études menées en utlisant MATLAB pour choisir les différents paramètres du processus de classification.

3.2 Recueil des données

Les signaux SEMG utilisées ont été recueillies auprès de 30 sujets à partir de 7 sites sur l'avant-bras et un autre sur le biceps (figure 3.1). Ces signaux ont été amplifiés avec un gain de 1000 et échantillonnés avec une fréquence de 3 KHZ. Les données obtenues on été sous-échantillonnées à une fréquence de 1 KHZ avant la classification [1].

Les signaux ont été recueillies sur des sujets effectuant sept mouvements diffé-

rents : *Main ouverte*, *Main fermé*, *Supination*, *pronation*, *Flexion du poignet*, *extension du poignet* et *Repos*.

Dans chaque essai, le sujet répète chaque mouvement d'une durée de 3s quatre fois. L'ordre de ces mouvement est choisie d'une manière aléatoire.

Un total de six essais a été réalisé dans chaque séance, avec quatre séances réalisées sur quatre jours différents.

Dans notre travail, seules les données d'une séance on été utilisée. Les données des deux premiers essais sont utilisées comme données d'entraînement et les données des quatre essais restant sont utilisées comme données de test.

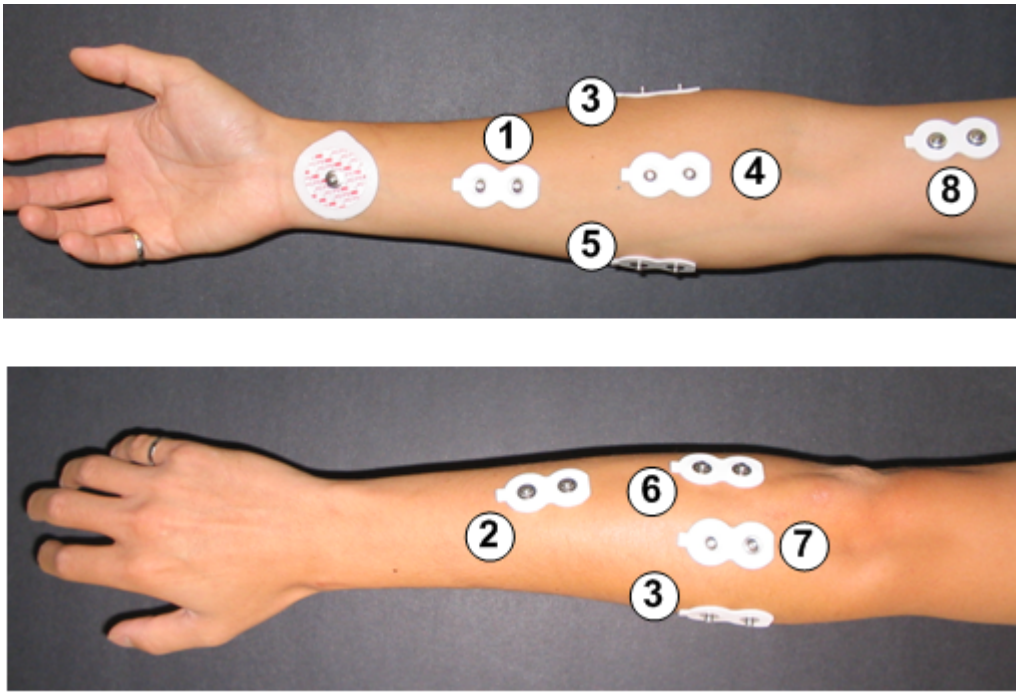

Figure 3.1 : *Emplacement des electrodes.*

3.3 Classification des signaux SEMG

Les performances du classifieur neuronal dépendent des paramètres suivants :
- type du réseau choisi et ces caractéristique.
- type de paramètres temporels et fréquentiels choisi pour la caractérisation du signal SEMG.
- Dimension des caractéristiques extraites (réduction).

Donc, il est impératif d'estimer l'influence de ces paramètres sur la classification avant de fixer leurs choix. Au terme des études que nous entameront dans cette partie, nous seront en mesure de prendre une décision sur le type de paramètres à utiliser lors de la caractérisation, la dimension de l'espace des paramètres et la structure du

réseau de neurones à implémenter pour la classification des signaux EMG.

Deux architectures de réseaux de neurones seront utilisées dans nos simulations. Le premier réseau est un MLP composé d'une couche cachée de 30 neurones et d'une couche de sortie de 7 neurones. Le deuxième réseau utilisé est le PNN qui est, par définition, composé de deux couches : une couche cachée de 7 noeuds (neurones) correspondant aux 7 centres de classes de mouvements et une couche de sortie de 7 neurones.
Rappelons que, dans un problème de classification en utilisant les réseaux de neurones, le nombre de neurone de la couche de sortie est imposé par le nombre de classes à différencier (codage 1 parmi K). ces réseaux sont crées, entrainés et testés en utilisant Neural Network Toolbox avec les valeurs prédéfinies.

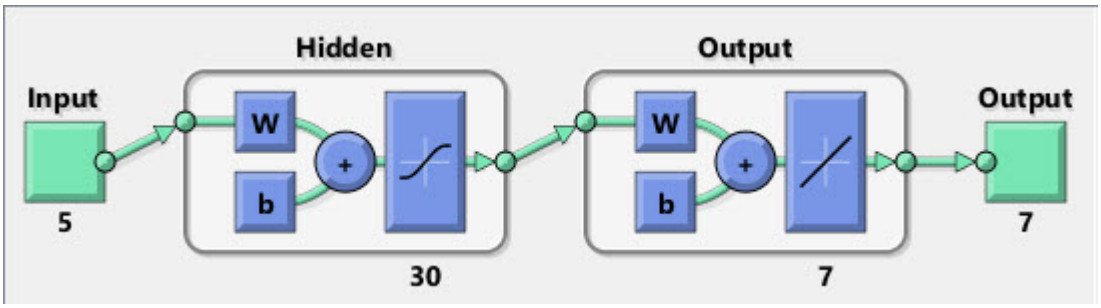

Figure 3.2 : *Réseau MLP avec une couche cachée (5 entrées)*

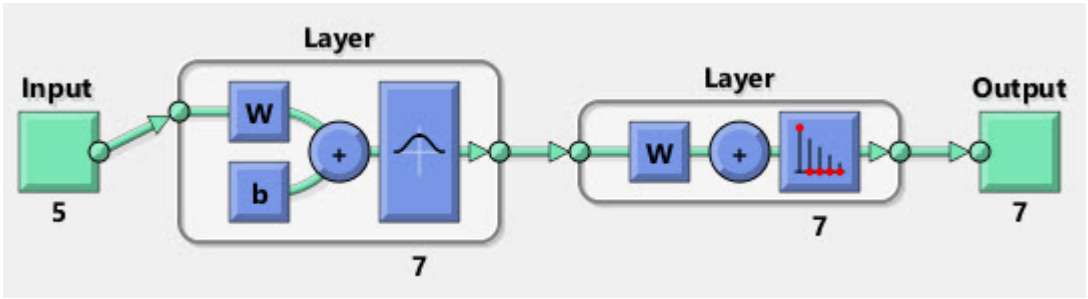

Figure 3.3 : *Réseau PNN (5 entrées)*

Les paramètres temporels et fréquentiels ont été calculés sur des segments du signal SEMG en considérant une fenêtre glissante comme montré par la figure 3.4. la largeur de la fenêtre a été fixée à 256ms (il est convenu que pour le contrôle avec SEMG, une durée de segment inférieure à 300ms est acceptable [1].

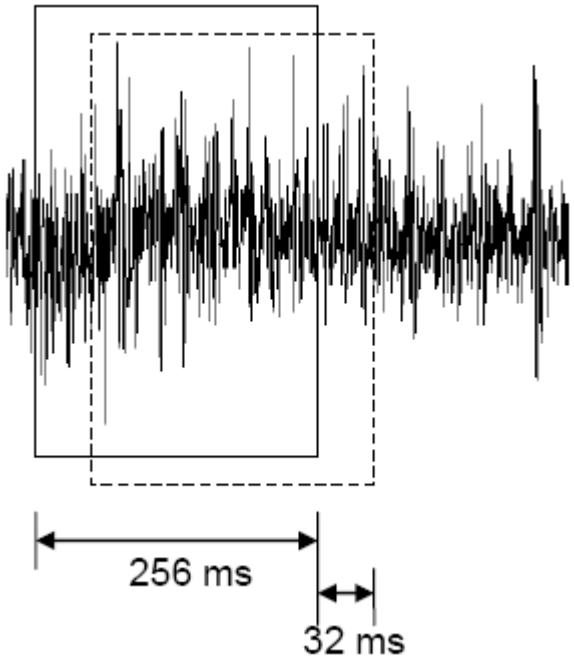

Figure 3.4 : *fenêtre de segmentation*

pour la réduction des paramètres, seule la méthode ULDA a été utilisée. Les fonctions du Myoelectric Control Toolbox ont été utilisées pour la réduction des paramètres ainsi que pour le calcul des caractéristiques [22].

3.3.1 Etude des caractéristiques

3.3.1.1 Etude 1 : influence du type de paramètre sur la classification

Dans cette partie nous nous intéressons à l'influence des paramètres suivants : RMS, MAV, IAV, WL, AR, Fmed, Fmoy, pour seul individu. Cette simulation couvre les deux cas de classification sans reduction et avec reduction a 5 paramètres. Les résultats de simulation pour le MLP et le PNN sont donnés par les tableaux 3.1 et 3.2 respectivement.

	RMS	IAV	MAV	WL	AR	Fmed	Fmoy
Sans reduction	94.56*	94.28*	92.99*	96.36*	93.49**	88.26*	85.75*
Avec reduction	94.27	94.56	94.00	95.85	95.29	85.25	81.08
* : 8 paramètres, ** : 32 paramètres							

Tableau 3.1 : *Taux de classification du MLP en fonction des différents paramètres*

La figure 3.5 présente en détails les résultats de classification du MLP en utilisant

	RMS	IAV	MAV	WL	AR	Fmed	Fmoy
Sans reduction	89.15	29.57	87.59	36.66	92.85	79.70	80.03
Avec reduction	92.70	91.95	91.95	91.44	95.02	83.35	83.71

Tableau 3.2 : *Taux de classification du PNN en fonction des différents paramètres*

le paramètre WL donnant le meilleur taux de classification. Nous remarquons que la majorité des misclassifications se produisent

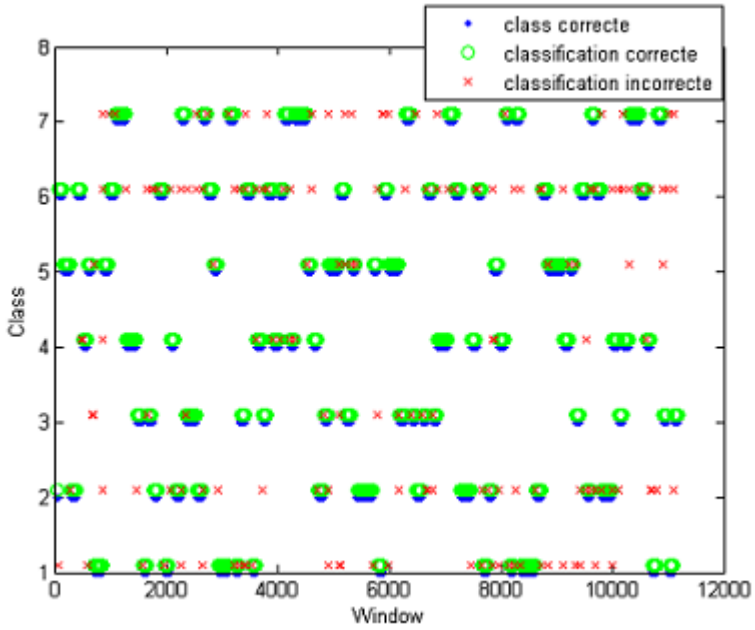

Figure 3.5 : *résultats de classification du MLP en utilisant WL avec reduction.*

La figure 3.6 donne une représentation graphique de la matrice de confusion. cette matrice carrée est générée en comparant les classes prévues avec les résultats de classification du réseau de neurones, elle fournie trois informations pour chacune des classes :

- le nombre des mouvements bien classées (diagonale).
- le nombre des misclassification intraclasses (colonnes) : classe i prévue, classe j obtenue.
- le nombre des misclassification interclasses (lignes) : classe i obtenue, classe j prévue.

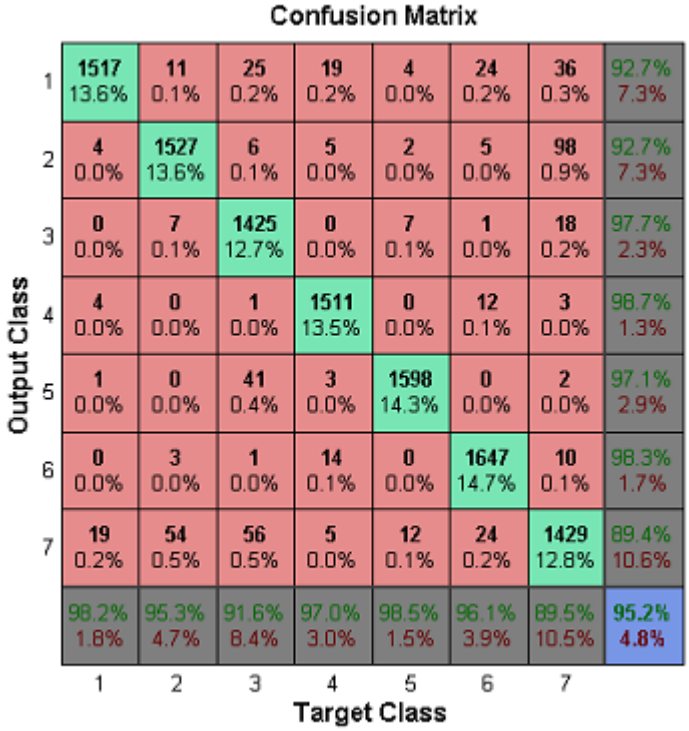

Figure 3.6 : *matrice de confusion.*

Les figures 3.7 et 3.8 présentent les détails des résultats de classification du PNN en utilisant le paramètre WL.

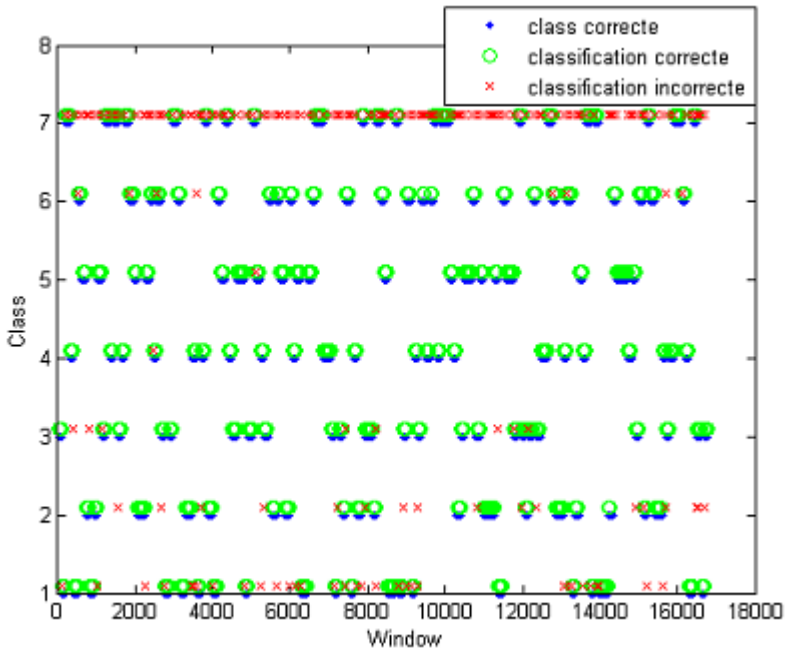

Figure 3.7 : *Résultats de classification du PNN (Coeficient WL avec reduction).*

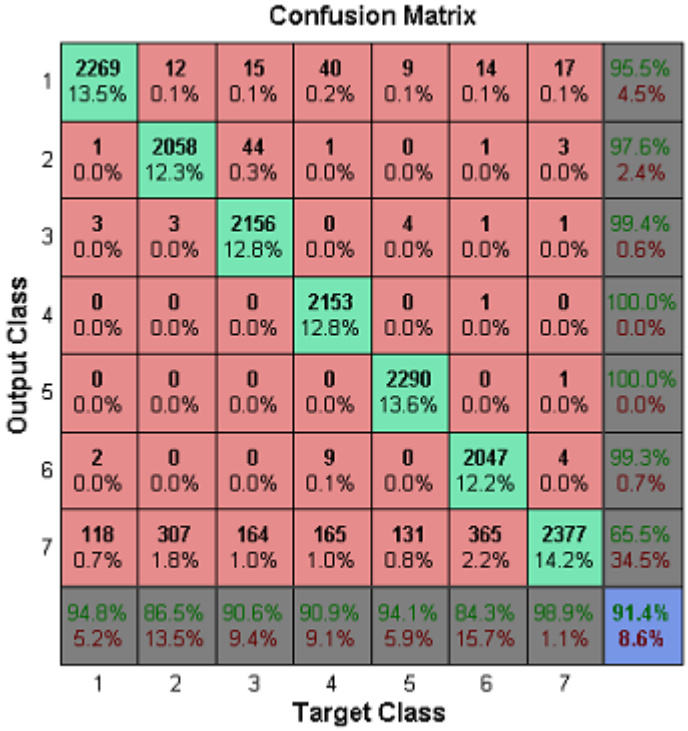

Figure 3.8 : *matrice de confusion.*

3.3.1.2 Discussion

Les résultats obtenus montrent que :

- La réduction de la dimension de l'espace des paramètres avec ULDA, améliore globalement les performances du réseau de classification (MLP et PNN). Cette amélioration est plus percevable pour le réseau PNN.
- Les paramètres les plus pertinents pour un réseau MLP avec réduction des paramètres sont : WL, AR , IAV et RMS.
- Les paramètres AR, RMS, IAV et MAV donnent les meilleurs taux de classification pour le PNN. nous constatons que ce sont presque les mêmes paramètres mais dans un ordre différent.
- Les paramètres fréquentiels caractérise moins les signaux EMG en comparaison aux paramètres temporels.
- En utilisant la réduction de dimension de l'espace des paramètres, les performances du réseau MLP sont légèrement meilleures que celle du PNN. A titre d'exemple, pour un même paramètre de caractérisation (AR) le MLP et le PNN ont un taux de classification de 95.29% et 95.02% respectivement.

3.3.2 Etude 2 : Contribution des canaux dans l'identification des mouvements

Cette étude est menée à titre démonstratif pour observer l'aptitude globale de chaque canal à l'identification des différents mouvements(tableaux : 3.3 et 3.4). Les résultats sont calculés pour un seul paramètre (RMS).

canal	canal 1	canal 2	canal 3	canal 4	canal 5	canal 6	canal 7	canal 8
Taux (%)	41.84	30.81	52.27	41.47	50.22	46.76	28.79	41.46

Tableau 3.3 : *Contribution de chaque canal dans la classification avec le MLP.*

canal	canal 1	canal 2	canal 3	canal 4	canal 5	canal 6	canal 7	canal 8
Taux (%)	36.72	28.53	48.33	34.03	45.51	45.05	22.59	45.23

Tableau 3.4 : *Contribution de chaque canal dans la classification avec le PNN.*

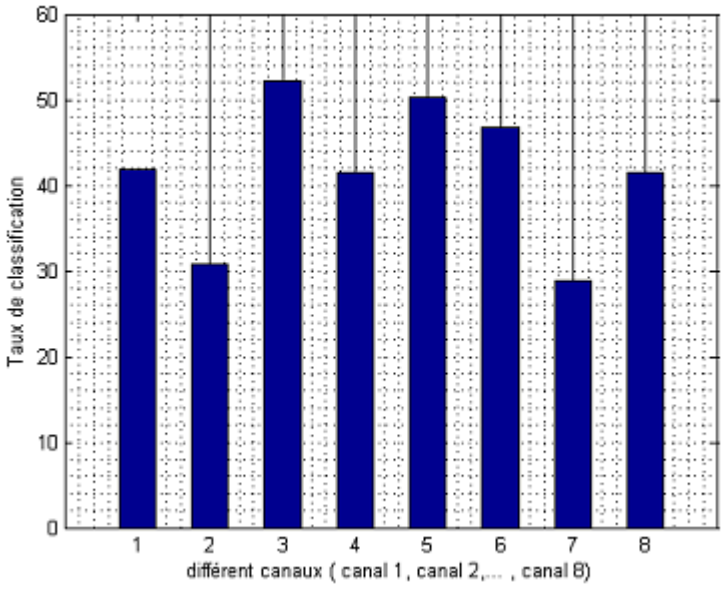

Figure 3.9 : *Contribution de chaque canal dans la classification avec le MLP.*

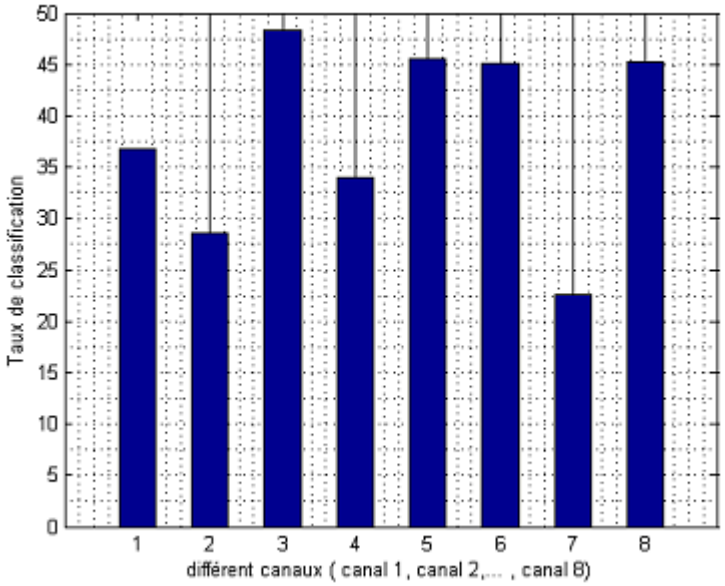

Figure 3.10 : *Contribution de chaque canal dans la classification avec le PNN.*

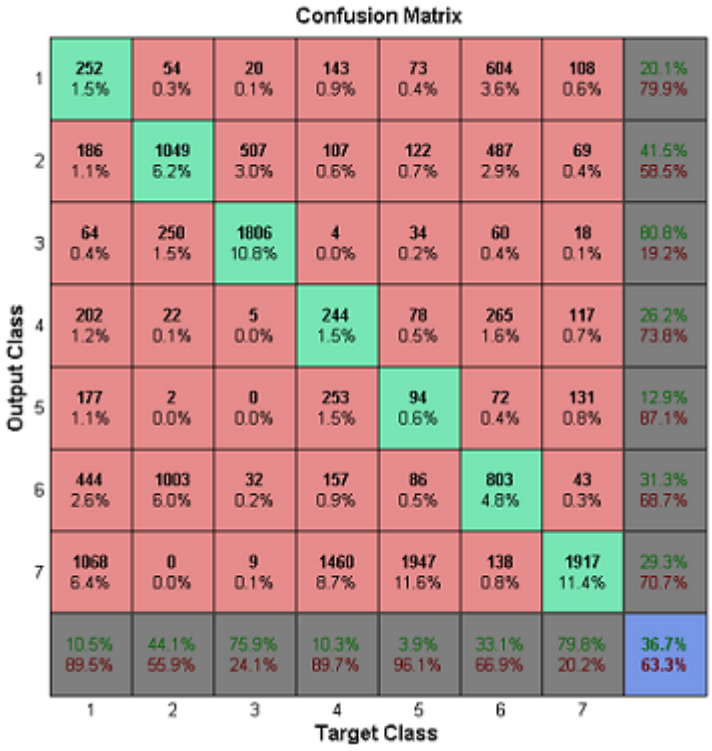

Figure 3.11 : *Matrice de confusion du PNN en utilisant le canal 1.*

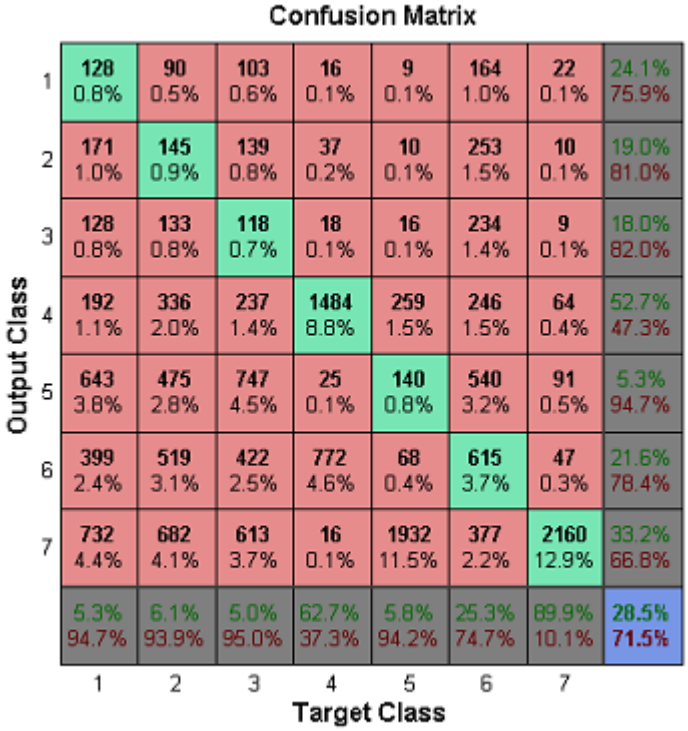

Figure 3.12 : *Matrice de confusion du PNN en utilisant le* 2^{eme} *canal.*

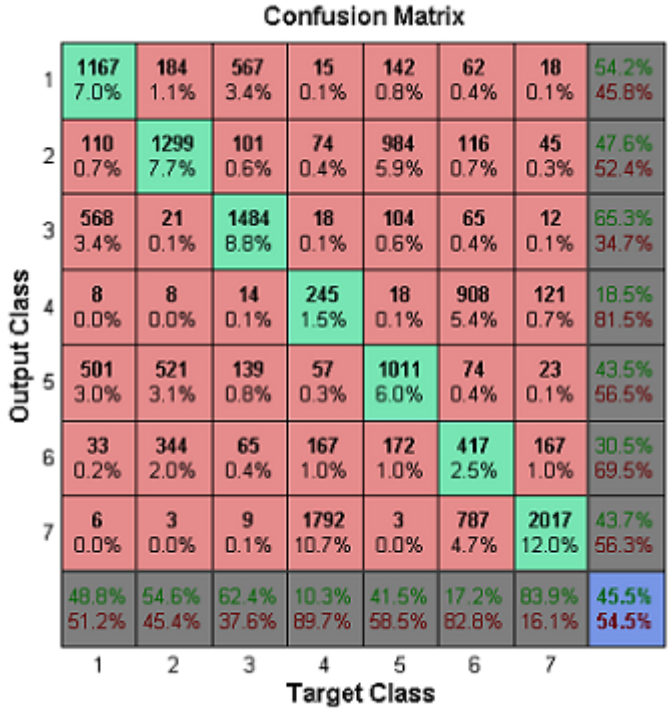

Figure 3.13 : *Matrice de confusion du PNN en utilisant le* 5^{eme} *canal.*

3.3.2.1 Discussion

Les canaux identifie différemment les classes. Le canal 1, par exemple, peut identifier 36.7% des classes considérées. En représentant sont taux de bonne classification par rapport à chaque classe (figure 3.11), nous constatons qu'il nous informe sur certaines classes mieux que d'autre. Les classes les mieux identifiées sont les classes 7 (repos) et 3 (flexion du poignet) à des taux de 79.8% et 75.9% respectivement. la classe très mal identifiée est la classe 5 (supination) avec un taux de 3.9%.

Le canal 2 est peu informatif pour toutes les classes(figure 3.12), à l'exception de la classe 1 (main ouverte) avec un taux de bonne classification de 89.9% .

Le canal 5 est informatif par rapport à toutes les classes (45.5%). Les classes les mieux distinguées sont les classes 7 et 3 avec des taux de bonne classification de 83.9% et 62.3% (figure 3.13). La classe la moins distingué est la classe 4 (extension poignet) avec un taux 10.3%.

Ces résultats identifient les muscles impliqués dans la réalisation d'un mouvement donné. Chaque mouvement du bras est exécuté par plusieurs muscles particuliers. L'optimisation du taux de classification globale doit obligatoirement passer par le choix rigoureux de l'emplacement des electrodes (cibler les muscles intervenant dans les mouvement et écarter les autres).

3.3.3 Etude 3 : influence de la combinaison de paramètres sur le taux de classification

Dans cette etude, en utilise des combinaisons de plusieurs paramètres afin d'améliorer le taux de classification. les combinaisons étudiées sont :

- combinaison des deux meilleurs paramètres Wl et AR pour le MLP, RMS et AR pour le PNN.
- combinaison des deux paramètres fréquentiels Fmed et Fmoy.
- combinaison de paramètres temporels et fréquentiels.

Selon les résultats précédents, les paramètres temporels donnent de grand taux de classification comparés à ceux fréquentiels.
Néanmoins, par cette étude, nous voulions vérifier l'amélioration du taux de classification en combinant l'aspect fréquentiel à travers les fréquences médiane et moyenne avec 2 paramètres temporels (RMS et IAV) de bonnes performances.

Les résultats de simulation sont résumés dans le tableau 3.5.

paramètres	WL et AR pour MLP RMS et AR pour PNN	Fmed et Fmoy	RMS, IAV, Fmoy et Fmed
MLP	97.36	85.93	95.74
PNN	97.24	87.05	96.29

Tableau 3.5 : *Taux de classification du MLP et PNN en utilisant différentes combinaisons.*

3.3.3.1 Discussion

Les résultats du tableau 3.5 montrent que :

- La combinaison des deux meilleur paramètres temporel augmente le taux de classification au delà de 97% pour les deux réseaux.
- La combinaison des paramètres fréquentiels, n'améliore pas les performances des deux réseaux.
- la combinaison de deux paramètres temporel de bonne performance (RMS, IAV) et les deux paramètres fréquentiels donne de bon résultats.
- la meilleure performance pour les deux classifieurs neuronaux a été obtenue en utilisant la combinaison des paramètres temporels (Wl, AR) pour le MLP (97.36%) et (RMS, AR) pour le PNN (97.24%).

3.4 Decision sur le processus de classification

3.4.1 Réseau de neurones et paramètres

Les résultats obtenus précédemment montrent l'efficacité des paramètres temporels pour la caractérisation des signaux SEMG et la nécessité de la réduction pour améliorer les performances des classifieurs neuronaux. Pour la suite de notre travail, nous adopterons un réseau de neurone de type PNN. les paramètres de caractérisation sont une combinaison de deux paramètres temporels de bonne performance, à savoir *RMS* et *AR*. Leur utilisation engendrera un espace de representation de 5 paramètres pour chaque canal (AR(4)+RMS(1)=5), l'équivalent de 40 paramètres pour tout les canaux. L'algorithme *ULDA* sera utilisé pour réduire cette dimension à 7 paramètres.

3.4.2 Test des performances du classifieur sur l'ensemble de la base de données

Dans cette partie, nous présentons l'aptitude de généralisation du classifieur retenu pour différentes sessions d'exercice et de différents individus en exploitant la base de signaux SEMG que nous disposons. Les résultats sont résumés dans les tableaux suivants :

Individus N01	Session 1	Session 2	Session 3	Session4
individu 1	97.24	97.22	97.32	97.28
individu 11	83.15	82.34	84.00	85.08
individu 22	91.75	90.42	90.36	91.42

Tableau 3.6 : *Taux de classification des quatres sessions d'exercice de plusieurs individus.*

Individus	N01	N02	N03	N04	N05	N06	N07	N08	N09	N10
Taux (%)	97.28	92.25	88.84	94.41	85.44	96.07	93.13	91.28	96.39	91.71
Individus	N11	N12	N13	N14	N15	N16	N17	N18	N19	N20
Taux (%)	85.08	93.68	93.42	93.74	96.15	89.89	93.47	88.13	94.94	94.94
Individus	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30
Taux (%)	87.65	91.42	94.68	89.87	91.09	92.63	96.89	91.62	86.50	92.47

Tableau 3.7 : *Taux de classification de tous les individus*

Le taux de classification en considérant la session 4 seulement pour tous les individus est compris entre 85.02% (individu 11) et 97.28% (individu 1). Le taux moyen

de classification est alors de 92.17%.
Le comportement du classifieur est différent d'un individu à l'autre, Cependant les résultats qu'il fourni sont assez satisfaisants.

3.5 Conclusion

Lorsque nous optons pour une solution de type réseaux de neurones pour traiter un problème de classification des signaux physiologiques, les questions fondamentales à résoudre sont : le choix d'une structure de réseau adaptée à la complexité du problème à traiter et les paramètres à utiliser pour caractériser ces signaux.

Divers paramètres ont été utilisés pour caractériser les signaux SEMG, et l'etude 1 a montré que les paramètres fréquentiels sont moins performants que les paramètres temporels pour caractériser ces signaux.

Les etudes 1,2 et 3 ont montré la nécessité de réduction de l'espace des entrées des réseaux de neurones pour la classification des signaux SEMG. Elle est trés apparente lors de l'utilisation du réseau probabiliste en combinant plusieurs paramètres pour la caractérisation des signaux (28.72% sans réduction et 96.29% avec réduction).

Le choix de l'emplacement des electrodes est primordiale pour améliorer le taux de classification du fait que chaque mouvement est réalisée par des muscles particuliers.

Deux type de réseau ont été traité, le MLP et le PNN. Le principal problème rencontré lors de l'utilisation du réseau MLP est la détermination de sa structure à savoir le nombre de couches et le nombre de neurones dans chaqu'une d'elle.
Même si les réseaux MLP et PNN ont tous les deux donné de bon taux de classification, il convient de noter quelque différences entre ces deux réseaux. La première porte sur leurs architectures : tandis qu'elle est figé sur deux couche pour le PNN, l'architecture d'un MLP peut comporter plusieurs couches. Le deuxième point qui les différencie est le nombre de neurones à utiliser dans la couche cachée (dans le cas ou le MLP est utilisé avec une seule couche cachée) : pour un PNN, ce nombre est égale au nombres de classes à différencier.

Après cette étude, le réseau adopté pour la suite de notre travail, qui est l'implementation de ce dernier sur un FPGA est le réseau PNN. Notre choix est motivé par la simplicité de sont architecture ainsi que sa taille et ces performances comparées au réseau MLP.

CHAPITRE 4

Implémentation matérielle de réseau PNN sur FPGA

4.1 introduction

La mise en oeuvre d'application pour la classification des signaux SEMG sur ordinateur est souvent adoptée. Cependant, la mise en oeuvre d'interface humaine pratique, exige que l'appareil soit compact et portatif comme les prothèses myoélectrique contrôler du bras supérieur.
Ces dernières années, l'augmentation des performances des circuits logiques programmables permet l'implementation de n'importe quelle architecture de réseau de neurone, et ainsi concevoir sur une même puce un système capable de classifier et contrôler d'un bras supérieur.
Dans ce chapitre, nous allons décrire les circuits logiques programmables et plus spécifiquement les FPGA. Par la suite, nous aborderons l'architecture matérielle implanté du réseau de neurones décrit dans le chapitre précédent.

4.2 Circuit logique programmable

Le principe de la logique programmable remonte au début des années 1960, il a cependant fallu attendre les années 1980 pour que les premières réalisations matérielles apparaissent sur le marché. L'apparition de ce type de circuit s'est d'abord faite au travers de circuits logiques programmables simples de type PAL (Programmable Array Logic), qui se programment comme des mémoires non volatiles de type ROM et sont utilisés pour implémenter des fonctions combinatoires simples, telles des décodeurs d'adresse, ou des contrôleurs de bus.
Avec les évolutions en microélectronique, différentes familles de circuits programmables ont commencé à apparaître : les CPLD (Complex Logic Programmable Device), puis les FPGA (Field Programmable Gate Arrays), introduits par la société Xilinx en 1985. L'industrialisation de ce type de circuits s'est faite à grande échelle avec l'apparition de circuits de plus en plus performants et reprogrammable à volonté. Les

circuits de type FPGA les plus récents offrent désormais l'équivalent de dix millions de portes logiques programmables, à des fréquences de fonctionnement atteignant les 200MHz[23].

4.2.1 Présentation des FPGA

le FPGA permet d'avoir une architecture conçue sur mesure à haute densité dans un circuit intégré, avec la possibilité de modifier cette architecture quand des nouvelles applications apparaissent. Les langages HDL (Hardware Description Langage) comme le VHDL ou le Verilog sont utilisés pour décrire les fonctionnalités qui seront implémentées sur le composant, puis la description matérielle est traduite dans un fichier de configuration pour le FPGA cible[24].

4.2.1.1 Architecture générale

L'architecture, retenue par Xilinx (figure 4.1), se présente sous la forme de deux couches distinctes, la première est la couche appelée circuit configurable et la deuxième est une couche de réseau mémoire SRAM.

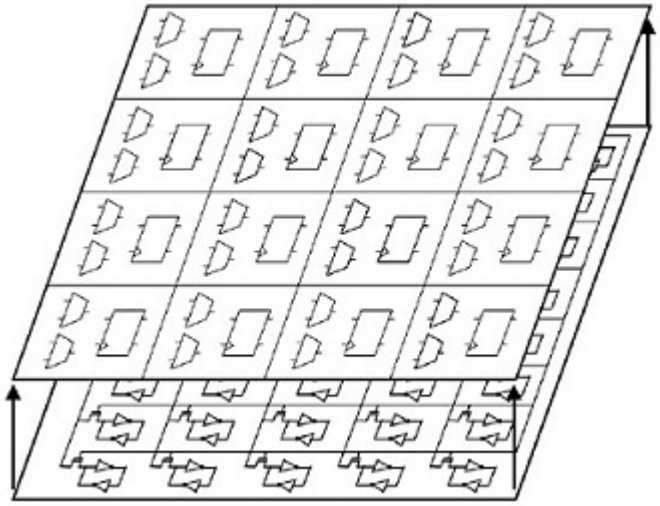

Figure 4.1 : *Architecture FPGA retenue par Xilinx, la première est la couche appelée circuit configurable et la deuxième est une couche de réseau mémoire SRAM*

La première couche (circuit configurable) est constituée d'une matrice de blocs logiques configurable (CLB - Configurable Logic Bloc). Les CLB permettent de réaliser des fonctions séquentielles et combinatoires. Autour de ces blocs logiques configurable, nous trouvons les blocs entrées/sorties (IOB - Input Output Bloc)). Ils permettent de gérer les entrées-sorties pour réaliser l'interface avec les modules extérieurs.

La seconde couche est un réseau de mémoire SRAM qui permet la programmation du circuit FPGA. La programmation est réalisée en appliquant les potentiels adéquats sur la grille de certains transistors à effet de champ pour interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux. Ces potentiels sont mémorisés dans le réseau de mémoire SRAM. Un dispositif interne au FPGA permet à chaque mise sous tension de charger les SRAM internes à partir de la ROM externe où est stockée la configuration du FPGA[25].

4.2.1.2 Architecture interne de FPGA

Les circuits FPGA du fabricant Xilinx utilisent deux types de cellules de base, les cellules d'entrées/sorties appelées IOB et les cellules logiques appelées CLB. Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions configurable. Donc les trois blocs principaux (figure 4.2) sont les blocs logiques configurables, les blocs d'entrées/sorties et les ressources de communications.

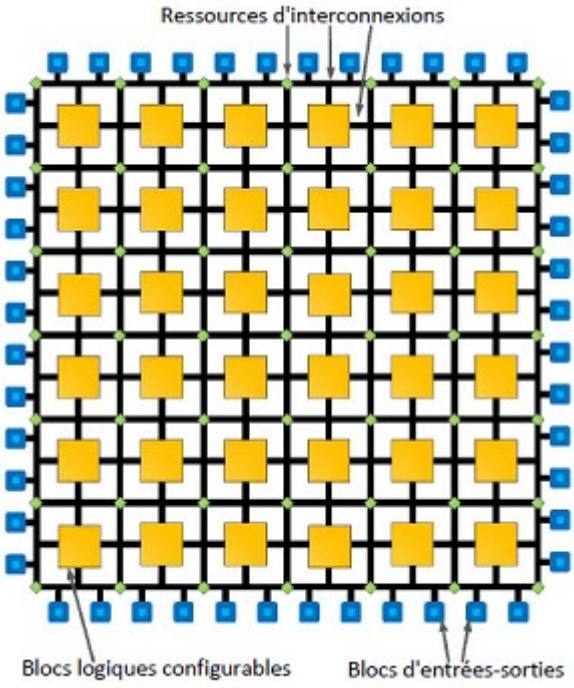

Figure 4.2 : *Architecture interne d'un FPGA*

4.2.1.3 Blocs principaux

Les blocs logiques configurables (CLB) Les blocs logiques configurable sont les principaux éléments d'un FPGA. Ils peuvent avoir un ou plusieurs générateurs de fonctions réalisées avec des tables de correspondance (LUT - look-up tables)

qui peuvent mettre en oeuvre une logique arbitraire en fonction de leur configuration. Autour d'une LUT, il y a une logique d'interconnexion qui permet les liaisons à destination et vers la LUT.Elle est mise en oeuvre à l'aide de portes logiques et de multiplexeurs. Pendant le processus de configuration d'un FPGA, les mémoires des LUT sont écrites pour y implémenter une fonction, et la logique qui l'entoure est configurée pour router correctement les signaux afin de construire un système complexe. Il existe différents types de CLB en fonction du FPGA utilisé. Pour Xilinx le bloc logique configurable FPGA est appelé slice.

Les blocs d'entrée-sortie (IOB) Les blocs d'entrée-sortie permettent l'interconnexion de la logique interne aux ports d'entrées et de sorties du FPGA. Les IOB ont leur propre mémoire de configuration, elle stocke les standards de tension et la direction des ports. Ces blocs sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé (haute impédance).

Les ressources d'interconnexion Les ressources d'interconnexion au sein d'un FPGA permettent la connexion arbitraire des CLB et des IOB. Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons :

- Les interconnexions directes
- Les longues lignes
- Les matrices d'interconnexion

comme le montre la figure 4.3

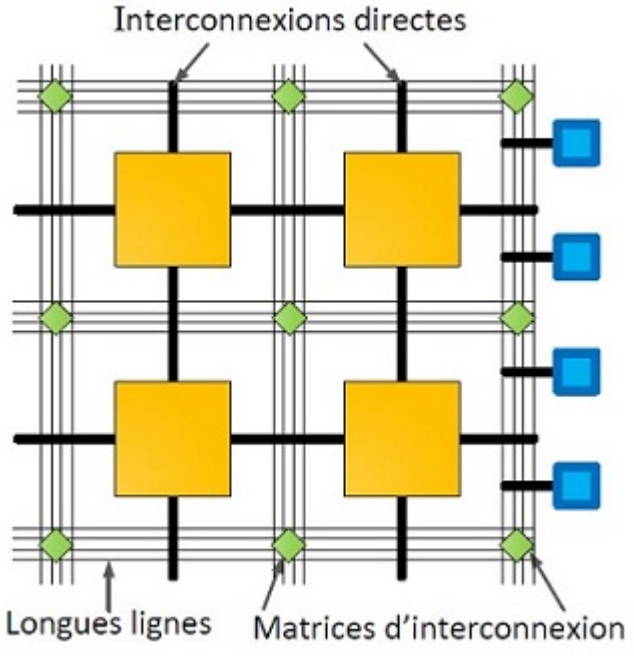

Figure 4.3 : *Interconnecxion interne d'un FPGA*

Toutes ces configurations sont stockées dans des SRAM qui sont volatiles : lorsque le composant est mis hors tension, sa configuration est perdue et il doit être redémarré avec une nouvelle configuration. Habituellement, une machine externe se charge de télécharger la configuration sur le FPGA via une de ses interfaces de configuration, et envoie une commande de démarrage pour signaler que la configuration a eu lieu. Certaines cartes ont une mémoire ROM d'intégrée. Elle permet de stocker la configuration, de sorte qu'elle puisse ensuite être téléchargée sur le FPGA. Dans ce cas, les données de configuration sont copiées sur la mémoire SRAM de configuration du FPGA au démarrage de celui-ci[25].

4.2.1.4 Bloc à usage spécifique

En plus des trois blocs décrits dans le paragraphe précédent, les FPGA possèdent souvent ce que l'on appelle des ressources additionnelles. Le fonctionnement et le placement de ces blocs diffèrent d'une référence de FPGA à une autre[25][24].

Les blocs RAM (BRAM) Les blocs RAM sont des mémoires définies par l'utilisateur, embarquées sur le circuit intégré FPGA, elles servent à stocker des ensembles de données. En fonction de la catégorie de FPGA, la mémoire RAM embarquée est configurable en blocs de 16 ou de 32 kilo-octets. Les mémoires RAM sont de type double ports, elles permettent une écriture et une lecture indépendante sur chaque port avec une horloge différente. Ceci est très utile, le composant peut produire (écrire) des données à une fréquence différente d'un autre composant

qui consomme (lire) les données.

Digital Signal Processing (DSP) Les *Digital Signal Processing* sont des blocs qui permettent des conceptions plus complexes, qui peuvent consister soit en traitement numérique du signal ou seulement certains assortiments de multiplication, addition et soustraction. Comme pour les BRAM, il est possible de mettre en oeuvre ces blocs grâce au CLB, mais il est plus efficace en termes de performances, et de consommation d'énergie d'intégrer plusieurs de ces composants au sein du FPGA. Un bloc DSP permet de réaliser un multiplicateur, un accumulateur, un additionneur, et des opérations logiques (AND, OR, NOT, et NAND) sur un bit. Il est possible de combiner les blocs DSP pour effectuer des opérations plus importantes, telles que l'addition avec virgule flottante simple, la soustraction, la multiplication, la division, et la racine carrée. Le nombre de blocs DSP est dépendant du dispositif.

Les processeurs embarqués Les processeurs embarqués enfouis sont l'un des ajouts les plus importants pour le FPGA. Beaucoup de conceptions nécessitent l'utilisation d'un processeur embarqué. Souvent, le choix d'un dispositif de FPGA avec un processeur embarqué (comme le Virtex de Xilinx 5) permet de simplifier grandement le processus de conception tout en réduisant l'utilisation des ressources et la consommation d'énergie. Le PowerPC IBM 405 et 440 processeurs sont des exemples de deux processeurs inclus dans le Virtex 4 et 5 de Xilinx. Ce sont des processeurs RISC classiques qui mettent en oeuvre un jeu d'instructions PowerPC.

Le gestionnaire d'horloge numérique (DCM) Un gestionnaire d'horloge numérique permet d'avoir des périodes d'horloge différentes qui sont générées à partir d'une horloge de référence unique. La plupart des systèmes disposent d'une horloge externe unique qui produit une fréquence d'horloge fixe. Cependant, il y a un certain nombre de raisons pour lesquelles un concepteur peut avoir besoin de fonction logique fonctionnant à des fréquences différentes. L'avantage d'utiliser un DCM est que les horloges générées auront moins de gigue.

4.2.2 Processus d'implémentation

L'expansion du monde des circuits logiques programmables et en particulier des circuits FPGAs est étroitement liée au développement de la technologie de fabrication des circuits intégrés permettant la fabrication d'unités de plus en plus denses. Mais cette expansion n'aurait jamais pu avoir une telle ampleur sans le développement, en parallèle, d'outils performants permettant de faciliter le processus de conception des circuits logiques avec de la logique programmable. Ces outils assurent d'une part la description aisée des circuits logiques complexes et d'autre part la réalisation automatique de circuit à partir de ces descriptions.

La réalisation d'un circuit logique, en fait le passage de sa description à l'information nécessaire pour la réalisation d'une implementation physique du circuit dans un FPGA (la configuration). Ce processus est habituellement décomposé en plusieurs étapes distinctes, représentées schématiquement à la Figure 4.4. Actuellement, grace aux outils informatisés, certaines étapes de ce processus peuvent être réalisées de façon entièrement automatique (notamment la synthèse, l'implementation ainsi que la génération de la configuration pour le FPGA cible) [24].

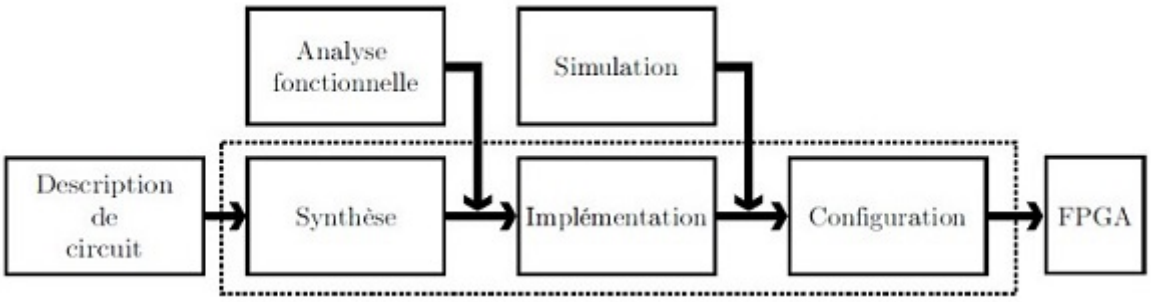

Figure 4.4 : *Différentes étapes du processus du désigne*

Description La première étape du design flow comprend la description formelle du circuit à partir des spécifications. Aujourd'hui la description des circuits logiques est faite la plupart de temps à l'aide des langages de description de matériel (Hardware Description Language - HDL) à haut niveau d'abstraction tels que Verilog ou VHDL. Souvent, lorsque l'utilisateur souhaite rester proche du matériel affin d'obtenir la meilleure performance possible de son circuit, la description est faite de façon schématique. La description d'un circuit peut être faite exclusivement à l'aide d'un formalisme bien particulier, ou combiner des formalismes différents. Les différentes parties d'un même circuit logique complexe peuvent donc être décrites de façon différente avant d'être traduites en une configuration particulière pour un (ou plusieurs) FPGA cible.

Synthèse Lors de l'étape de la synthèse, les différentes descriptions établies lors de la première étape sont ramenées à une description du circuit unique, optimisée pour une performance maximale ou une occupation minimale des ressources. Après l'étape de la synthèse il est possible d'effectuer une analyse fonctionnelle du circuit (vérification de la description à travers la simulation) et de se rendre compte d'éventuelles erreurs, avant de passer à l'étape de l'implementation, généralement considérée comme lente. L'analyse fonctionnelle n'est qu'une vérification partielle car elle ne peut tenir compte des délais introduits par le circuit réel, celui-ci n'étant pas encore réalisé. Lors d'une analyse fonctionnelle tous les délais du circuit obtenu après la synthèse sont considérés comme nuls

Implementation L'étape de l'implementation est généralement divisée en trois sous-

étapes distincts : le mapping, le placement et le routage. Le mapping consiste à traduire le circuit logique obtenu par le processus de synthèse en mémoires des cellules logiques élémentaires d'un circuit FPGA particulier. Lors de la sous-étape de placement, une cellule logique existante, avec sa position unique au sein du FPGA cible, est attribuée à chaque cellule logique nécessaire pour la réalisation du circuit. Ceci est bien entendu fait sous la contrainte des longueurs minimales des connexions entre les différentes cellules. Enfin la dernière sous-étape, le routage, consiste à établir les différentes connexions physiques entre les cellules élémentaires placées. Lorsque l'étape de l'implementation est achevée, il est possible de réaliser une simulation réaliste de fonctionnement du circuit, car à ce stade on dispose de l'information précise sur les délais réels qui seront introduits par le circuit physique (vérification de l'implementation).

Configuration Le processus de design flow se termine par la génération du fichier de configuration correspondant au circuit réalisé. Ce fichier peut être directement transféré dans le circuit FPGA cible depuis la plate-forme du développement ou depuis une mémoire programmable dans le cas d'une solution embarquée. Le circuit conçu devient alors opérationnel.

4.3 Implantation de réseaux de neurones sur FPGA

4.3.1 Architecture de base

l'architecture matérielle du réseau PNN utilisée est déduite de l'architecture d'un MLP présentée dans [26]. Pour expliquer cette architecture , nous considérerons un réseau avec la structure suivante (figure 4.5) :

- 5 entrées.
- couche cachée contenant 7 neurones.
- 3 sorties.

la sortie d'un neurone est régie par l'équation suivante :

$$Y_j^{(L)} = F(S_j^{(L)}) = F(\sum_{i=1}^{N_{L-1}} W_{ij}^L X_i^{L-1})$$

avec :

- W_{ij}^L : poids entre le neurone i de la couche (L-1) et le neurones j de la couche L.
- X_i^{L-1} : sortie du neurone i dans la couche (L-1).
- $F(.)$: fonction d'activation de type sigmoïde.

Les données entrantes sont envoyées en série au réseau de neurones et arrivent par paquets A, B,... etc. Les paquets contiennent les I éléments de chaque entrée. L'architecture se décompose en deux niveaux : le calcul de couche cachée (CC) et celui de la couche de sortie (CS).
Les cinq entrées (codées sur 18 bits) arrivent séquentiellement et sont fournies directement aux cinq multiplieurs disposés en parallèle.A_j correspond à la j^{me} entrée de l'ensemble présent et B_j correspond à la j^{me} entrée de l'ensemble suivant. Les entrées sont synchronisées sur l'horloge et sont maintenues pendant H cycles d'horloge. Les multiplieurs permettent de réaliser le produit des entrées avec leurs poids respectifs stockés dans des mémoires ROM. Chacune de ces mémoires contient l'ensemble de poids reliant une entrée à tous les neurones de la couche suivante.A chaque cycle d'horloge, une accumulation est réalisée et la somme transite à travers tous les additionneurs. La sortie du dernier additionneur est transmise à un opérateur de remise en forme (ReF) afin de transformer le signal initialement codé sur 48 bits en un signal sur 16 bits. Ce signal adresse ensuite une mémoire tabulant la fonction sigmoïde d'activation (SIGMOID0) en 65536 valeurs de 18 bits.
Les résultats de la couche cachée sont transmis à la couche de sortie pour faire les multiplications-accumulations (MAC) correspondantes. Chaque unité MAC calcule la somme pondérée associée à un neurone de sortie. Dans l'étage CS, chaque mémoire ROM contient l'ensemble des poids reliant une sortie aux différents neurones de la couche cachée. Après l'opération MAC, les données sont à nouveau mises en forme sur 16 bits afin d'adresser la seconde mémoire sigmoïde d'activation. Le flot de données est multiplexé de manière à réduire l'utilisation des mémoires.
L'étage de contrôle permet le séquencement des différentes opérations dans l'architecture. Il permet la synchronisation des multiplications dans la couche cachée par le biais de la commande de lecture dans la ROM. Le procédé est le même en ce qui concerne les unités MAC de la couche de sortie. Enfin, c'est aussi l'étage de contrôle qui commande le multiplexage des données en sortie du réseau de neurones.

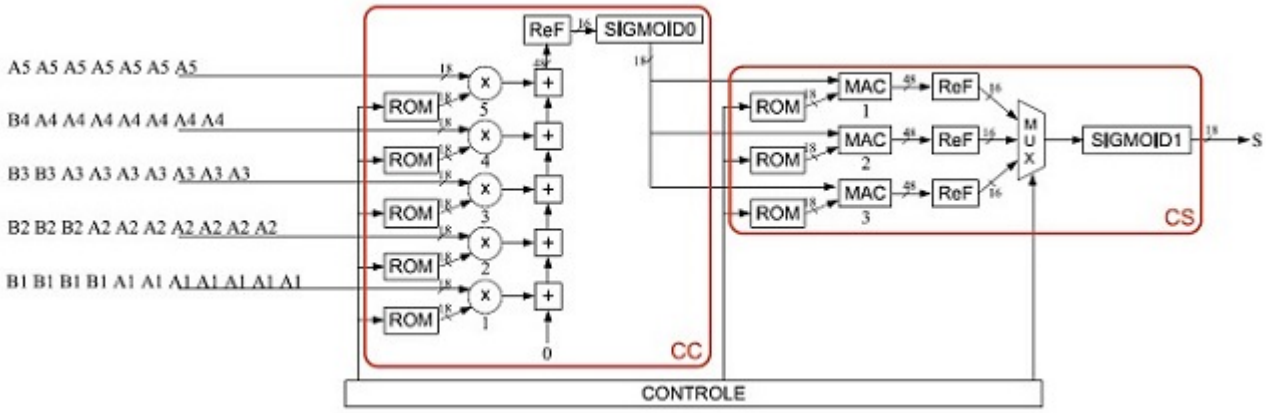

Figure 4.5 : *Exemple d'architecture matérielle d'un MLP*

4.4 développement de l'implémentation du PNN

4.4.1 L'outil System Generator

Afin de countouner la complexité du développement du l'architecture implantant le réseau PNN avec un langage HDL, nous avons utilisé l'outil Xilinx System Generator (SysGen), qui nous a permis de modéliser notre système et de le vérifier à partir de Simulink.

cet outil permet de créer pour les algorithmes des modèles optimisés pour l'implementation dans les FPGA Xilinx, en utilisant des composants de haut niveau à partir des bibliothèques Xilinx. ces bibliothèques incluent des blocs pour la logique de contrôle, le traitement du signal, la communication, les fonctions mathématiques et les mémoires. SysGen permet aussi d'intégrer du code HDL existant, des fonctions MATLAB et des composants matériels ciblés pour les FPGA Xilinx afin de créer des modèles de systèmes complets simulable dans l'environnement Simulink. Une fois développé, le système peut être compilé dans une netlist HDL prêt à être traitée par un outil de synthèse et implémentée avec les outils de conception physique Xilinx. System Generator produit en plus les fichiers de projet et de contraintes pour Xilinx ISE Design Suite, ainsi qu'un banc d'essai VHDL (testbench) pour une vérification fonctionnelle.

Avant de procéder à l'implementation de l'architecture du PNN en utilisant SysGen, il faut d'abords traiter deux aspects importants qui sont le codage des données (nombres signés) et la méthode d'implantation de la fonction exponentielle impliquée dans le calcul de la sortie de la première couche selon l'algorithme du PNN.

4.4.1.1 Représentation des données

Il existe 2 manières pour représenter les nombres réels dans les systèmes numériques pratiques : la représentation en virgule fixe et en virgule flottante. En représentant les nombres en format virgule flottante (32bits ou 64bits), les résultats seront plus précis du fait de sa capacité de gérer un bien plus grand nombre de valeurs, mais cette précision nécessite un circuit plus complexe à cause de la complexité de l'arithmétique associée, ce qui engendre une plus grande utilisation des resources du circuit FPGA. La représentation en virgule fixe présente l'avantage de l'arithmétique simplifiée (circuit moins complexe et facile à concevoir). De ce fait, la méthode du point fixe sera utilisée pour représenter les données dans notre système. Pour la représentation du signe, nous utiliserons le format complément à 2, qui est le plus utilisé et le plus approprié pour l'arithmétique.

Le deuxième aspect important est la longueur des mots binaires manipulés, c-à-d le nombre de bits à considérer pour les parties entière et fractionnaire. Ces nombres doivent être déterminés en fonction de la dynamique des valeurs et la précision souhaitée. Pour notre application, les entrées du PNN (paramètres) seront codées sur 16 bits en réservant 15 bits pour la partie fractionnaire. La taille des données varie selon l'étage considérée à causse de la succession des opérations arithmétiques de multiplication, addition et soustraction. Pour le dépassement de capacité, SysGen permet plusieurs manières pour arranger le résultat(troncature, arrondi, saturation...). Ces alternatives sont résumées par la figure 4.6 qui présente la configuration de la sortie d'un additionneur à titre d'exemple.

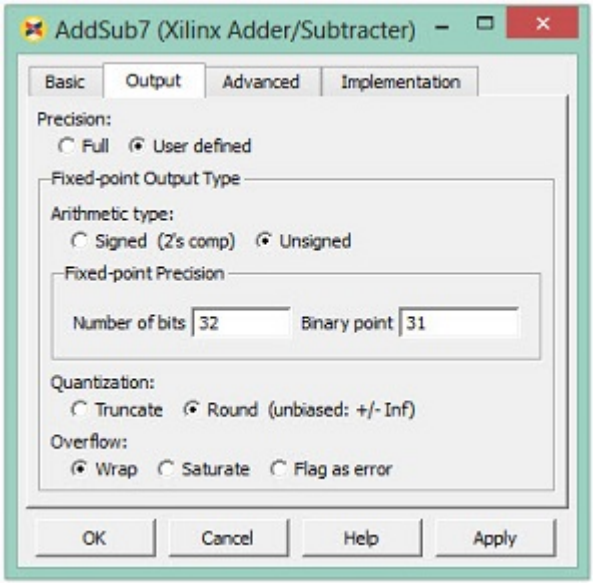

Figure 4.6 : *Options de la représentation en virgule fixe*

4.4.1.2 Implementation de la fonction exponentielle

les principales méthodes pour implémenter l'exponentielle utilisent soit des LUT (Look Up Table) dans lesquelles sont stockés des échantillons pré-calculés, soit une décomposition en série de Taylor de la fonction exponentielle pour la traiter avec des opérations simples. Pour la solution à base de LUT, on obtient la valeur de $EXP(X_i)$ en accèdent à l'adresse i de cette table. La précision des échantillons stockés en mémoire varie en fonction de la taille, plus le nombre d'échantillons stockés dans la ROM est important, plus l'intervalle entre 2 échantillons successifs est faible. Pour obtenir une bonne précision, il est souhaitable d'utiliser une ROM aussi importante que possible. Cela a cependant des conséquences négatives particulièrement sur la surface occupée.

L'alternative que nous avons adoptée consiste à utiliser un processeur CORDIC disponible sous forme d'un bloc dans la bibliothèque de SysGen. *CORDIC* (sigle de COordinate Rotation DIgital Computer : " calcul numérique par rotation de coordonnées ") est un algorithme de calcul des fonctions trigonométriques et hyperboliques, notamment utilisé dans les calculatrices. A précision égale, un module CORDIC occupe beaucoup moins de surface qu'une architecture à base de LUT. Le module CORDIC sera configuré pour délivrer à ses sorties, le sinus et le cosinus hyperboliques figure 4.7. La fonction exponentielle est alors générée en utilisant la relation exp(x)=sinh(x)+cosh(x).

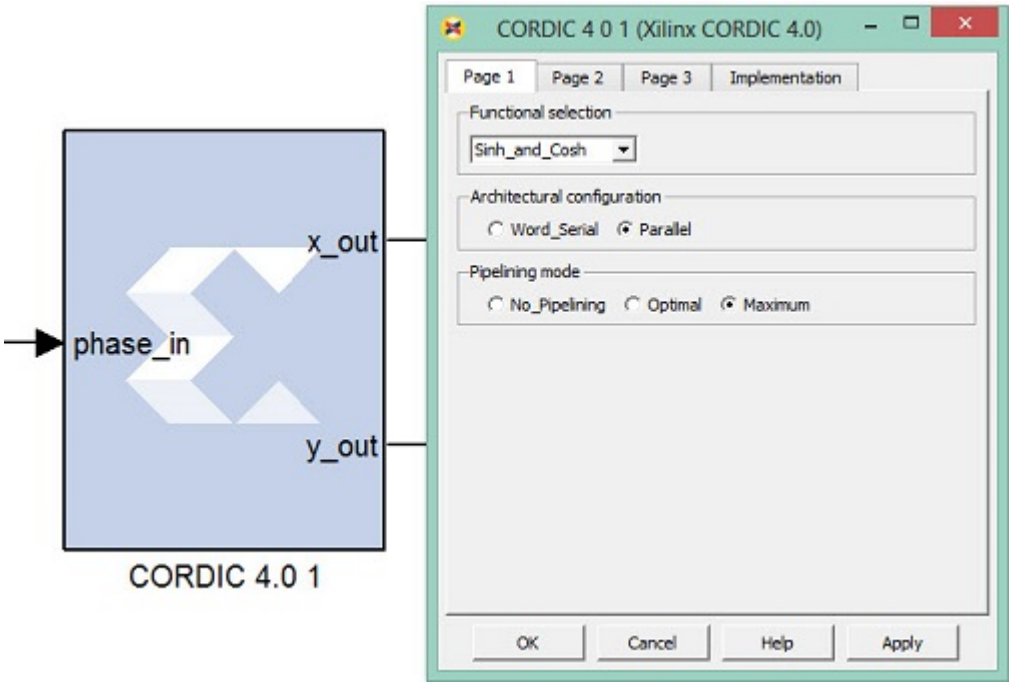

Figure 4.7 : *Module CORDIC et sa configuration*

4.4.2 Architecture du réseau PNN

Le réseau PNN à implementer est décrit par la figure 4.8. Ce réseau ce compose de 7 entrées, une couche cachée de 7 neurones et 7 neurones de competition pour la couche de sortie.

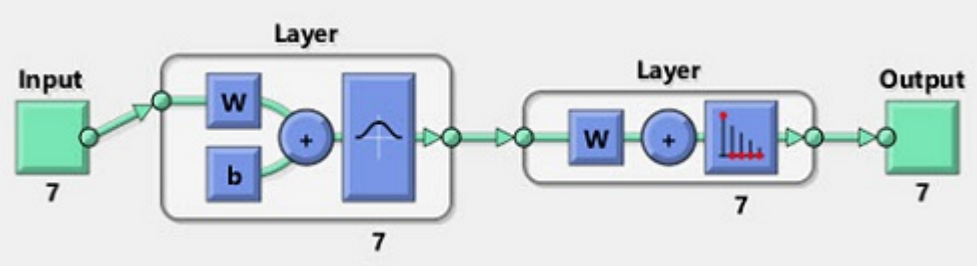

Figure 4.8 : *Structure du réseau PNN à implementer*

4.4.2.1 Structure de la 1^{er} couche

Les 7 entrées sont appliquées séquentiellements aux 7 étages de calcul disposés en parallèle. l'application de chaque valeur d'entrée est synchronisée avec l'horloge et maintenue pendant 7 cycles. Chaque étage (figure 4.9) permet de calculer le carrée de la différence entre l'entrée et le poids mémorisé. les 7 mémoires contiennent l'ensemble des poids reliant une entrée à tous les neurones de la couche. la somme cumulée fournie par le dernier étage chaque top d'horloge correspond à : $S_H = \sum_{i=1}^{7}(P_i - W_{iH})^2$, avec i : numéro de l'étage.

Cette somme est transmise au bloc *FCN* (figure 4.12) Ce bloc réalise la fonction : $X = FCN(S_H) = -(\beta\sqrt{(S_H)})$, avec β : biais.
La sortie du bloc FCN est fournie au bloc exponentiel (figure4.11) qui calcul la sortie de la 1^{er} couche.

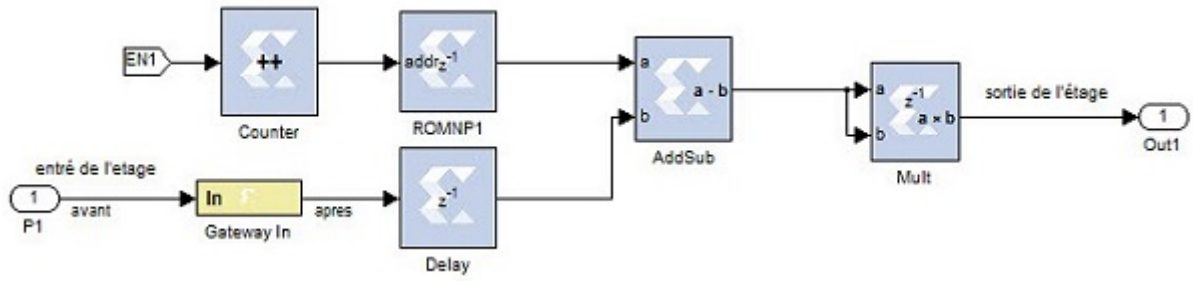

Figure 4.9 : *Structure de l'étage 1*

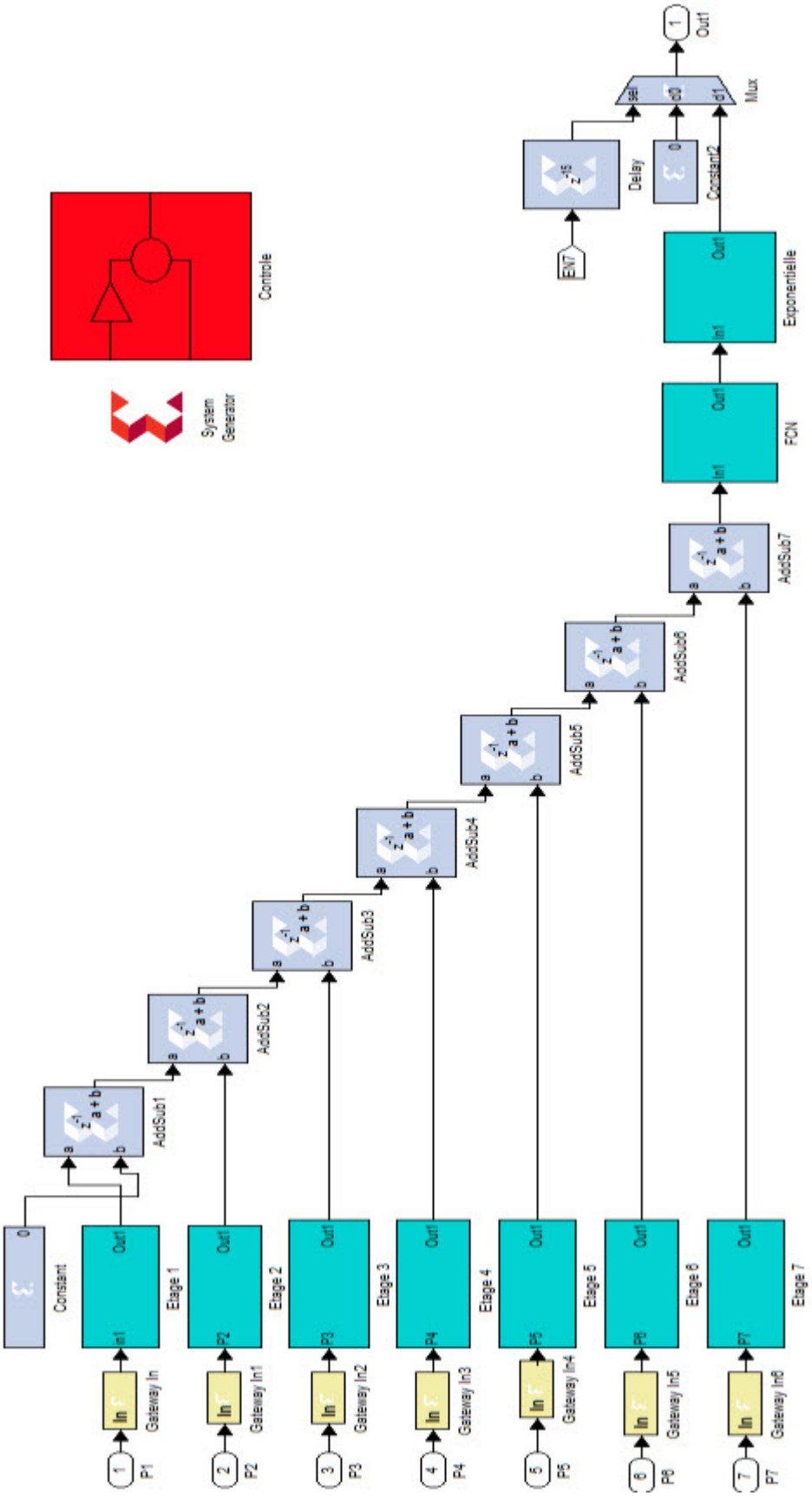

Figure 4.10 : *Modèle de la couche 1*

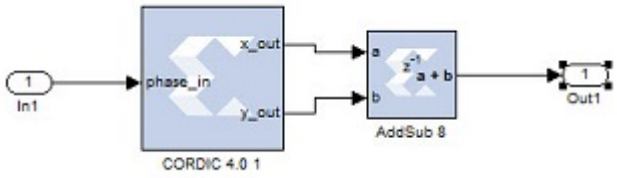

Figure 4.11 : *Structure du bloc exponentiel*

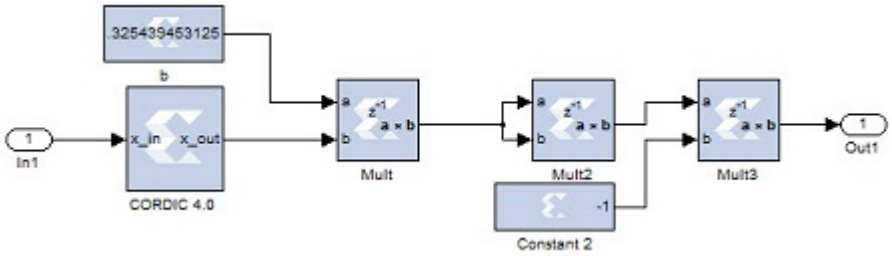

Figure 4.12 : *Structure du bloc FCN*

La figure 4.13 montre la structure de l'étage de contrôle responsable de la synchronisation du fonctionnement des différents blocs.

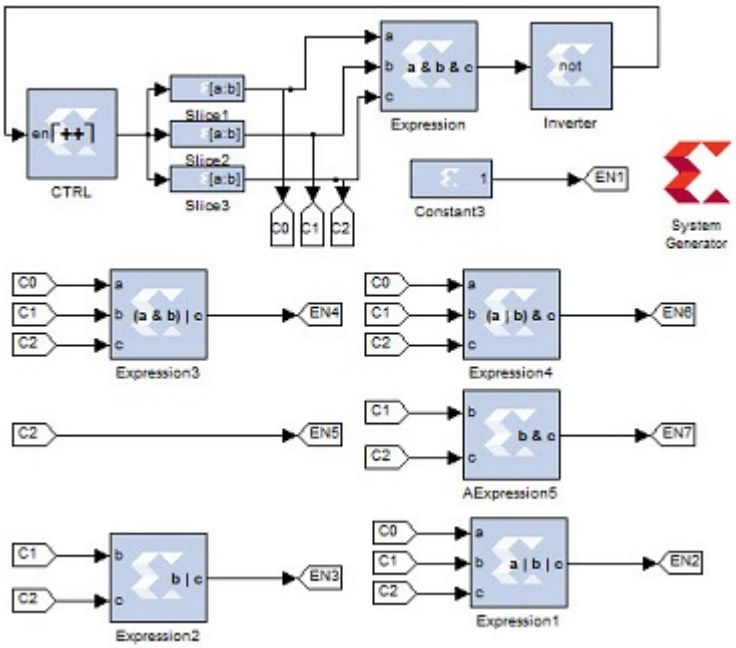

Figure 4.13 : *Structure de l'étage de contrôle*

La figure 4.14 présente les valeurs des entrées avant et après codage en virgule fixe sur 16 bits. l'erreur de quantification engendrée est donnée par la figure 4.15.

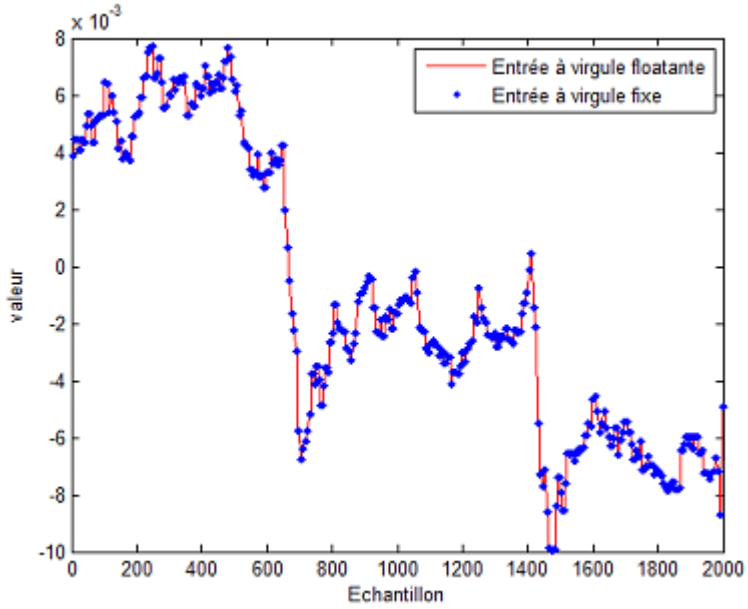

Figure 4.14 : *influence du codage sur 16bits.*

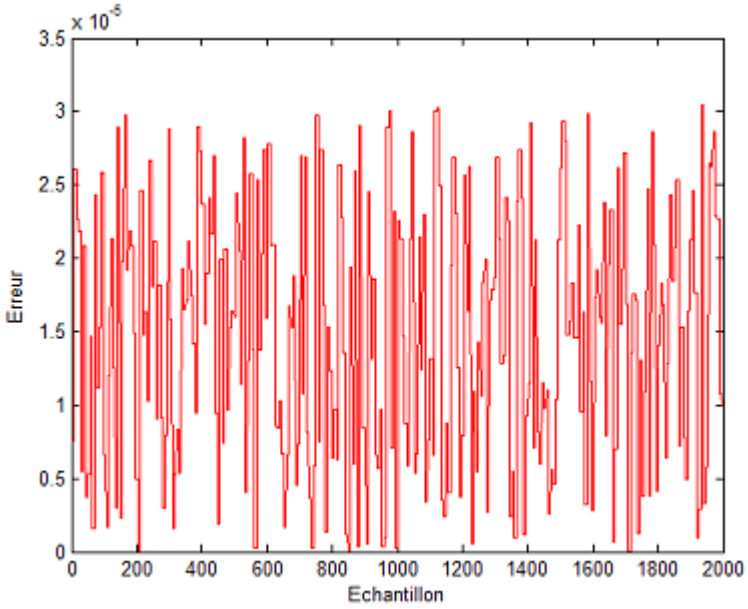

Figure 4.15 : *Erreur de quatification*

La figure 4.16 montre que les valeurs générées par la 1^{er} couche du circuit sont presque identique à ceux obtenus avec le calcul en utilisant les fonctions Matalb.

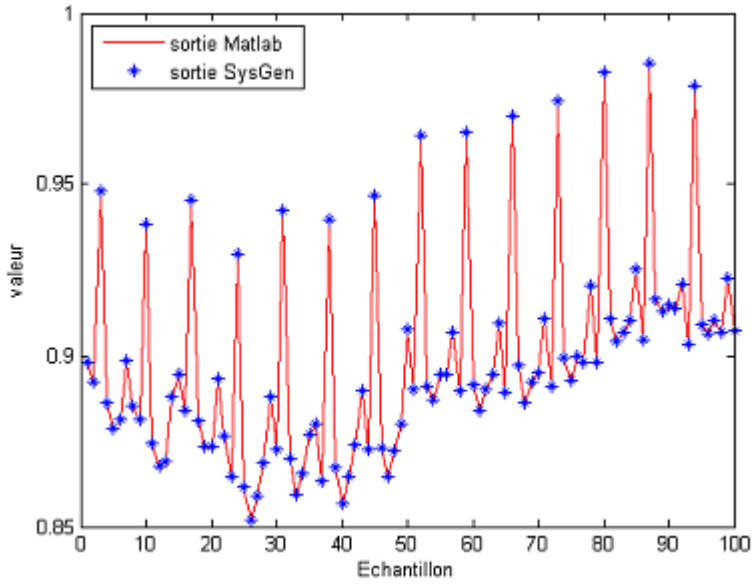

Figure 4.16 : *Performance du circuit de la couche 1*

4.4.2.2 Structure de la 2^{eme}couche

Les résultats de la couche cachée sont transférés en série à la couche de sortie. la fonction de cette couche consiste à comparer les 7 valeurs reçus représentant les sorties des neurones précédents et déterminer la classe : l'indice de la valeur maximale représente le numéro du mouvement correspondant (figure4.17).

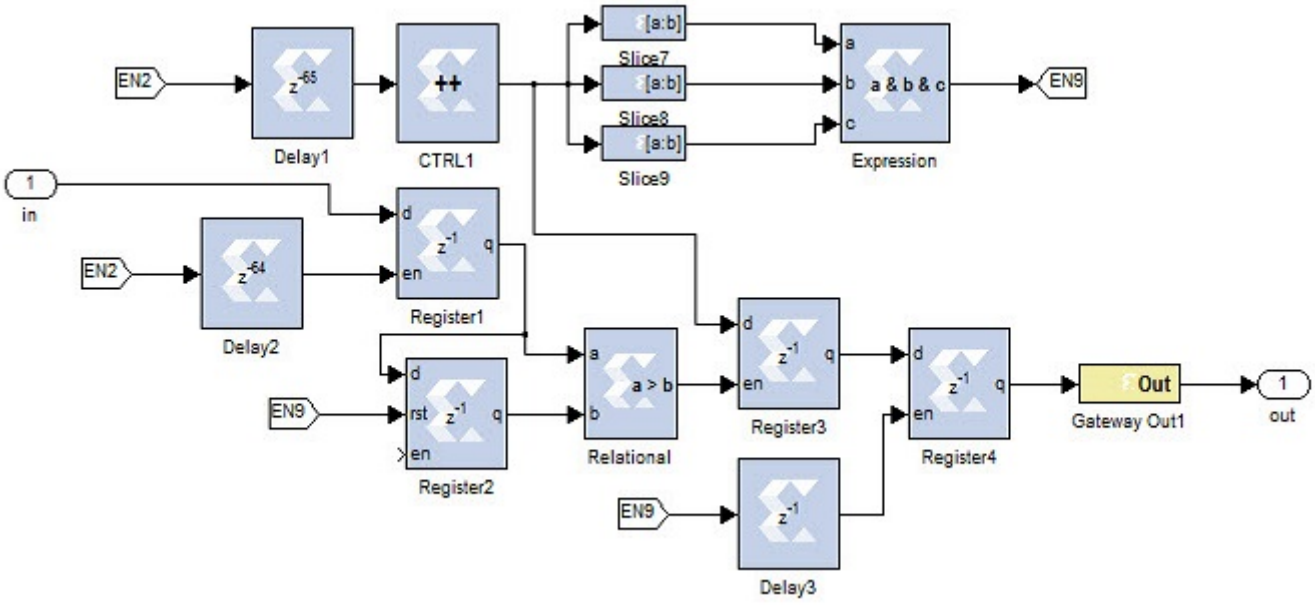

Figure 4.17 : *Modèle de la couche 2*

Les deux registres de comparaison sont les registres 1 et 2. pour chaque cycle, le circuit effectue une comparaison entre la nouvelle valeur reçus (registre 1) et la valeur maximale sauvegardée (registre 2), si la nouvelle valeur et supérieure : les registres 2 et 3 sont actualisés par la nouvelle valeur et son indice. A la fin de toutes les comparaisons, la valeur du registre 3 est transférée au registre 4 qui représente la classe du mouvement.

4.4.3 Co-simulation Matlab/SysGen

4.4.3.1 Simulation sous SysGen

les différents blocs développés ont été testés et validés par simulation sous simulink. La figure 4.18 donne quelques résultats de simulation réalisée pour vérifier le bon fonctionnement des étages de calcul de la couche 1. cette figure montre le séquencement des données aux entrées du circuit. La 1^{er} entrée est disponible dans le 1^{er} étage de calcul au 2^{eme} coup d'horloge et sera traitée puis transmise au 1^{er} additionneur. La 2^{eme} entrée est appliquée une période après. La sortie du 2^{eme} étage arrivera au deuxième additionneur au même temps que la sortie de son prédécesseur. La même opération se répète pour les autres étages. La sortie du 7^{eme} étage sera disponible aux 8^{eme} coup d'horloge et sera additionnée avec la sortie de l'additionneur 6. La somme accumulée est générée au 10^{eme} cycle. La latence de cette partie du circuit est égale à 10.

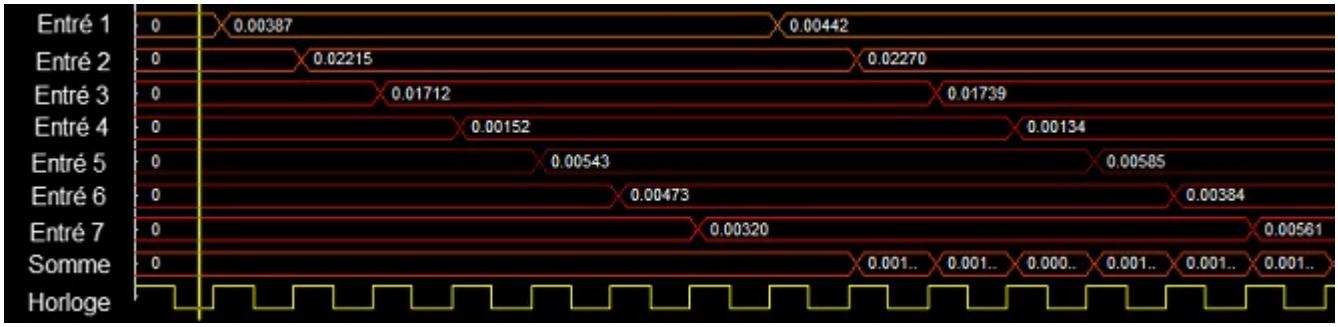

Figure 4.18 : *séquencement des données à l'entrée de la couce 1*

La figure 4.19 présente les sorties des blocs *FCN* et *Exponentiel*. La somme est disponible au 10^{eme} coup d'horloge, celle ci est transmise au bloc *FCN* pour executer les opérations décrite dans la figure 4.12. La succession de ces opérations dure 19 périodes qui représente la latence de ce bloc.

La sortie du bloc FCN est transmise au bloc Exponentiel. Ce bloc fournie sa première sortie valide après 38 périodes. En effet, ce bloc fournie une réponse 10 cycles après le bloc FCN. La durée totale des opérations effectuées dans la 1^{er} couche pour

obtenir la 1^{er} sortie valide est de 65 périodes. Cette valeur est considerer comme la latence totale du circuit de la 1^{er} couche.

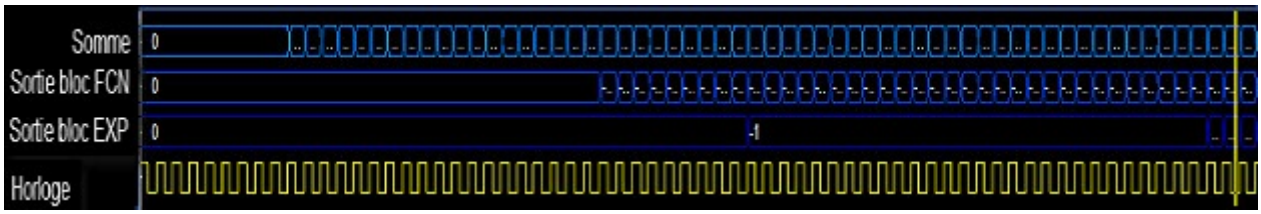

Figure 4.19 : *Séquence d'execution des différents bloc de la couche 1*

La figure 4.20 décrit la séquence d'execution des différentes fonctions de la couche 2. La 1^{er} donnée valide est attendu a la 65 période. Les registres utilisé sont en retard d'une période par rapport a la disponibilité d'une donné a leur entré. Le bloc de comparaison fournie une sortie binaire à l'instant que les registres 1 et 2 lui fournisse les données a comparé.

La latence de cette 2^{eme} couche est égale a la somme des latence des registres 1, 3 et 4. La 1^{er} sortie de tout le circuit sera disponible à la 68^{eme} période (latence de tout le circuit C=68).

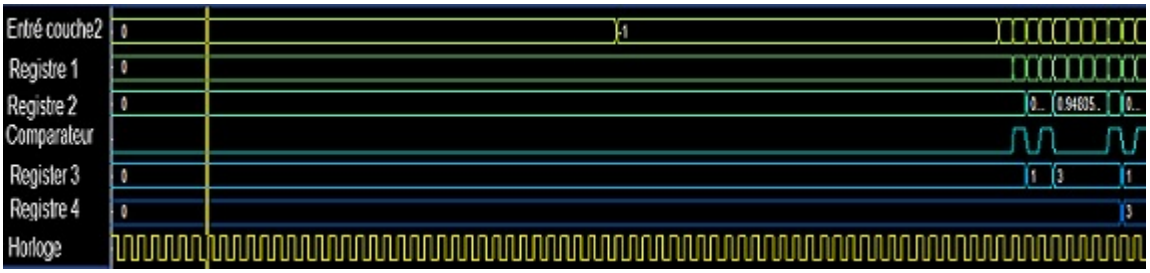

Figure 4.20 : *Séquence d'execution des différents bloc de la couche 2*

4.4.3.2 Résultat de la co-simulation

La figure 4.21 représente les résultât de la co-simulation en utilisant Matlab et SysGen.

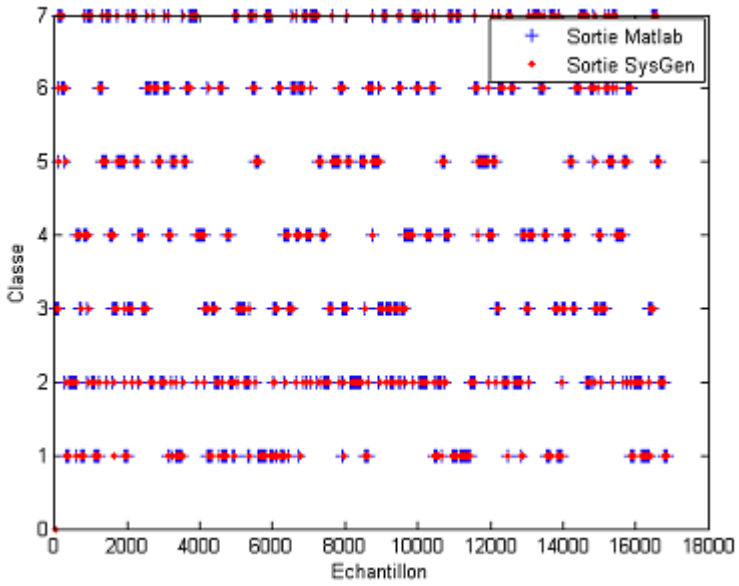

Figure 4.21 : *Résultat de la co-simulation*

La figure 4.22 représente l'erreur de classification du circuit par rapport au résultat obtenue avec Matlab.

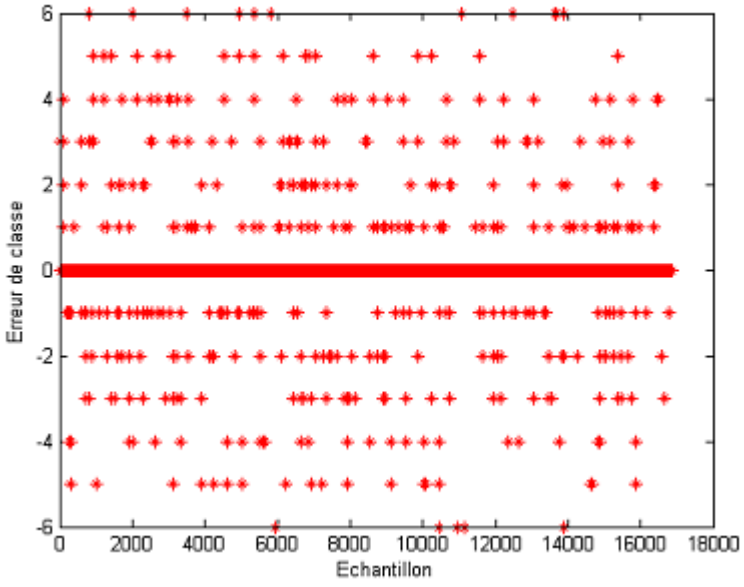

Figure 4.22 : *Erreur de classification par rapport au résulats obtenu sous Matlab.*

Les figures 4.21 et 4.22 montre une différence entre les résultats obtenu avec Matlab et système generator. Cela pourrait s'expliquer par la perte de précision en representant les données en virgule fixe sous système generator.

4.4.4 Performance obtenue

L'architecture à entièrement été simulée et implantée dans un circuit FPGA de la famille Xilinx Spartan 3E. Le circuit obtenue peut fonctionné à une fréquence maximal de 63.243 Mhz avec une période minimal de 15.812 ns.
La figure 4.23 résume les ressources occupé aprés l'implémentation.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	4,999	9,312	53%
Number of 4 input LUTs	4,844	9,312	52%
Number of occupied Slices	2,739	4,656	58%
Number of Slices containing only related logic	2,739	2,739	100%
Number of Slices containing unrelated logic	0	2,739	0%
Total Number of 4 input LUTs	5,014	9,312	53%
Number used as logic	4,714		
Number used as a route-thru	170		
Number used as Shift registers	130		
Number of bonded IOBs	115	232	49%
IOB Flip Flops	16		
Number of RAMB16s	7	20	35%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	12	20	60%
Average Fanout of Non-Clock Nets	1.92		

Figure 4.23 : *resources occupé*

4.5 Conclusion

Dans ce dernier chapitre, on à commencer par la présentation des circuit logique programmable de type FPGA. On à présenter leur structure interne, les bloc principaux et les bloc à usage spécifique d'on il dispose.

La deuxième partie de ce chapitres à été consacré à la description d'une architecture matériels pour implementation de réseaux de neurones de type MLP. Cette architecture nous à servi de reference pour aborder l'implementation d'un réseau PNN qui est relativement plus complexe.

La dernière partie est consacré à l'implementation d'un réseaux neurones de type PNN. Système generator à été adopter comme outils pour décrire le réseaux neuronal pour sont implementation à partir des spécification du réseau simuler sous MAtlab. La simulation du circuit à montré sa bonne performance (89.6%). Néanmoin on a constater une légère diminution du taux de classification par rapport au résultat sous Matlab (97.2%). Cela est due à la representation des entrées en virgule fixe. La representation des données en vergule fixe diminue la complexité du circuit au detriment de ces performances.

Conclusion Générale

Les objectifs principaux fixé dans le cadre de ce memoire sont atteints, il consiste à dimensionner un classifieur neuronal permettant de différencie 7 mouvement d'un bras et d'implementer ce dernier sur un circuit logique programmable de type FPGA.

Nous avon vue que plusieurs traitement sont successivement appliquer aux signaux SEMG pour qu'il puissent être interprétée. Les deux 1^{er} étape consiste ainsi à l'extraction des caractéristiques les plus discriminantes et réduire leur dimension de representation. Puis un classifieur neuronal est utiliser pour différencie les mouvements du bras.

Deux structures de réseaux de neurones on été utilisé comme classifieur : le réseau PNN et le réseau MLP. Ces deux architecture on montré leur efficacité pour la séparation des classes des différents mouvements. Néanmoins, leur performance dépende de plusieurs critères :l'emplacement des electrode pour recueillir les signaux SEMG, le type de paramètres utiliser pour la caractérisation des signaux et en fin la dimension des representation des paramètres ainssi extraie.

La technologie basé sur les circuit comme les FPGA à atteint un haut niveau de performance permettant ainsi l'implementation de reseaux neurones. Les résultats ainsi obtenue démontre la fesabilité de l'implementation matériel des réseaux PNN. Reste à revoir certain traie de la conception du circuit et augmenté ces performance.

Au terme de ce travail, nous estimons que nos efforts ont été récompensés par des résultats assez intéressants qui nous ont permis de nous imprégner de connaissances techniques nouvelles sur la technologie de conception des système embarqué en général contribuant ainsi à notre formation.

Bibliographie

[1] G. C. G. Adrian D C Chan, *MYOELECTRIC CONTROL DEVELOPMENT TOOLBOX*, Department of Systems and Computer Engineering Carleton University Ottawa ON.

[2] R. Boostani and M. H. Moradi, "Evaluation of the forearm emg signal features for the control of a prosthetic hand," *Physiol*, 2003.

[3] L. OUKHELLOU, "Paramétrisation et classification de signaux en contrôle non destructif." Ph.D. dissertation, UNIVERSITÉ DE PARIS-SUD, 1997.

[4] S. BRETON, "Une approche neuronale du contrôle robotique utilisant la vision binoculaire par reconstruction tridimensionnelle," Ph.D. dissertation, UNIVERSITE DE HAUTE-ALSACE U.F.R. DES SCIENCES ET TECHNIQUES, 1999.

[5] C. TOUZET, *INTRODUCTION AU CONNEXIONNISME*, 1992.

[6] M. Parizeau, *RESEAUX DE NEURONES*. Université LAVAL, 2004.

[7] M. Y. AMMAR, "Mise en oeuvre de reseaux de neurones pour la modelisation de cinetiques reactionnelles en vue de la transposition batch/continu." Ph.D. dissertation, INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE, 2007.

[8] P. Wira, "Des réseaux de neurones artificiels pour l'identification et le contrôle de systèmes électriques," in *International Conference on Electrical Engineering and its Applications (Sidi Bel-Abbes)*, 2008.

[9] stephane huet, "classification de signaux myoelectriques," 2004.

[10] G. M. Dreyfus G, Martinez M, *reseaux de neurones*. Eyrolles, 2002.

[11] P. Thomas, "Contribution à l'identification de systèmes non linéaires par réseaux de neurones," Ph.D. dissertation, UNIVERSITE HENRI POINCARE, NANCY 1, 1997.

[12] H. CAO, "Modelisation et evaluation experimentale de la relation entre le signal emg de surface et la force musculaire," Ph.D. dissertation, Universite de Technologie de Compiegne.

[13] samer CHENTAF, "Biometrie par signaux physiologiques," Ph.D. dissertation, université de paris.

[14] P. konrad, *The ABC of EMG*. Noraxon inc, 2005.

[15] P. J. P. Roberto Merletti, *Electromyography : Physiology, Engineering, and Non-Invasive Applications*. Wiley-IEEE Press (Wiley), 2004.

[16] R. e. h. B. Sofiane BOUDAOUD, "Identification du signal emg de surface par un système de multiples réseaux de neurones," *Université ferhat ABBAS de Sétif, Algérie*, 2000.

[17] V. MAHEU, "Développement des critères d'apprentissage pour le controle d'un bras robot manipulateur à 7 ddl par traitement des signaux emg chez les blés-sés médullaires," Master's thesis, École de technologie supérieure université du québec, 2011.

[18] B. Pasquet, "Etude de la specificite de la commande motrice et de sa regulation pendant differents types de contractions musculaires," Ph.D. dissertation, Université libre de bruxelles, 2009.

[19] S. H. S. V. T. H. Luca Mesin, Stuart Smith, "Effect of spatial filtering on crosstalk reduction in surface emg recordings," *Medical Engineering & Physics*, 2009.

[20] M. CHENDEB, "Détection et classification des signaux non stationnaires par utilisation des ondelettes. application aux signaux électromyographiques utérins," Ph.D. dissertation, UNIVERSITE DE TECHNOLOGIE DE TROYES, 2006.

[21] M. DIAB, "Classification des signaux emg utérins afin de détecter les accouchements prématurés," Ph.D. dissertation, Université de Technologie de Compiègne, 2007.

[22] A. D. C. Chan, "http ://www.sce.carleton.ca/faculty/chan."

[23] S. DERRIEN, "Étude quantitative des techniques de partitionnement de réseaux de processeurs pour l'implantation sur circuits fpga," Ph.D. dissertation, Université de Rennes 1, 2002.

[24] D. Milojevic, "Implémentation des filtres non-linéaires de rang sur des architectures universelles et reconfigurables," Ph.D. dissertation, Université Libre de Bruxelles, 2004.

[25] N. MARQUES, "Méthodologie et architecture adaptative pour le placement eficace de taches matérielles de tailles variables sur des partitions reconfigurables," Ph.D. dissertation, Université de Lorraine, 2012.

[26] J.-C. P. L. K. Sonia KHATCHADOURIAN, Narayanan RAMANAN, "Architecture matérielle pour implantation de réseaux de neurones en temps réel."

[27] L. K. A. Siou, "Modélisation des crues de bassins karstiques par réseaux de neurones. cas du bassin du lez (france)," Ph.D. dissertation, Université de Montpellier II, 2011.