REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE

Mémoire de Magister en Electronique

Option : traitement du signal et communications

Présenté par : Dalila SALHI

THEME

IMPLEMENTATION D'UN CLASSIFICATEUR NEURONAL BASE SUR L'ALGORITHME RPG DES RYTHMES CEREBRAUX SUR FPGA

Encadré par : ***Melle MOUSSAOUI AICHA***

Soutenu le : 18/03/2009

Devant le jury composé de :

Président :	**Mr Mohamed MEHENNI**	**Professeur, ENSP**
Rapporteur :	**Melle Aicha MOUSSAOUI**	**Chargée de Cours, ENSP.**
Examinateurs :	**Melle M'Hania GUERTI**	**Professeur, ENSP.**
	Mme Latifa HAMAMI	**Professeur, ENSP.**
	Mr Rabah SADOUN	**Chargé de Cours, ENSP.**

PROMOTION : 2007/2008

AVANT PROPOS

L'objectif de notre travail qui entre dans le cadre de la préparation du mémoire de magister au sein du laboratoire Signal et Communications du Département d'Electronique de l'Ecole Nationale Polytechnique, consiste en la conception et l'implémentation sur circuit FPGA d'un classificateur neuronal basé sur l'algorithme de la rétro propagation du gradient (RPG) pour les trois rythmes principaux, dans l'activité cérébrale, à noter le rythme alpha, le rythme bêta et le rythme thêta.

SOMMAIRE

INTRODUCTION GENERALE

CHAPITRE I
SYSTEME NERVEUX ET ELECTROENCEPHALOGRAPHIE

CHAPITRE II LES RESEAUX DE NEURONES ARTIFICIELS

CHAPITRE III
L'ALGORITHME DU RETRO PROPAGATION DU GRADIENT

CHAPITRE IV
CONCEPTION SOFTWARE DU CLASSIFICATEUR

CHAPITRE V
L'IMPLEMENTATION HARDWARE DUCLASSIFICATEUR SUR FPGA

INTRODUCTION GENERALE

L'habilité de contrôler ou de faire bouger des objets par une simple pensée (réflexion mentale) était toujours attribuée à des formes de vie d'alians que l'on rencontre dans les films et les romans de science fiction. Faire convertir cette idée en un fait scientifique devient de plus en plus populaire dans la communauté scientifique durant la dernière décennie; même si la première citation de l'idée d'établir une communication directe cerveau machine existe depuis 1973[1]. De nos jours, plusieurs travaux de recherche et de développement ont été menés dont l'objectif est de développer des systèmes complexes (figure 1) ; pour l'aide au diagnostic ; d'assistance de personnes handicapées (communication directe homme machine ou BCI « brain-computer interface ») ; ou même dédier aux systèmes de stimulation thérapeutique (pacemaker neurologique) en utilisant le signal EEG. Ces systèmes complexes constitués de plusieurs étages nécessitent la contribution de plusieurs disciplines telle que la médecine ; la neurologie ; la physio-neurologie ; la mathématique ; l'informatique ; l'électronique, ….etc.

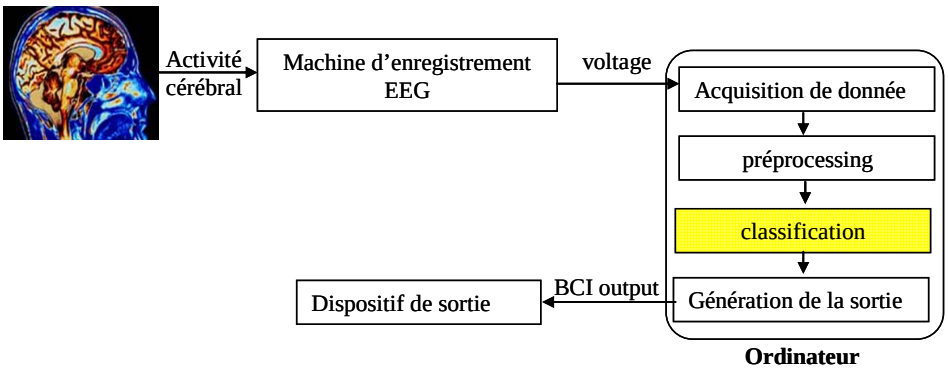


Figure 1. Architecture de base des systèmes de communication Homme-machine

Pour envisager de mettre au point des dispositifs capables de détecter et de distinguer qu'un sujet réalise une activité mentale donnée ; nous avons besoin de classificateurs ou de systèmes de classification [2] capables de discriminer les rythmes cérébraux et les différentes modifications qu'engendrent les différentes activités mentales sur le signal EEG. Cet EEG est considéré comme méthode complémentaire invasive de diagnostic ; qui contient une quantité d'informations importante ; pour le médecin afin de bien situer l'état d'un sujet et pour les concepteurs dans le développement de tels systèmes.

Les progrès technologiques qu'a connu le développement des circuits intégrés ont permis d'obtenir des circuits de très grande fiabilité et à haut taux d'intégration tout en réduisant le coût et la consommation d'énergie. Actuellement, les circuits intégrés spécifiques tels que les FPGAs et les ASICs ont permis une réduction de la taille des systèmes numériques, analogiques ou mixtes et aussi la réalisation des circuits de plus en plus complexes, et très rapides.

L'objet de notre travail, effectué au laboratoire de traitement du signal et communications, est de contribuer à développer des systèmes fiables et usuels par la conception et l'implémentation sur un circuit FPGA d'un classificateur neuronal basé sur l'algorithme de la rétro propagation du gradient (RPG), des trois rythmes principaux dans l'activité cérébrale en utilisant le signal EEG.

Le choix de l'algorithme RPG dans cette application est basé sur l'exploitation de la méthode (off chip training) dans la classification ; chose qui a permis de simplifier l'implémentation hardware réduite dans la partie Feed-Forward.

L'étude théorique de ce travail comporte quatre chapitres :

- le système nerveux : comprendre son fonctionnement, son mode d'action et l'origine de son activité électrique.
- L'étude des réseaux de neurones artificiels et leurs applications en particulier à la classification.
- l'algorithme de la rétro propagation du gradient (RPG) et l'apprentissage supervisé des réseaux de type perceptron multicouches (MLP).
- Les circuits programmables FPGAs en particulier les FPGAs de la famille VIRTEX-II de Xillinx.

La phase de l'étude théorique est suivie par une conception software ; où nous avons construit une base de données (contenant un ensemble de vecteurs) pour pouvoir effectuer l'apprentissage et le test. L'outil « Matlab » est utilisé pour programmer les étapes de l'algorithme RPG appliqué à la classification ; cette programmation nous a permis de déterminer la topographie du réseau et aussi d'estimer les performances de fonctionnement du classificateur pour pouvoir passer à la dernière phase dans notre travail (l'implémentation hardware). Dans cette phase nous avons utilisé l'outil « FPGA advantage » de Mentor Graphics pour la description VHDL et la synthèse des différents composants dans notre architecture, l'outil « ISE Fondation » de Xillinx pour le placement et routage ainsi que l'implémentation. Il est à noter que le circuit ciblé pour l'implémentation est le circuit Virtex-II XC2v1000 de Xillinx [3].

CHAPITRE I

SYSTEME NERVEUX ET ELECTROENCEPHALOGRAPHIE

I. 1 Introduction

Le cerveau est l'organe le plus complexe du corps humain. Son fonctionnement est intimement lié à sa structure infiniment complexe qui présente une variété de couches neuronale sous forme de réseaux fortement connectes. Grâce à ses cent milliards de neurones, autant qu'il y a d'étoiles dans notre galaxie, nous pouvons penser, aimer, planifier, parler, imaginer, faire des merveilles . . .

Contrairement aux autres méthodes d'imagerie fonctionnelle et anatomique du cerveau, l'Electroencéphalographie (EEG) enregistre directement l'activité électrique du cerveau. Grâce a son excellente résolution temporelle, de l'ordre de milliseconde, elle permet de suivre en temps réel la chronologie des opérations mentales, et d'étudier la dynamique des phénomènes cérébraux, dont la compréhension nécessite une connaissance du système nerveux et de son mode de fonctionnement.

A cet effet, dans ce chapitre nous allons faire une présentation du système nerveux : son organisation, son mode de fonctionnement, son activité électrique, et les rythmes cérébraux qui nous intéressent le plus dans notre travail. Par la suite, nous allons donner un aperçu sur l'enregistrement encéphalographie « 'EEG », ces techniques d'enregistrement, les électrodes utilisés ainsi que ses applications.

I. 2 Le système nerveux

I.2.1 Organisation structurelle du système nerveux

Le système nerveux est un système fort complexe qui tient sous sa dépendance toutes les fonctions de l'organisme. Il se compose de centres nerveux, qui sont chargés de recevoir, d'intégrer et d'émettre des informations, et de voies nerveuses qui sont chargées de conduire ces informations [4]. Ce système est en fait constitué de deux types de cellules de base répartis sur trois parties : le système nerveux central (SNC), le système nerveux périphérique (SNP) et le système nerveux végétatif (SNV) [5].

- ✓ **Le système nerveux central (SNC) :** est formé du cerveau, du tronc cérébral et de la moelle épinière. Figure I.1
- ✓ **Le système nerveux périphérique (SNP) :** Il est constitué de nerfs, dont certains recueillent de l'information et d'autres diffusent les ordres. Figure I.1.
- ✓ **Le système nerveux végétatif (SNV) :** Ses nerfs interviennent plutôt dans la régulation des fonctions vitales internes. Ils contribuent donc `a l'équilibre de notre milieu intérieur en coordonnant des activités comme la digestion, la respiration, la circulation sanguine, la sécrétion d'hormones.

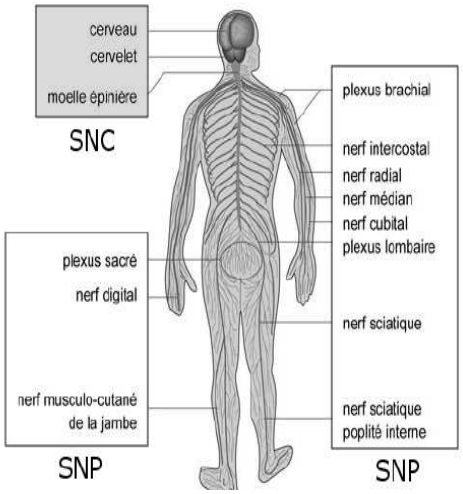


Figure I.1 les grandes divisions du système nerveux [5]

I.2.2 Cellules de base

Le système nerveux est formé de deux types de cellules de base : les cellules gliales et les neurones. Toutes nos sensations, nos mouvements, nos pensées et nos émotions sont le résultat de la communication entre les neurones, assurée par deux processus complémentaires : la conduction électrique et la transmission chimique. Les cellules gliales ne conduisent pas d'influx nerveux, elles n'en demeurent pas moins essentielles pour le bon fonctionnement des neurones [5].d'ou l'identification des structures du système nerveux à l'échelle microscopique est une étape qui conditionne sa compréhension dans son ensemble.

I.2.3 Le neurone

L'élément fonctionnel de base du système nerveux est le neurone. C'est une cellule composée d'un corps cellulaire (Somma) et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites, par lesquelles l'information sera acheminée de l'extérieur vers le somma, corps du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe.

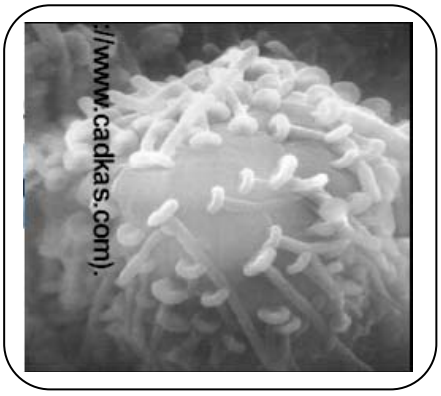


Figure I.2a Neurone biologique à l'échelle microscopique

En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms (10^{-9} m) entre l'axone du neurone émetteur et les dendrites du neurone récepteur [6]. La jonction entre deux neurones est appelée la synapse Figure I.2a_b.

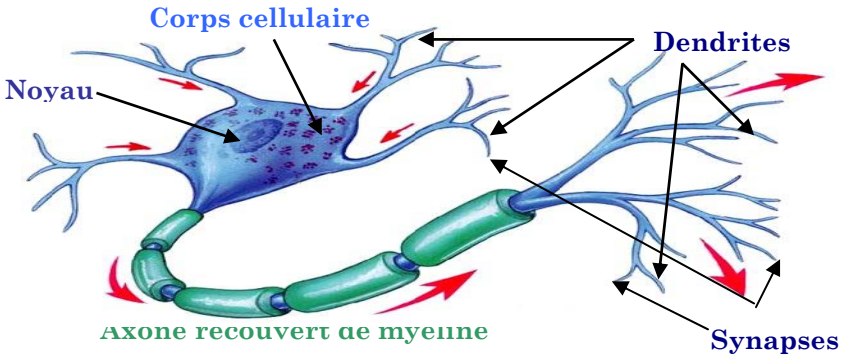


Figure I.2b Neurone biologique

Ces cellules sont caractérisées par l'influx nerveux, qui est assimilable à un signal électrique, se propageant dans les neurones de la manière suivante :

- ✓ Les dendrites reçoivent l'influx nerveux d'autres neurones.
- ✓ Le neurone évalue alors l'ensemble de la stimulation qu'il reçoit (c'est à dire

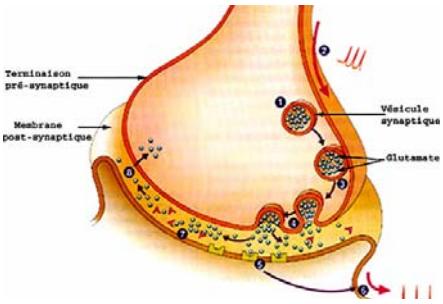


Figure I.2.c synapse

- ✓ En fonction de cette stimulation (si la dépolarisation est suffisante >-50mV par exemple), le neurone transmet ou non un signal du type «tout ou rien » le long de son axone. On dira alors que le neurone est excité ou non.
- ✓ L'excitation du neurone est propagée jusqu'aux autres neurones ou fibres musculaires qui sont connectés via les synapses Figure I.2.c.

I.2.4 Les cellules gliales

On entend peu parler des cellules gliales. Pourtant, elles sont 10 à 50 fois plus nombreuses que nos 100 milliards de neurones. Chose due au fait qu'elles ne conduisent pas l'influx nerveux comme les neurone. Mais cela ne les empêche pas pour autant d'être essentielles. A tel point que sans elles, les neurones ne fonctionneraient pas correctement : car les cellules gliales procurent aux neurones leur nourriture, les supportent et les protègent. Elles éliminent aussi les déchets causés par la mort neuronale et accélèrent la conduction nerveuse en agissant comme gaine isolante de certains axones comme les cellules gliales appelées Oligodendrocyte Figure I.3 [5] [7].

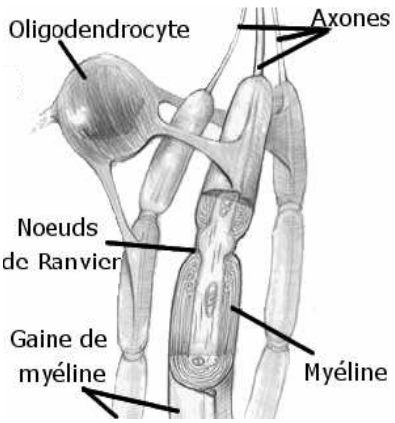


Figure I.3 Positionnement des oligodendrocytes vis-à-vis des neurones.

I.3 Système nerveux central

Il contient les centres nerveux et les voies neurologiques centrales motrices et sensitives. Il est formé principalement de la moelle épinière et du cerveau. Ces deux

structures sont encastrées dans des cavités osseuses et protégées par trois couches superposées de méninges.

- ***Le cerveau***

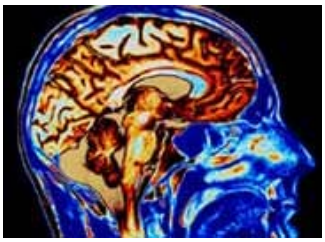


Figure I.4 Le cerveau

C'est la structure biologique la plus élaborée que nous connaissons. C'est la zone intégrative principale du système nerveux, l'endroit où les souvenirs sont stockés, les pensées conçues, les émotions générées et d'autres fonctions liées à notre psychisme ou aux commandes complexes de notre organisme réalisées.

Pour exécuter ces activités complexes, le cerveau lui-même est divisé en de nombreuses parties fonctionnelles.

- ***La moelle épinière***

Elle a deux fonctions. Premièrement, c'est un conducteur pour de nombreuses voies nerveuses allant vers le cerveau ou venant de ce dernier. Deuxièmement, c'est une zone intégrative pour coordonner de nombreuses activités nerveuses inconscientes.

I.4 Le système nerveux périphérique.

Il est constitué d'une multitude de fibres nerveuses organisées en faisceaux de nerfs qui permettent de mettre en relation l'individu et son environnement afin qu'il réagisse. Ces fibres sont de deux types fonctionnels [8]:

- ✓ ***Les fibres afférentes (les voies sensitives)*** : Qui transmettent les informations sensorielles vers la moelle épinière et le cerveau.

- ✓ ***Les fibres efférentes (les voies motrices)*** : Qui transmettent les signaux du système nerveux central à la périphérie, particulièrement vers les muscles squelettiques.

I.5 Mode d'action général du système nerveux

On peut décomposer le mode d'action du système nerveux en trois grandes étapes :

I.5.1 Détection des modifications survenant : Il serait évidemment impossible de contrôler la stabilité d'un système sans connaître, à tout moment, l'état des différentes composantes de ce système. La réception de l'information se fait par l'intermédiaire des récepteurs nerveux, des cellules nerveuses sensibles à certaines modifications de l'environnement. La Figure I.5 illustre le mode de fonctionnement du système nerveux [9].

I.5.2 Analyse d'informations et prise de décision : Toutes les informations perçues par les éléments sensoriels sont soigneusement analysées et comparées avec

d'autres informations mémorisées. Le résultat de cette analyse permettra au système nerveux de déterminer s'il doit ou non réagir aux modifications qu'il a enregistrées. Parfois l'analyse est réduite à sa plus simple expression. C'est le cas lors d'une réaction réflexe.

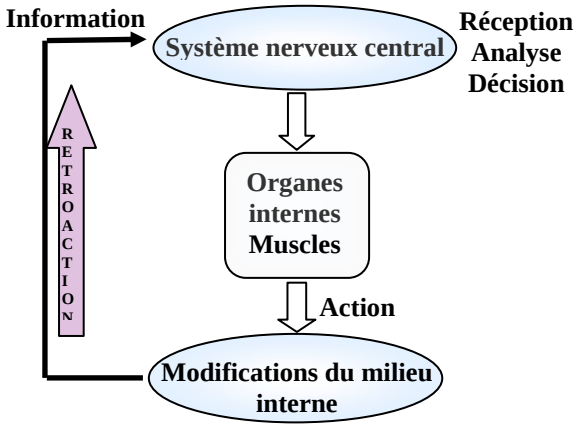


Figure I.5 Mode d'action du système nerveux

I.5.3 Action sur les organes internes ou sur les muscles : Une fois la décision prise, il faut agir. Le système nerveux peut agir sur les organes internes ainsi que sur les muscles. Cette action sur les muscles, c'est, bien sûr, ce que nous appelons le comportement.

De cette description du mode de fonctionnement du cerveau on peut noté que le fonctionnement du corps humain est basé sur les réactions de son système nerveux. Ce dernier dépend des activités des ses cellules de base (les neurones). Donc le disfonctionnement de ce système ne peut être détecté que par la compréhension du mode de fonctionnement de ses cellules. De ce fait, et après cette brève description du système nerveux nous allons expliquer le fonctionnement des cellules de base (l'activité électrique du cerveau) ainsi que les technique de surveillance de cette activité qui est l'électroencéphalographie.

I.6 Activité électrique du système nerveux (l'activité nerveuse)

Les neurones peuvent réagir à certaines modifications survenant dans leur milieu ambiant. Certains neurones réagissent à la présence de substances chimiques particulières entrant en contact avec leur membrane, d'autres peuvent réagir lorsque leur membrane subit une déformation mécanique, d'autres réagissent aux changements de température ou à la lumière. Le plus souvent, le neurone sensible à un de ces changements réagit en produisant un influx nerveux, c'est-à-dire un genre de courant électrique parcourant la membrane du neurone.

L'activité nerveuse s'explique par la circulation de ces influx nerveux dans de complexes réseaux de neurones interconnectés, qui est due aux Changements rapides de potentiels de part et d'autre de la membrane neuronale. Ces changements rapides de potentiels sont dus au passage transmembranaire de différents types d'ions. Les ions passent à travers la membrane au niveau de structures membranaires très particulières nommées les canaux ioniques. Ces canaux sont constitués de protéines. Ils ont trois propriétés importantes :

- ils conduisent les ions,
- ils sont spécifiques pour un ou plusieurs types d'ions,
- ils s'ouvrent ou se ferment en réponse à des signaux spécifiques (électriques ou chimiques).

Pour bien comprendre le fonctionnement du système nerveux, il faut absolument comprendre comment les influx nerveux sont produits et transmis dans un neurone (l'influx nerveux), et de neurone en neurone (transmission synaptique).

I.6.1 Propriétés de la membrane

La capacité d'un neurone à véhiculer les messages au sein du système provient des propriétés de sa membrane. De part et d'autre de celle ci se trouvent des charges positives et négatives, réparties de sorte à créer une différence de potentiel d'environ -80 millivolts, appelée potentiel transmembranaire. Cette différence de potentiel est maintenue au repos grâce aux ions des liquides intracellulaire et interstitiel.

I.6.1.1 Polarisation de la membrane du neurone :

C'est à la fin du XVIIIe siècle que l'Italien Luigi Galvani démontra qu'une forme d'électricité circulait dans les nerfs. Assez vite, on se rendit compte qu'il ne s'agissait pas d'un simple courant électrique comme celui qui circule dans les fils électriques en métal. Il faudra attendre les travaux de deux Anglais, Alan Hodgkin et Andrew Huxley, dans les années 50 (prix Nobel 1963) pour enfin comprendre la nature exacte du phénomène électrique responsable de la communication entre les neurones.

Une première expérience réalisée par Hodgkin et Huxley, elle consiste à placer les bornes d'un voltmètre de part et d'autre de la membrane d'un neurone isolé. Ils ont enregistré alors une différence de potentiel de 70 mV. L'intérieur de la membrane du neurone est chargé négativement par rapport à l'extérieur chargé positivement Figure I.6.

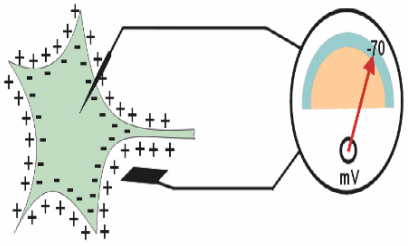


Figure I.6 Potentiel de repos -70 mV

Après cet expérience on constaté que les cellules possèdent une charge électrique car :

- La différence de concentration en ions de part et d'autre de la membrane Figure I.6.
- La perméabilité de la membrane à ces ions.
- La présence de « pompes », c'est à dire de transporteurs actifs dont le rôle est de maintenir cette différence de concentration en ions de part et d'autre de la membrane.

I.6.2 L'influx nerveux

L'influx nerveux résulte d'une variation transitoire, de l'ordre d'une milliseconde, de la répartition des ions situés de part et d'autre de la membrane cellulaire. Lorsque le potentiel d'action se développe sur la zone stimulée, le taux élevé d'ions sodium qui traverse la membrane influence la zone immédiatement voisine, qui devient aussi plus perméable aux ions sodium, et finit par développer également un potentiel d'action[13]. La répétition de ce phénomène de proche en proche conduit à une onde de dépolarisation ou influx-nerveux, qui se propage le long du neurone comme l'illustre Figure I.7.

I.6.3 La transmission synaptique

Au point de contact entre deux neurones (synapse), il existe un petit espace de quelques nanomètres, appelé fente synaptique, qui empêche le passage direct de l'influx nerveux.

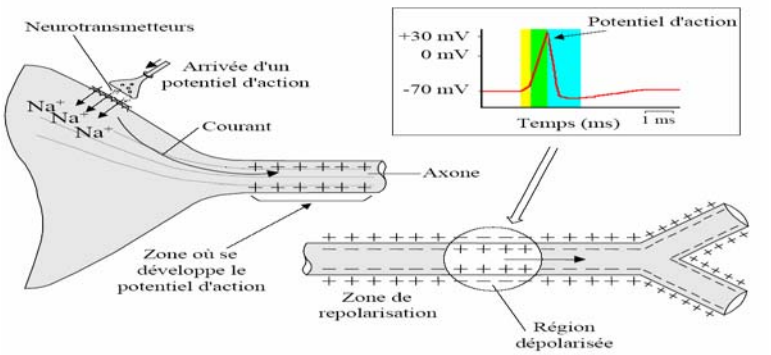


Figure I.7 L'influx nerveux

L'arrivée de ce dernier provoque plutôt la libération de substances chimiques emprisonnées dans des vésicules situées dans la région pré synaptique. Pour que le neurone post synaptique puisse développer un potentiel d'action à un instant donné, il faut que l'effet cumulé des substances chimiques reçues de tous les neurones pré synaptiques à cet instant force une dépolarisation.

I.7 L'électroencéphalographie

I.7.1 Définition

L'électroencéphalographie EEG est une technique de mesure de l'activité électrique du cerveau qui est provoqué par les courants qui se produisent dans les neurones [10]. L'histoire de l'électroencéphalographie commence avec le biologiste Richard Caton qui a détecté en 1875 « la présence de courants électriques attestée par des oscillations du galvanomètre » chez le singe et le lapin. Hans Berger a appliqué à l'homme cette technique et a enregistré le premier EEG en 1929 sous forme de variations permanentes de potentiel enregistrées avec des électrodes impolarisables appliquées sur une lacune crânienne ou à la surface du crâne intact. L'inscription à jet d'encre, introduite par Grass en 1935, a permet de visualiser les activités électriques sur papier. Les bases de l'examen ont été posées dès 1945 et sont toujours appliquées aujourd'hui. Depuis quelques années, avec l'avènement des

micros ordinateurs, l'enregistrement papier est de plus en plus souvent remplacé par l'enregistrement numérique [11].

Figure I.10 l'enregistrement papier et l'enregistrement numérique.

I.7.2 L'électroencéphalogramme

L'électroencéphalogramme (EEG) est le tracé que l'on obtient par l'enregistrement de l'ensemble des variations de potentiel du cortex cérébral sur toute la boîte crânienne à travers le cuir chevelu. En effet, Les neurones reçoivent incessamment des impulsions des autres neurones. L'EEG est le résultat de la sommation des potentiels dérivés du mélange des courants extra cellulaires générés par des neurones [2].

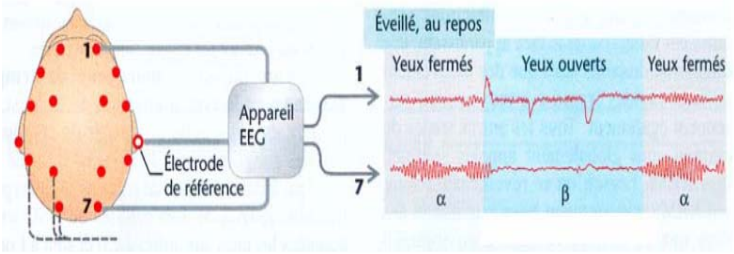


Figure I.11 L'électroencéphalogramme (EEG)

Le principe de l'EEG est de recueillir les potentiels électriques sur un appareil qui amplifie les signaux, puis les transcrit pour qu'ils puissent être analysés (E.E.G.). L'électroencéphalogramme représente la transcription sous forme d'un tracé des variations dans le temps des potentiels électriques recueillis sur la boîte crânienne en différents points du scalp.

On réserve le nom d'électrocardiographie au tracé obtenu lorsque les électrodes sont placées directement sur le cortex et l'on parle de stéréo électroencéphalographie (Figure I.12) lorsque les électrodes sont implantées directement dans les structures cérébrales (lors d'interventions neurochirurgicales), le signal recueilli est bien différent.

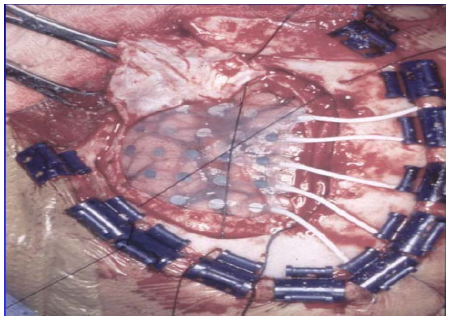


Figure I.12 stéréo électroencéphalographie.

I.7.3 Caractéristiques du signal EEG et rythmes E.E.G

Le signal EEG de l'être humain est un signal très faible, il varis dans un intervalle de 0.5 à 100 µV d'amplitude, et dans une gamme de fréquence qui s'étale entre 0 et 40 Hz. Les variations de ces signaux dans cette gamme de fréquence, sont appelées les rythmes cérébraux. Les caractéristiques des rythmes cérébraux dépendent de l'état psychologique et pathologique de la personne chez qui on les enregistre. Ainsi, l'enregistrement de l'activité rythmique cérébrale permet d'étudier les phases du sommeil ou de caractériser des maladies neurologiques, telles que l'épilepsie

I.7.3.1 Le rythme alpha (α) :

Ce rythme est caractérisé par sa fréquence comprise entre 8 et 12 Hz, et son amplitude de 20 à 50 µv. il a un aspect d'ondes sinusoïdales régulières formant le plus souvent des fuseaux, en général synchrones à droite et à gauche [14]. Son signal est caractérisé par une symétrie en fréquence, mais l'amplitude et la longueur des fuseaux varient souvent suivant la dominance hémisphérique.

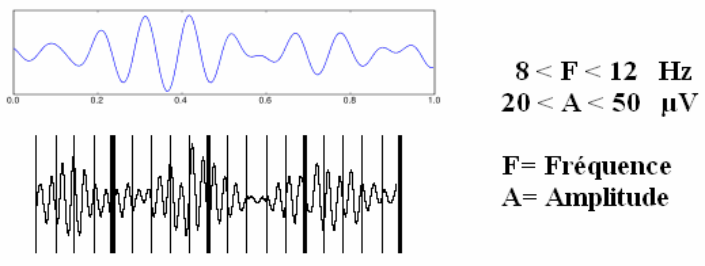


Figure I.13 Le rythme alpha (α)

Ce rythme est dit "de repos"; il survient quand le sujet est allongé, les yeux fermés à l'abri de toute stimulation sensorielle, sans toutefois s'endormir. Ce rythme est labile : il disparaît à l'ouverture des yeux, caractéristique appelée « réaction d'arrêt ». On l'enregistre aussi lors d'un effort d'attention (calcul mental) ou d'une réaction émotive. Ce rythme est observé dès l'âge de 7-8 ans. Il devient prédominant vers 15 ans.

I.7.3.2 Le rythme bêta (β) :

Ce rythme est caractérisé par sa fréquence comprise entre 13 et 25 Hz, et son amplitude de 5 à 10 µv. il a un aspect d'ondes sinusoïdales très peu amples, irrégulières, difficiles à visualiser, et surtout provoquées par l'EMG [14].

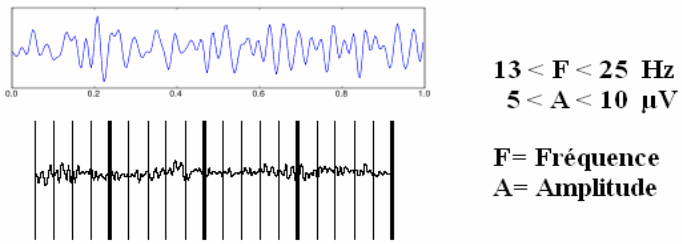


Figure I.14 Le rythme bêta (β)

Ce rythme est dit "de l'adulte" au repos et éveillé, il peut être bloqué lors de l'exécution volontaire d'un mouvement. Il apparaît vers l'âge de 15 ans, à l'adolescence.

I.7.3.3. Le rythme thêta (θ) :

Ce rythme est caractérisé par sa fréquence comprise entre 4 et 7 Hz, et son amplitude de 50 µv. il a un aspect d'ondes sinusoïdales assez ample, il survient en général par bouffées fusiformes, brèves et bilatérales.

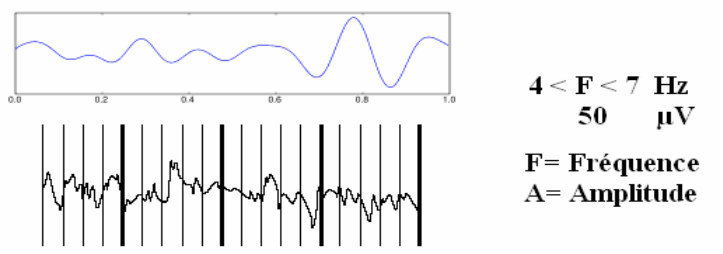


Figure I.15 Le rythme thêta (θ)

Ce rythme est normalement présent, mais peu abondant, il est souvent masqué par le rythme alpha. Abondant chez l'enfant.

I.7.3.4 Le rythme delta (δ) :

Ce rythme est caractérisé par une fréquence inférieure à 4 Hz une amplitude grande mais très variable. On distingue classiquement les ondes delta monomorphes lentes et régulières, et les ondes delta polymorphes plus irrégulières, plus lentes et moins amples.

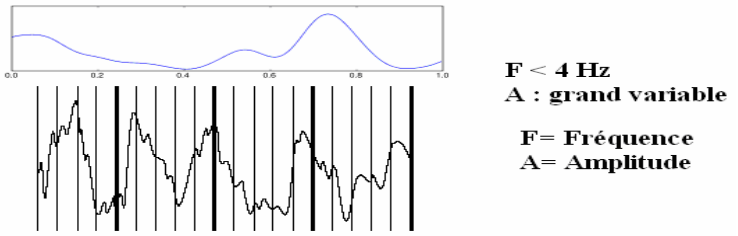


Figure I.16 Le rythme delta (δ)

Ce rythme n'est jamais présent chez l'adulte éveillé au repos, mais on le rencontre pendant le sommeil lent et profond. Le Tableau I.1 résume les caractéristiques du signal EEG.

Tableau I.1 les caractéristiques du signal EEG

Type de signal	*Gamme de fréquence (Hz)*	*Gamme d'amplitude (µV)*
Alpha (α)	8 à 12	20 à 50
Bêta (ß)	13 à 25	5 à 10
Thêta (θ)	4 à 7	50
Delta (δ)	< 4	< 100

I.8 Technique de d'enregistrement EEG

Elle consiste à mesurer des différences de potentiel entre électrodes disposées à la surface de la tète, le contact électrique étant assuré par un gel conducteur. Le nombre d'électrodes utilisées peut être très variable, allant de 20 électrodes dans le montage international 10-20 à des nombres plus importants 64, 128 voire 256 (Figure I.17). Dans ce dernier cas, les électrodes sont disposées dans un bonnet électrique posé sur la tête du patient, alors que pour des enregistrements de longue durée, sur plusieurs jours comme c'est le cas en épilepsie, les électrodes sont collées directement sur le scalp avec une pâte conductrice. Etant donné qu'on ne peut pas mesurer de potentiel absolu, le choix d'une électrode de référence est indispensable. Quand la tension est mesurée entre deux électrodes successives, le montage est dit bipolaire. Le plus souvent, une seule électrode est utilisée comme référence pour l'ensemble des autres.

Figure I.17 Disposition des électrodes

Cette technique d'enregistrement de l'activité électrique cérébrale nécessite l'utilisation du matérielle suivant [12] :

- Les électrodes : c'est l'élément clef du système d'enregistrement. Leur choix et leur fonctionnement approprié sont une nécessité absolue pour la réussite de l'opération d'enregistrement.
- Amplificateur et filtre : La relative faiblesse des tensions enregistré, de l'ordre du microvolt (µV), et leur basse fréquence comprises entre 0,5 à 40 (comprises entre 0,5 à 80 voire 100 Hertz environ mais, dans la pratique courante, on se contente d'enregistrer les fréquences comprises entre 0,5 et 40 Hz), nécessitent d'une part un dispositif d'amplification sélective ayant un gain élevé (il est exprimé en dB), à le quel correspondait un système d'inscription, qui transmet à une plume les variations de potentiel qu'il reçoit, de sorte que celles-ci se trouvent traduites sur le papier d'enregistrement en déflexions de la plume proportionnelles au voltage. Et d'autre part, l'utilisation des filtres passe bas et passe haut.
- Convertisseur analogique numérique : on l'utilisent comme interface avec l'ordinateur lors des opérations d'enregistrement ou de stockage numérique.
- Système d'enregistrement (ou de stockage) : sous forme d'appareille avec un certain nombre de plumes servant à l'inscription des signaux EEG sur des rouleaux de papier (maintenant il sont remplacé par les ordinateurs).

I.8.1 les artéfacts

A travers l'analyse du tracé EEG il apparaît quelques distorsions du signal qu'on appelle artéfacts, qui sont une activité électrique enregistrée n'ayant pas pour origine l'activité cérébrale. Habituellement, ils un grand amplitude et une forme différente de celle d'un tracé non contaminé. Les artéfacts dans un EEG peuvent avoir deux origines :

- Philosophiques : c'est des artéfacts reliés au patient lui-même, le mouvement de la tête, des yeux et des paupières, son état psychique…etc.
- Techniques : ce type d'artéfacts est dû mouvement des électrodes, aux files de connexion des électrodes avec le système d'enregistrement, au secteur (artefacts à 50/60 Hz ou terre non raccordée), mauvais contact des électrodes….etc.

Si on examine le système d'enregistrement de la Figure I.17, on constate que ce système n'est pas approprié pour une bonne mesure du EEG (On parle de EEG bien précis avec un signal de bonne caractéristiques), car il est source d'un grand nombre d'artéfacts (nombre et type d'électrodes, langueur des fils, interférence, position du patient….etc.).

I.8.2 Applications de l'enregistrement EEG

Le plus grand avantage de l'enregistrement EEG est ça vitesse qui nous permet d'enregistrer des modèles complexes de l'activité neuronale issue d'un stimulus dans une fraction de seconde sous forme de tracés.

Selon R. Bickford la recherche et les applications cliniques de l'EEG chez l'homme et l'animal sont employées pour [15]:

1) Surveillez la vigilance, le coma et la mort du cerveau.
2) Localisez les secteurs endommagés dans le cerveau (la tumeur).
3) Examinez les voies afférentes (par des potentiels évoqués).
4) Surveillez l'enclenchement cognitif (rythme d'alpha).
5) Produire la situations de rétroaction biologique, alpha, etc.
6) Commandez la profondeur d'anesthésie ("anesthésie cerveau").
7) Etudiez l'épilepsie et localisez l'origine de saisie.
8) Tester les effets de la drogue sur l'épilepsie.
9) Aidez à l'excision corticale expérimentale du foyer épileptique.
10) Surveillez le développement du cerveau humain et animal.
11) Tester les effets convulsifs de la drogue.
12) Etudier la physiologie et les troubles du sommeil.

I.9 Conclusion

Ce chapitre nous a permis de dégager les notions de physiologie et d'anatomie nécessaires à la compréhension du mode de fonctionnement du système nerveux et de son activité électrique en général.

Les grandes divisions du système nerveux ont été présentées, ainsi que l'activité électrique des neurones. Nous avons également tenu à présenter l'électroencéphalographie en tant que technique d'imagerie fonctionnelle. De même, nous avons présenté les caractéristiques du signal EEG ainsi que ces formes d'ondes utilisées et ses applications. En plus, nous avons présenté un système d'enregistrement et ses limitations. Une alternative à ce système fera l'objet de notre travail dans les chapitres prochains.

CHAPITRE II

LES RESEAUX DE NEURONES ARTIFICIELS

II.1 Introduction [17] [18]

Les réseaux de neurones, fabriquées de structures cellulaires artificielles, constituent une approche permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement. Ils s'avèrent aussi des alternatives très prometteuses pour contourner certaines des limitations des ordinateurs classiques. Grâce a leur traitement parallèle de l'information et `a leurs mécanismes inspirés des cellules nerveuses (neurones), ils infèrent des propriétés émergentes permettant de solutionner des problèmes jadis qualifies de complexes.

L'histoire des réseaux de neurones date de ***1943*** avec les travaux de ***McCulloch*** et ***Pitts.*** Qui montraient qu'un réseau de neurones discret sans contrainte de topologie, peut représenter n'importe quelle fonction booléenne et donc émule un ordinateur. En ***1958***, ***Rosenblatt*** propose le premier algorithme d'apprentissage, qui permet d'ajuster les paramètres d'un neurone. En ***1969***, ***Minsky*** et ***Papert*** publient le livre Perceptrons dans lequel ils utilisent une solide argumentation mathématique pour démontrer les limitations des réseaux de neurones à une seule couche. Ce livre aura une influence telle que la plupart des chercheurs quitteront le champ de recherche sur les réseaux de neurones. En ***1982***, ***Hopfield*** propose des réseaux de neurones associatifs et l'intérêt pour les réseaux de neurones renaît chez les scientifiques. En ***1986***, ***Rumelhart***, ***Hinton*** et ***Williams*** publient, l'algorithme de la Rétro-propagation de l'erreur (RPG) qui permet d'optimiser les paramètres d'un réseau de neurones à plusieurs couches. À partir de ce moment, la recherche sur les réseaux de neurones connaît un essor fulgurant et les applications commerciales de ce succès académique suivent au cours des années ***90*** et jusqu'à nos jours.

Aujourd'hui, on retrouve les réseaux de neurones solidement implantés dans diverses industries : dans les milieux financiers pour la prédiction des fluctuations de marché [2], dans les applications biomédicales et biopharmacie, dans le domaine bancaire pour la détection de fraudes sur les cartes de crédit et le calcul de cotes de crédit ; dans les départements de marketing de compagnies de diverses industries pour prévoir le comportement des consommateurs ; en statistique ;en traitement du signal et communications ; en aéronautique pour la programmation de pilotes automatiques, dans le pilotage automatique des voiture moderne, en robotique, et presque dans tous les application que l'homme confronte quotidiennement. Ce qui fait que ces applications sont nombreuses et partagent toutes un point commun essentiel à l'utilité des réseaux de neurones : les processus pour lesquels on désire émettre des prédictions comportent de nombreuses variables explicatives et surtout, il existe possiblement des dépendances non linéaires de haut niveau entre ces variables qui, si elles sont découvertes et exploitées, peuvent servir à l'amélioration de la prédiction du processus.

Dans ce chapitre nous allons présenter d'une manière générale les réseaux de neurones on commençons par la définition des concepts de base et les constructions des réseaux, par la suit nous décrivons en générale les différents types et règles employer dans les algorithmes d'apprentissage, a la fin nous présentons l'implémentation des réseaux de neurones et les domaines d'applications arrivons a la conclusion de ce chapitra.

II.2 Définition des réseaux de neurones :

On peut considérer qu'un réseau neuronal reflète sensiblement le comportement de la structure de neurones organiques présente au sein d'un cerveau humain, les messages étant véhiculés à travers les différents nœuds interconnectés et ce, jusqu'à la sortie. Le réseau peut être agencé de multiples manières, voire avec des branches reliant la sortie de certains nœuds à d'autres situés en amont (opérant ainsi une boucle de retour des informations) et modéliser de ce fait des fonctions très complexes.

Selon CLAUDE TOUZET [6] « ***Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau*** ».

Et d'après Kohonen [19] « ***Les réseaux de neurones artificiels sont des réseaux massivement connectés en parallèle d'éléments simples (habituellement adaptatifs) et leur organisation hiérarchique. Ils sont censés interagir avec les objets du monde réel de la même manière que les systèmes nerveux biologiques*** ».

Dans toutes définitions d'un réseaux de neurones, l'élément de base a mettre on évidence est le neurone ce dernier, comme une pièce de monnaie, possède deux faces qui l'identifient, celle du neurone biologique présentée dans le chapitre précédant et celle du neurone formel ou mathématique.

II.2.1 Le neurone mathématique (Formel) :

Mathématiquement un neurone artificiel est un processeur élémentaire essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées dont chacune on associé un poids représentatif de la force de la connexion. Ces entrées sont en provenance de neurones

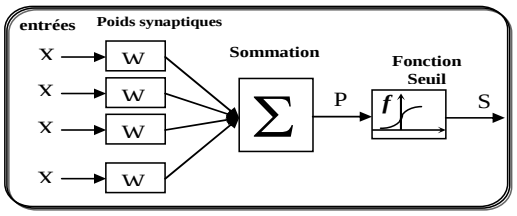


Figure II.1. Modèle mathématique du neurone

appartenant à un niveau situé en amont. Le résultat de cette somme est ensuite transformé par une fonction de transfert qui produit la sortie à un ou plusieurs neurones appartenant à un niveau situé en aval. Cette présentation décrit un comportement en deux phases qui sont :

1. le calcule de la somme pondérée P selon l'équation :

$$P=\Sigma (w_n . x_n) \qquad \text{(II-1)}$$

2. a partir de cette valeur, une fonction de transfert calcule la valeur de l'état du neurone.

$$S = f (p) \qquad \text{(II-2)}$$

Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur le **Tableau II.1** On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de

transfert sont continues, offrant une infinité de valeurs possibles comprises dans l'intervalle [0 , +1] ou [-1 , +1] .

Nom de la fonction	Relation d'entrée/sortie	Icône
seuil	$a = 0 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$	
seuil symétrique	$a = -1 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$	
linéaire	$a = n$	
linéaire saturée	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } 0 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$	
linéaire saturée symétrique	$a = -1 \quad \text{si } n < -1$ $a = n \quad \text{si } -1 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$	
linéaire positive	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } n \geq 0$	
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$	
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$	
compétitive	$a = 1 \quad \text{si } n \text{ maximum}$ $a = 0 \quad \text{autrement}$	C

Tableau II.1 Fonctions de transfert les plus courantes

Les réseaux de neurones se distinguent par leur mode d'apprentissage et par leur topologie (architecture).ces derniers qui diffère eux aussi d'une application à une autres.

II.3 Architecture des réseaux de neurones [20] [21]

Du point de vue architectural les réseaux de neurones peuvent être regroupés en deux catégories :

- Les «***Feed-forward networks***» : Ce sont des réseaux à circulation de l'information vers l'avant et dans lesquels l'organisation des neurones est en couches successives. Le calcul se fait en propageant les données de l'entrée vers la sortie. Dans cette catégorie on distingue les réseaux à une seule couche et les réseaux multicouches (possédant une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées) Figure II.2.

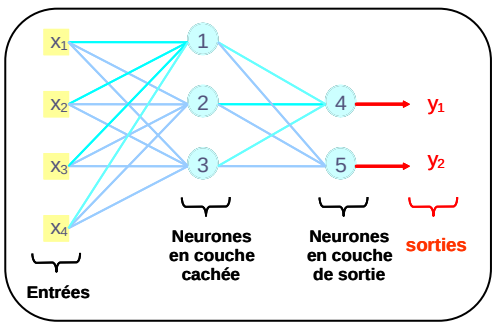


Figure II.2 Exemple de réseau non bouclé

- Les «***Feed-back networks***» ou récurrents :

Il présentent au moins une boucle de rétroaction, se sont des modèles plus difficiles à mettre en œuvre (problème de convergence). Dans cette catégorie on distingue les : réseaux compétitifs, le réseau de Kohonen, les réseaux de Hopfield et les modèles ART (Théorie de la Résonance Artificielle). Dans notre mémoire nous nous intéressons au réseau Feed-forward multicouche Figure II.3.

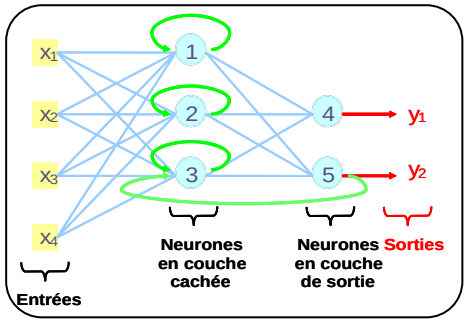


Figure II.2 exemple de réseau bouclé

Dans la Figure II.4 Résume quelques architectures appartenant aux deux catégories d'architectures fondamentales précédemment expliquées.

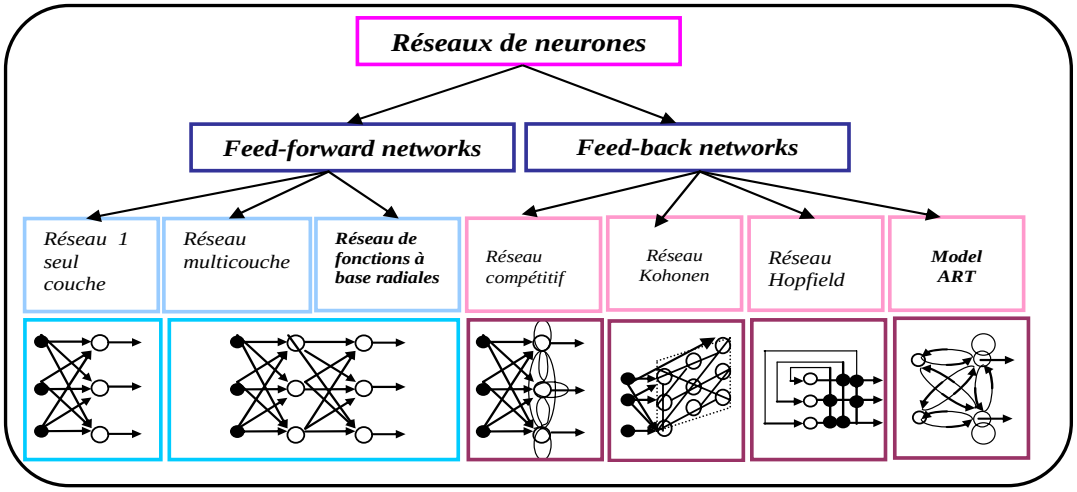


Figure II.4 Différentes architectures des réseaux de neurones

II.4 Les algorithmes d'apprentissages [22]

II.4.1 Définition :

L'apprentissage est une phase de développement d'un réseau de neurone durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. Pour cela il nécessite :

- ✓ Un ensemble d'exemples d'apprentissages : en effet les réseaux de neurones sont des fonctions paramétrées, utilisées pour réaliser des modèles statistiques à partir d'exemples (dans le cas de la classification) ou de mesures (dans le cas de la modélisation) ; leurs paramètres sont calculés à partir de ces exemples ou couples {entrée, sortie}.
- ✓ La définition d'une fonction de coût qui mesure l'écart entre les sorties du réseau de neurone et les sorties désirées.
- ✓ Un algorithme de minimisation de la fonction de coût par rapport aux paramètres.

II.4.2 Types d'apprentissage :

Dans les réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage. Le modèle sans apprentissage présente en effet peu d'intérêt. 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 dont l'optique d'accorder la réponse du réseau aux exemples et à l'expérience. Il est souvent impossible de décider a priori des valeurs des poids des connexions d'un réseau pour une application donnée. A l'issue de l'apprentissage, les poids sont fixés : c'est alors la phase d'utilisation. Certains modèles de réseaux sont improprement dénommés à apprentissage permanent. Dans ce cas il est vrai que l'apprentissage ne s'arrête jamais, cependant on peut toujours distinguer une phase d'apprentissage (en fait de remise à jour du comportement) et une phase d'utilisation. Cette technique permet de conserver au réseau un comportement adapté malgré les fluctuations dans les données d'entrées.

Il existe trois types d'apprentissage :

II.4.2.1 Apprentissage supervisé :

Dans l'apprentissage supervisé, un professeur qui connaît parfaitement la sortie désirée ou correcte, guide le réseau en lui apprenant à chaque étape le bon résultat. Donc l'apprentissage ici, consiste à comparer le résultat obtenu avec le résultat désiré, puis à ajuster les poids des connexions pour minimiser la différence entre les deux comme l'indique la Figure II.5

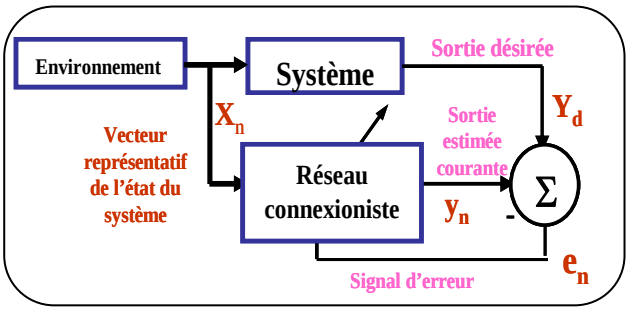


Figure II.5 Principe de l'apprentissage supervisé

Dans cette figure le Système à apprendre constitue un professeur pour le réseau, le Réseau connexioniste prend comme stimulus le même vecteur des variables explicatives que le Système.

Ce système, en réponse à l'entrée présentée, fournit une Sortie désirée qui représente pour le réseau le comportement de référence. Parmi les algorithmes à apprentissage supervisé, on reconnaît dans la littérature l'algorithme à Rétro-propagation du gradient.

II.4.2.2 Apprentissage renforcé :

C'est un apprentissage qui élabore et assimile une critique à l'issue de chaque exemple traité par la machine d'apprentissage. En effet l'algorithme de critique utilise des indications imprécises sur le comportement final «échec ou succès»du réseau à travers le vecteur des variables explicatives de la Figure II.6. L'ajustement des poids du réseau se fait uniquement à partir d'une simple évaluation de la qualité de sa réponse.

Un apprentissage par renforcement est nécessaire pour la plupart des réseaux en interaction avec l'environnement. Ces interactions

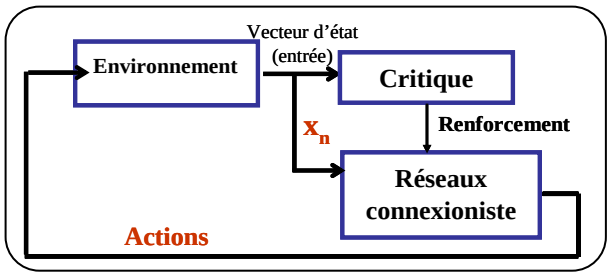


Figure II.6 Principe de l'apprentissage renforcé

du comportement désiré mais uniquement une évaluation du comportement (une récompense ou une pénalité). Le réseau doit alors découvrir les réponses qui lui apportent le maximum de récompenses.

II.4.2.3 Apprentissage non supervisé :

L'apprentissage supervisé s'effectue sous le contrôle d'un expert, alors que l'apprentissage non supervisé est autodidacte. Les paramètres internes du réseau ne sont modifiés qu'avec les seuls stimuli, aucune réponse désirée n'est prise en considération. La Figure II.7 montre que la sortie du réseau n'est pas utilisée par la procédure d'apprentissage.

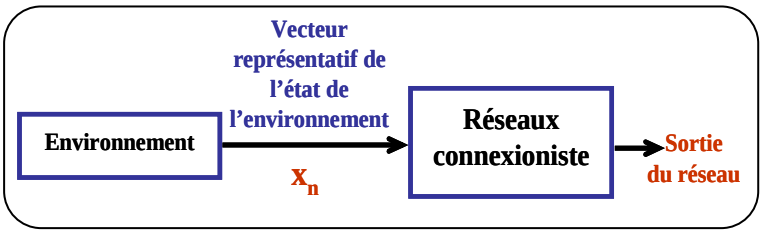


Figure II.7 principe de l'apprentissage non supervisé

Par nature, ce type d'apprentissage construit une représentation interne de la connaissance issue de l'environnement. Cet apprentissage est basé sur une mesure de la qualité de la représentation de la connaissance pour ajuster en conséquence les paramètres internes du réseau de neurone, un critère interne souvent utilisé pour modifier les poids des neurones.

Il faut souligner que l'efficacité de l'apprentissage augmente avec le nombre d'exemples d'apprentissage. Ceci est un facteur très important qu'il fallait noter : quelque soit l'algorithme choisi, la qualité de l'apprentissage des réseaux de neurone est d'autant meilleur que l'on dispose d'un ensemble d'apprentissage riche en exemples. On sait par ailleurs que la capacité de généralisation des réseaux de neurone nécessite également des exemples nombreux, qui de plus, doivent être bien distribués dans le domaine de validité souhaité par le modèle.

II.4.3 Règles d'apprentissage [24]

La méthode d'ajustement des poids synaptiques pendant l'apprentissage du réseau peut être choisi dans une gamme variée de règles d'apprentissage dont les plus connues sont citées ci- dessous :

II.4.3.1 la règle de Hebb :

L'apprentissage de Hebb est le premier mécanisme d'évaluation des synapses, proposée par D.O Hebb en 1949. Il s'applique aux connexions entre neurones, La règle de Hebb s'exprime de la façon suivante :

1. « ***Si deux neurones de part et d'autre d'une synapse (connexion) sont activés simultanément (d'une manière synchrone), alors la force de cette synapse doit être augmentée*** ».
2. « ***Si les mêmes deux neurones sont activés d'une manière asynchrone, alors la synapse correspondante doit être affaiblie ou carrément éliminée*** ».

La modification de poids dépend de la coactivation des neurones pré-synaptiques et post- synaptique. La loi de Hebb peut être modélisée par les équations suivantes :

$$W_{ij}(t+1) = W_{ij}(t) + \eta S_i S_j \qquad \text{(II.3)}$$

Où,

W_{ij} (t) est le poids de la connexion reliant les neurones S_i et S_j à l'étape t.

W_{ij} (t+1) est le nouveau poids à l'étape t+1.

η : est un nombre compris entre 0 et 1, représentant le taux d'apprentissage.

t : représente l'étape d'apprentissage.

L'algorithme d'apprentissage modifie de façon itérative les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement.

1. Initialisation des poids et du seuil « s » à de faibles valeurs aléatoires.

2. Présentation d'une entrée X= (x_1,... x_n) de la base d'apprentissage.

3. Calcul de la sortie obtenue S pour cette entrée.

4. Si la sortie S est différente de la sortie désirée pour cet exemple d'entrée alors on modifie les poids synaptiques selon l'équation (II-3).

5. Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement, retour à l'étape 2.

II.4.3.2 Règle du perceptron [25] [26]

Dans le cas du Perceptron, la fonction d'activation est une fonction discrète. Les sorties prennent des valeurs binaires (0 ou 1). La règle d'apprentissage du Perceptron est la suivante :

$$\begin{cases} W_{ij}(t+1) = W_{ij}(t) + \eta x_i & \text{Si la sortie actuelle est égale à 0 et doit être égale à 1.} \\ W_{ij}(t+1) = W_{ij}(t) - \eta x_i & \text{Si la sortie actuelle est égale à 1 et doit être égale à 0.} \\ W_{ij}(t+1) = W_{ij}(t) & \text{Si la sortie est correcte.} \end{cases}$$

Où,

$\eta > 0$: représente le coefficient d'apprentissage.

t : l'étape d'apprentissage.

x_i : l'entrée du neurone i.

W_{ij} : le poids synaptique ou connexion entre neurone i et le neurone j.

L'algorithme d'apprentissage du Perceptron est semblable à celui utilisé pour la loi de Hebb. Les différences se situent au niveau de la modification des poids. L'inconvénient majeur du Perceptron est qu'il ne s'applique que si les classes sont linéairement séparables.

II.4.3.3 la règle de Windrow-Hoff (la règle delta) :

Nous avons vu, que le Perceptron est limité à des sorties binaires, Widrow et Hoff ont étudié l'algorithme d'apprentissage du Perceptron en considérant une fonction d'activation continue et différentielle. Elle est aussi connue sous le nom de la méthode des moindres carrées ou **LMS** (**L**east **M**ean **S**quare). Le principe est le suivant :

- ✓ Calculer l'erreur quadratique selon la formule :

$$E = \sum_{j=1}^{n} (d_j - y_j)^2 \qquad \text{(II-4)}$$

$$Yi = \sum_{i=1}^{m} x_i W_{ji} \qquad \text{(II-5)}$$

- ✓ Minimiser cette erreur en modifiant les poids de chaque neurone suivant la règle :

$$W_{ji}(t+1) = W_{ji}(t) + \eta x_i (d_j - y_j) \qquad \text{(II-6)}$$

Où,

n : nombre de neurones à la sortie
d_i : est la sortie désirée
y_j : est la sortie calculée
x_i : l'entrée i du neurone j
m : le nombre de neurones à l'entrée
η : Coefficient d'apprentissage.

La règle de Widrow-Hoff pose le problème de l'apprentissage comme un problème de minimisation de fonction (erreur) globale.

II.4.3.4 la règle delta généralisé :

La règle delta généralisée ou encore appelée règle de Rétro-propagation du gradient est une généralisation de la règle de Widrow-Hoff et s'applique à un réseau multicouche utilisant des fonctions d'activation différentiables et un apprentissage supervisé. Nous verrons en détail cette règle au prochain chapitre.

II.5 les avantages et les inconvenants des réseaux de neurone :[27]

II.5.1 Avantages :

- ✓ Implémentation du parallélisme.
- ✓ Apprentissage.
- ✓ Robustesse : données bruites ou incomplètes.
- ✓ Généralisation à des Modèles similaire.
- ✓ Trouve des solutions aux problèmes non linéaires.
- ✓ Trouve des solutions aux problèmes qui n'ont pas une modélisation.

II.5.2 Inconvenants :

- ✓ N'ont pas encore expliqué le fonctionnement du cerveau.
- ✓ Les poids ne sont pas interprétables.
- ✓ L'apprentissage n'est pas toujours évident.
- ✓

II.6 les domaines d'application des réseaux de neurones :[19][28]

II.6.1 la régression non linéaire ou modélisation des données statistiques

Il existe une immense variété de phénomènes statiques qui peuvent être caractérisés par une relation déterministe entre des causes et des effets ; les réseaux de neurones sont de bons candidats pour modéliser de telles relations à partir d'observations expérimentales, sous réserve que celles-ci soient suffisamment nombreuses et représentatives.

II.6.2 la modélisation de processus dynamiques non linéaires :

Modéliser un processus, c'est trouver un ensemble d'équations mathématiques qui décrivent le comportement dynamique du processus, c'est-à-dire l'évolution de ses sorties en fonction des entrées ; c'est donc typiquement un problème qui peut être avantageusement résolu par un réseau de neurones, si le phénomène que l'on désire modéliser est non linéaire. La prédiction de séries chronologiques (prédictions financières, prédiction de consommation, etc.) entre dans ce cadre.

II.6.3 la commande de processus :

Commander un processus, c'est imposer à celui ci un comportement défini à l'avance en fonction des signaux de commande, l'ensemble (commande/processus) peut donc être considéré comme un système qui réalise une fonction (non linéaire) qu'un réseau de neurones peut approcher.

II.6.4 la classification :

Supposons que l'on désire classer des formes en deux catégories, A ou B, en fonction de certaines caractéristiques de ces formes ; on peut définir une fonction qui vaut (+1) pour toutes les formes de la classe A et (-1) pour toutes les formes de la classe B. Les réseaux de neurones sont de bons candidats pour réaliser une approximation de cette fonction, et l'on peut démontrer que cette approximation constitue une estimation de la probabilité d'appartenance de la forme inconnue à la classe A. Les réseaux de neurones fournissent donc une information très riche, qui est loin d'être une simple réponse binaire. Cette propriété remarquable (que les réseaux de neurones partagent avec d'autres classificateurs) n'est malheureusement pas mise à profit dans la plupart des applications.

Pour utiliser les réseaux de neurones dans l'une des ces applications il faut suivre certaines procédures :

- ✓ Il faut tout d'abord choisir l'architecture du réseau, c'est-à-dire les entrées externes, le nombre de neurones cachés, et l'agencement des neurones entre eux, de telle manière que le réseau soit en mesure de reproduire ce qui est déterminé dans les données, le nombre de poids ajustables est un des facteurs fondamentaux de la réussite d'une application : si le réseau possède un trop grand nombre de poids, c'est-à-dire si le réseau est trop "souple", il risque de s'ajuster au bruit qui est présent dans les données de l'ensemble d'apprentissage, et, même en l'absence de bruit, il risque de présenter des oscillations non significatives entre les points d'apprentissage, donc de posséder de mauvaises propriétés d'interpolation ou "généralisation", si ce nombre est trop petit, le réseau est trop "rigide" et ne peut reproduire la partie déterministe de la fonction. Le problème de la détermination de l'architecture optimale est resté pendant longtemps un problème ouvert, mais il existe actuellement diverses méthodes, mettant notamment en jeu des tests statistiques, qui permettent de déterminer cette architecture pour une vaste classe de réseaux.

- ✓ Il faut calculer les poids du réseau ou, en d'autres termes, estimer les paramètres du réseau à partir des exemples, en minimisant l'erreur d'approximation sur les points de l'ensemble d'apprentissage, de telle manière que le réseau réalise la tâche désirée. Ce calcul des coefficients synaptiques constitue l'apprentissage supervisé pour le réseau de neurones.

- ✓ Il faut enfin estimer la qualité du réseau obtenu en lui présentant des exemples qui ne font pas partie de l'ensemble d'apprentissage.

En plus de toutes ces application, les réseaux de neurones artificiels sont aussi employer dans la reconnaissance des forme dans le traitement d'image, l'identification, la prédiction, le filtrage, traitement du signale, la robotique et dans plusieurs disciplines et domaines que nous rencontrons quotidiennement dans notre vie.

II.7 Implémentation des réseaux de neurone : [18][24][26]

Selon les performances requises de l'application, les réseaux de neurones peuvent être implémentés en software sur un ordinateur conventionnel, à l'aide de processeurs DSPs (Digital Signal Processor) programmables (circuits à application générale), en VLSI (Very Large Scale Integration) sur du silicium (circuits dédiés en full custom, cellules standards, FPGA), en optique ou en combinant ces techniques.

L'implémentation VLSI des réseaux de neurones est justifiée en premier lieu par la rapidité de traitement qui peut être atteinte, en utilisant des architectures massivement parallèles, vient ensuite la possibilité de réaliser des systèmes portables. On distingue deux grandes approches d'implémentation VLSI des réseaux de neurones [16] :

- ✓ Les implémentations qui intègrent la phase d'apprentissage et la phase de test/généralisation dans un même circuit d'où le terme anglais "***on-chip training circuits***". Ce genre d'implémentation permet une flexibilité et une adaptabilité du circuit à plusieurs applications, néanmoins ces circuits ne sont pas très performants (complexité d'interconnexion et rapidité de traitement).

- ✓ Les implémentations qui intègrent seulement la phase de généralisation. Dans ce genre de circuits, l'apprentissage est réalisé en software permettant ainsi de générer les poids synaptiques. L'implémentation hardware du réseau de neurone consiste à charger ces poids synaptiques dans des mémoires d'où le terme anglais "***off-chip training circuits***", et à implémenter les fonctions de sommation et d'activation. Ce genre d'implémentation est le plus utilisé ; et les circuits réalisés sont très performants néanmoins, ils ne peuvent s'adapter à d'autres applications.

Dans notre étude nous nous intéressons à l`application de l'algorithme de la rétro propagation du gradient (RPG) a la classification, dans l`annexe I nous présenterons l'état de l'art des différentes implémentations de l'algorithme RPG.

II.8 Conclusion :

Les réseaux de neurones sont de puissants outils de modélisation et de prédiction. La principale caractéristique de ces réseaux est leur capacité d'adaptation. Sans connaissance préalable, et au fur et à mesure qu'on leur fournit des exemples, les réseaux de neurones forgent un modèle de connaissance. Toute cette capacité d'assimilation des exemples réside dans la possibilité d'adapter le vecteur des poids synaptiques W.

Les réseaux de neurones sont donc une façon d'élaborer un modèle de connaissance à partir d'une base de donnée. La mise au point de ce modèle de connaissance se fait par une phase d'apprentissage qui dépend de l'architecture de ce réseau et d'une règle d'apprentissage des poids.

- ✓ Le modèle de réseau doit être adapté à la complexité de la tâche à apprendre : il doit disposer de paramètres suffisants lui assurant un degré de flexibilité.

- ✓ La règle d'apprentissage des poids découle d'un critère d'erreur, sa minimisation itérative conduit à une solution optimale des poids.

Ces réseaux peuvent être implémentés sur des circuits FPGAs, cela se fait soit sur des grosses machines (Super-ordinateur) soit sur des cartes dédiées a des applications ou des projets spécifiés suivant leurs critères de performances.

CHAPITRE III

L'ALGORITHME DE RETRO PROPAGATION DU GRADIENT

III.1 Introduction

Nous avons vu dans le chapitre précédent qu'il existe un type très classique de réseau de neurones qui est le Perceptron ; Celui-ci est composé de deux ou davantage de couches successives, qui comportent chacune un certain nombre de neurones. Les nœuds d'une couche sont interconnectés avec les nœuds de la couche précédente et de la couche suivante. Ainsi, le signal se propage de la couche d'entrée jusqu'à la couche de sortie en passant à travers les différentes couches intermédiaires, ou couches cachées. Mais l'inconvénient c'est qu'il ne s'applique que sur les classes qui sont linéairement séparables.

En 1985 la Rétro propagation de gradient (RPG) apparaît comme alternative au Perceptron. C'est un algorithme d'apprentissage adapté aux réseaux de neurones multicouches (aussi appelés Perceptron multicouches). Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau en décomposant cette fonction en une suite d'étapes linéairement séparables. De nos jours, les réseaux multicouches et la Rétro propagation de gradient reste le modèle le plus étudié et le plus productif au niveau des applications.

Dans ce chapitre nous présenterons le modèle d'apprentissage de la Rétro propagation du gradient, on va décrire d'abord le réseau du RPG de point de vue structurel et comportemental, après on passe à la description de son algorithme d'apprentissage et son modèle mathématique et nous finirons par conclusion.

III.2 Réseau à Rétro propagation du gradient [6][17][20]

Ce réseau a été développé pour résoudre le fameux problème de la non séparabilité linéaire qui limitait les applications possibles des réseaux de neurones depuis de nombreuses années. Par conséquent, c'est le modèle qui a suscité le plus d'intérêt de la part des chercheurs et qui suscite encore beaucoup d'études et d'applications. En résumé, ce réseau est constitué de plusieurs couches de Perceptron.

III.2.1. Architecture

L'architecture de ce réseau *Figure III.1* est constituée de plusieurs couches d'unités à propagation vers l'avant (Feed-forward nets). L'information à l'entrée se propage par une ou plusieurs couches d'unité cachée et par une couche d'unité de sortie.

Notation :

x_i = valeurs d'entrée.
yd = valeurs de sorties désirées.
Wi = valeurs des poids de connexions.

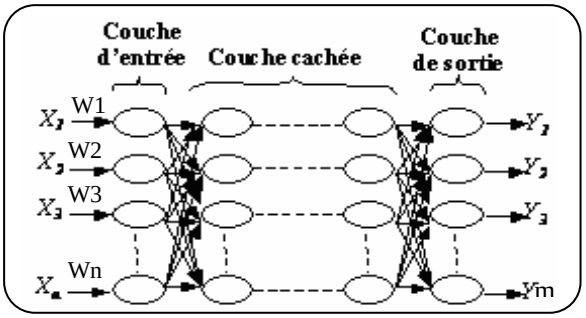


Figure III.1 Réseau à rétro-propagation du gradient

III.2.2. Transmission

- L'entrée d'un neurone x_j est donnée en fonction des sorties des neurones de la couche précédente y_i et les connexions entre les deux couches W_{ji} comme étant :
  - ✓ La sortie d'un neurone y_j est obtenue par l'application de la fonction d'activation uniformément dérivable f sur l'entrée x_j :

$$y_j = f(x_j) \qquad \text{(III.1)}$$

On prend f la fonction sigmoïde illustrée sur le graphe de la *Figure III.2* :

$$y_j = f(x_j) = \frac{1}{1 + \exp(-\alpha x_j)} \qquad \text{(III.2)}$$

Cette fonction possède deux asymptotes, une à y = 0 et une autre à y = 1. Par conséquent le modèle peut utiliser comme vecteur d'entrée, **x**, n'importe quelles valeurs réelles. Mais, les valeurs admises du superviseur doivent être comprises entre [0, 1] ou [-1,1].

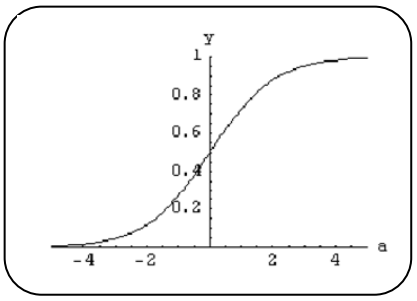


Figure III.2 Graphe de la fonction sigmoïde

Le fait d'utiliser cette règle acquiert au réseau un double avantage. Premièrement, en modifiant la pente de la fonction,α, on peut avoir une fonction de sortie qui va de simplement linéaire à une fonction non linéaire. Deuxièmement, comme nous le verrons, la fonction sigmoïde permet de simplifier les calculs pour la mise à jour des poids de connexions.

La sortie désirée d'un neurone j est notée d_j. L'erreur par rapport à la sortie désirée est notée Ed. On utilise l'erreur quadratique en Sortie du réseau, définie par :

$$E_d = \frac{1}{2}\sum_{j=1}^{n_3} (d_j - y_j)^2 \qquad \text{(III.3)}$$

Le gain du réseau ou coefficient d'apprentissage, qui détermine l'importance des modifications induites sur les poids à chaque cycle, est noté η et la modification à apporter au poids wji est donnée en fonction de η par :

$$\Delta W_{ji} = -\eta \frac{\partial E_p}{\partial W_{ji}} \quad \eta > 0 \qquad \text{(III.4)}$$

III.2.3. Apprentissage

L'idée de base de cet algorithme est que l'on veut minimiser l'erreur par rapport aux poids de connexions. L'erreur définie par la différence entre les sorties désirées et les sorties obtenues doit donc être minimisée par rapport aux poids de connexions. Par conséquent, on doit dériver cette erreur par rapport aux poids de connexions de la couche de sortie (***W***) et dériver l'erreur par rapport à la couche cachée.

III.3. Algorithme de la Rétro propagation du gradient [17][20]

On convient qu'un exemple P est formé de vecteur ***i(p)*** et ***d(p)*** qui représentent respectivement, l'état imposé à l'entrée du réseau et l'état désiré. A chaque neurone du réseau correspond une composante de ***d(p)*** et ***i(p)***. La dissemblance entre le vecteur ***dj*** et le vecteur des états des neurones ***σj*** peut être exprimée par un coût quadratique E(p) :

$$E(p) = \frac{1}{2}\sum_{j=1}^{n}(d_j(p) - \sigma_j(p))^2 \qquad \text{(III.5)}$$

Le coût quadratique doit être minimisé pour tout exemple P, cela revient à minimiser le coût quadratique global :

$$E = \sum E(p)$$

$$E = \frac{1}{2}\sum_{p}\sum_{j=1}^{n}(d_j(p) - \sigma_j(p))^2 \qquad \text{(III.6)}$$

Cet algorithme est une méthode d'évaluation du gradient dont les calculs ont été astucieusement organisés de façon à réduire leur complexité temporelle et spatiale. Il est aussi local, c'est à dire que la modification d'un poids W_{ij} ne dépend que de grandeurs définies localement au sein des neurones **i** et **j**. Cette propriété est utile pour concevoir une implantation parallèle efficace.

A chaque itération d'un algorithme de gradient le moyen le plus simple de réduire la valeur de **E** est d'ajouter à chaque poids W_{ij} une quantité ΔW_{ij} définie de la façon suivante :

$$\Delta W_{ji}^{[s]} = -\eta \frac{\partial E_p}{\partial W_{ji}^{[s]}} \qquad \eta > 0$$

Le coefficient η , appelé pas du gradient, doit être choisi suffisamment grand pour réduire au mieux le nombre d'itérations. Mais pas trop pour que l'algorithme du gradient reste stable. En introduisant le potentiel $\mu_i(P)$ du neurone i pour un exemple P et en utilisant la formule des dérivations composées, on obtient :

$$\Delta W_{ji} = -\eta \sum_{p} \frac{\partial E_p}{\partial \mu_{jp}} \cdot \frac{\partial \mu_{jp}}{\partial W_{ji}} \qquad \text{(III.7)}$$

Les neurones du réseau sont numérotés selon l'ordre suivant : si l'état du neurone i dépend de celui du neurone j, alors j<i.

En remplaçant la dépendance par rapport au temps dans les équations (III-6) et (III-7) par l'indice P de l'exemple courant présenté au réseau, on obtient :

$$\mu_j(p) = \sum_j W_{ij} \bullet \sigma_j(p) + ... \qquad \text{(III.8)}$$

$$\sigma_j(p) = f(\mu_j(p)) \qquad \text{(III.9)}$$

On en déduit :

$$\frac{\partial \mu_i(p)}{\partial W_{ij}} = \sigma_j(p) \qquad J < i$$

On pose

$$\delta_j(p) = -\frac{\partial E_p}{\partial \mu_j(p)} \tag{III.10}$$

Soit i le numéro d'un neurone quelconque du réseau, soit $E_i(P)$ la quantité :

$$E_j(p) = \frac{1}{2}\sum_n (d_n(p) - \sigma_n(p))^2 \tag{III.11}$$

On a :

$$E_j(p) = \frac{1}{2}(d_j(p) - \sigma_j(p))^2 + E_{j+1}(p)\, j \le n \tag{III.12}$$

On vérifie facilement l'identité suivante :

$$\delta j(p) = -\frac{\partial E(p)}{\partial \mu_j(p)} = -\frac{\partial E_k(p)}{\partial \mu_j(p)} \forall k \le j \tag{III.13}$$

D'où, en utilisant les expressions (III-9) et (III-12) et la formule des dérivations composées :

$$\delta_j(p) = f'(\mu_j(p))(d_j(p) - \sigma_j(p)) - \sum_{l=i+1}^{n-1} \frac{\partial E_{l+1}}{\partial \mu_l} \bullet \frac{\partial \mu_l}{\partial u_j} \tag{III.14}$$

Ou f' (μ(p)) désigne la dérivée de la fonction de transfert. On convient que la somme à droite de l'expression (III-12) n'existe pas si i=n-1. Comme l > i+1, selon l'identité (III-13) on observe :

$$-\frac{\partial E_{j+l}}{\partial \mu_l} = \delta_l(p) \tag{III.15}$$

D'autre part :

$$\frac{\partial \mu_l(p)}{\partial \mu_j(p)} = \frac{\partial\left\{\sum_{j=0}^{l-1} W_{ij}\sigma_j(p) + ...\right\}}{\partial \sigma_j(p)} \bullet \frac{\partial \sigma_j(p)}{\partial \mu_j(p)} = W_{ij} \bullet f'(\mu_j(p)) \tag{III.16}$$

En remplaçant dans (III-14) les terme calculés en (III-15) et (III-16) on obtient l'expression définitive de $\delta_i(P)$:

$$\delta_i(p) = f'(\mu_i(p)) \bullet \left\{(d_i(p) - \sigma_i(p) + \sum_{l=i+1}^{n-1} W_{ij} \bullet \delta_l(p))\right\} \tag{III.17}$$

En clair, si le neurone i est un neurone de sortie, $\delta_i(P)$ contient le terme $(d_i(P)-\sigma_i(P))$ qui représente l'erreur que commet le neurone i par rapport à la sortie désirée

$d_i(P)$. De plus, la valeur de $\delta_i(P)$ dépend des valeurs de $\delta_l(P)$, ou l est le numéro de neurones placés sur des couches supérieures à celle du neurone i. Cette information se propage donc en sens inverse de l'activation $\sigma(P)$ et ne dépend que des erreurs commises par les neurones de sortie ainsi que des poids du réseau.

Tout se passe comme si l'algorithme permettait d'attribuer une erreur commise à tout neurone du réseau, même ceux qui sont cachés. Ces considérations justifient le nom de « rétro propagation des erreurs » pour cet algorithme. Ce dernier permet de calculer une valeur exacte du gradient : il s'agit d'un algorithme de gradient total. Il est résumé ci-dessous :

- ✓ Soit $\delta(t)$ une mesure de l'erreur commise par le réseau pour tous les exemples de l'ensemble d'apprentissage à l'itération t.
- ✓ Soit E_{max} , l'erreur maximale acceptable.
- ✓ Soit c_{ij} le booléen tel que :

$$\begin{cases} C_{ij} = 1 & \text{s'il existe une connexion du neurone j vers i ;} \\ C_{ij} = 0 & \text{sinon.} \end{cases}$$

Tous les exemples sont présentés en séquence durant une époque d'apprentissage qui constitue pour cet algorithme une itération de l'algorithme du gradient, il y en a autant que nécessaire pour que l'erreur E(t) soit minimale, s'arrêter si le coût quadratique du minimum tend vers E_{max} . L'algorithme RPG peut se résumer comme suit :

Tant que $\delta(t) > \varepsilon_{max}$ **, faire** :

Pour P=0 à P-1 faire

Pour i=0 à n-1 faire -- calcul de l'état

$$\mu_j(p) = \sum_j W_{ij} \bullet \sigma_j(p) + ...$$

$$\sigma_j(p) = f(\mu_j(p))$$

Fin pour

Pour i=n-1 à0 faire --calcul de l'erreur par rétro propagation.

$$\delta_i(p) = f'(\mu_i(p)) \bullet \left\{ (d_i(p) - \sigma_i(p) + \sum_{l=i+1}^{n-1} W_{ij} \bullet \delta_i(p)) \right\}$$

Fin pour

Fin pour

--Calcul de la modification des poids après une présentation des exemples

Pour i=0 à n-1 faire

Pour j=0 à n-1 faire

$W_{ij}(t+1) := W_{ij}(t) + \Delta W_{ij}$

Fin pour

Fin pour

$\varepsilon(t) :=$ erreur (l, d, W) -- fonction qui retourne l'erreur commise par le réseau

t = t+1 ;

Fin tant que.

III.4. Calcul détaillé [6][22][24]

On considère que le MLP [14] est constitué de trois couches de neurones ; la première couche avec $\mathbf{n_0}$ entrées et $\mathbf{n_1}$ unités (neurones), la seconde avec $\mathbf{n_2}$ unités et la sortie avec $\mathbf{n_3}$ unités.

La *Figure III.3* montre une représentation fonctionnelle de l'algorithme.

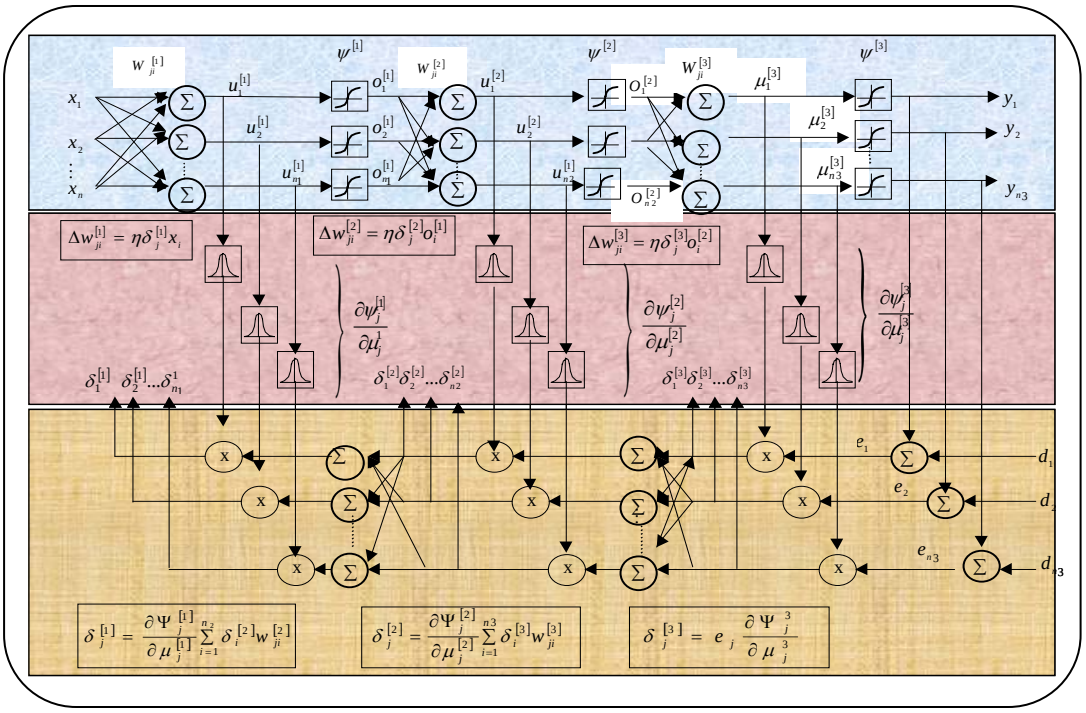


Figure III.3 représentation fonctionnelle de l'algorithme de la Rétro propagation du gradient

L'algorithme de Rétro propagation du gradient est utilisé dans le Perceptron multicouche. La phase importante ou ce dernier est utilisé est bien évidemment l'apprentissage : On présente au réseau des entrées et on lui demande de modifier sa pondération de telle sorte que l'on retrouve la sortie correspondante. L'algorithme consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau. La seconde étape compare la sortie calculée à la sortie désirée. On modifie alors les poids de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée et connue soit minimisée.

Malgré tout, il ne faut pas oublier que l'on a une couche cachée. On rétropropage alors l'erreur commise vers l'arrière de la couche de sortie jusqu'à la couche d'entrée tout en modifiant la pondération. On répète ce processus sur tous les exemples jusqu'à ce qu'on obtienne une erreur de sortie considérée comme négligeable.

L'algorithme de la Rétro propagation standard utilise la méthode de plus forte descente pour la minimisation de la moyenne carrée de l'erreur locale, qui est donnée par :

$$E_p = \frac{1}{2}\sum_{j=1}^{n_3}(d_{jp} - y_{jp})^2 = \frac{1}{2}\sum_{j=1}^{n_3} e_{jp}^2 \qquad \text{(III.18)}$$

Et de l'erreur globale donnée par :

$$E = \sum_p E_p = \frac{1}{2}\sum_p \sum_j (d_{jp} - y_{jp})^2 \qquad \text{(III.19)}$$

Où, d_{jp} et y_{jp} représentent la sortie désirée et la sortie actuelle du *j-ième* neurone de sortie pour le *p-ième* exemple.

Le problème de l'apprentissage peut être formulé comme suit : étant donné l'état actuel des poids synaptiques, on doit déterminer l'augmentation ou la diminution $\Delta W_{ji}^{[s]}$ (s étant le numéro de la couche : s=1, 2,3) des poids afin de minimiser l'erreur globale, *E*. La variation $\Delta W_{ji}^{[s]}$ s'écrit :

$$\Delta W_{ji}^{[s]} = -\eta \frac{\partial E_p}{\partial W_{ji}^{[s]}} \qquad \eta > 0 \qquad \text{(III.20)}$$

η est le coefficient d'apprentissage

Il est montré que si le paramètre η est suffisamment faible, cette procédure minimise l'erreur globale, $E = \sum_p E_p$

Pour la couche de sortie (s=3)

$$\Delta W_{ji}^{[3]} = -\eta \frac{\partial E_p}{\partial W_{ji}^{[3]}} = -\eta \frac{\partial E_p}{\partial \mu_j^{[3]}} \cdot \frac{\partial \mu_j^{[3]}}{\partial W_{ji}^{[3]}} \qquad \text{(III.21)}$$

Avec $$\mu_j^{[3]} = \sum_{i=1}^{n_2} W_{ji}^{[3]} x_i^{[3]} = \sum_{i=1}^{n_2} W_{ji}^{[3]} O_i^{[2]} \qquad (j=1,\ldots,n_3) \qquad \text{(III.22)}$$

On définit l'erreur locale à la sortie appelée delta par :

$$\delta_j^{[3]} = -\frac{\partial E_p}{\partial \mu_j^{[3]}} = -\frac{\partial E_p}{\partial e_j} \cdot \frac{\partial e_j}{\partial \mu_j^{[3]}} = e_{jp} \cdot \frac{\partial \psi_j^{[3]}}{\partial \mu_j^{[3]}} \qquad \text{(III.23)}$$

Où $\psi_j^{[3]}$ représente la fonction d'activation.

On obtient la formule pour la mise à jour des poids à la couche de sortie

$$\Delta W_{ji}^{[3]} = \eta \delta_j^{[3]} x_i^{[3]} = \eta \delta_j^{[3]} O_i^{[2]} \qquad \text{(III.24)}$$

Où, $$\delta_j^{[3]} = e_{jp}(\psi_j^{[3]})' = (d_{jp} - y_p)\frac{\partial \psi_j^{[3]}}{\partial \mu_j^{[3]}} \qquad \text{(III.25)}$$

Pour la mise à jour des poids synaptiques de la seconde couche :

$$\Delta W_{ji}^{[2]} = -\eta \frac{\partial E_p}{\partial W_j^{[2]}} = -\eta \frac{\partial E_p}{\partial \mu_j^{[2]}} \cdot \frac{\partial \mu_j^{[2]}}{\partial W_{ji}^{[2]}} \qquad \text{(III.26)}$$
$$= \eta \delta_j^{[2]} x_i^{[2]} = \eta \delta_j^{[2]} O_i^{[1]}$$

Où l'erreur locale pour la couche cachée est définie par :

$$\delta_j^{[2]} = -\frac{\partial E_p}{\partial \mu_j^{[2]}} \qquad \text{(III.27)}$$
$$= -\frac{\partial E_p}{\partial O_j^{[2]}} \cdot \frac{\partial O_j^{[2]}}{\partial \mu_j^{[2]}}$$

Avec $\mu_j^{[2]} = \sum_{i=1}^{n_1} W_{ji}^{[2]} x_i^{[2]} = \sum_{i=1}^{n_1} W_{ji}^{[2]} O_i^{[1]}$ (j=1, … n_2) (III.28)

Sachant que

$$O_j^{[2]} = \psi_j^{[2]}(\mu_j^{[2]}) \qquad \text{(III.29)}$$

On obtient

$$\delta_j^{[2]} = -\frac{\partial E_p}{\partial O_j^{[2]}} \cdot \frac{\partial \psi^{[2]}}{\partial \mu_j^{[2]}} \qquad \text{(III.30)}$$

Le facteur $-\partial E_p / \partial O_j^{[2]}$ peut s'écrire :

$$\frac{-\partial E_p}{\partial O_i^{[2]}} = -\sum_{i=1}^{n_3} \frac{\partial E_p}{\partial \mu_j^{[3]}} \cdot \frac{\partial \mu_j^{[3]}}{\partial O_i^{[2]}}$$
$$= \sum_{i=1}^{n_3} \left(-\frac{\partial E_p}{\partial \mu_j^{[3]}}\right) \frac{\partial}{\partial O_j^{[2]}} \left[\sum_{i=1}^{n_3} W_{ji}^{[3]} x_i^{[3]} \right] \qquad \text{(III.31)}$$
$$= \sum_{i=1}^{n_3} \delta_i^{[3]} \frac{\partial}{\partial O_i^{[2]}} \left[\sum_{i=1}^{n_3} W_{ji}^{[3]} O_i^{[2]} \right]$$
$$= \sum_{i=1}^{n_3} \delta_i^{[3]} W_{ji}^{[3]}$$

L'erreur locale dans la couche cachée peut être évaluée en utilisant la formule :

$$\delta_j^{[2]} = \frac{\partial \psi_j^{[2]}}{\partial \mu_j^{[2]}} \sum_{i=1}^{n_3} \delta_i^{[3]} W_{ji}^{[3]} \qquad \text{(III.32)}$$

Par analogie, la mise à jour des poids synaptiques de la couche d'entrée s'écrit comme suit :

$$\Delta W_{ji}^{[1]} = \eta \delta_j^{[1]} x_i^{[1]} = \eta \delta_j^{[1]} O_i^{[0]} = \eta \delta_j^{[1]} x_i \qquad \text{(III.33)}$$

Où l'erreur locale est déterminée par :

$$\delta_j^{[1]} = \frac{\partial \psi^{[1]}}{\partial \mu_j^{[1]}} \sum_{i=1}^{n_2} \delta_j^{[2]} W_{ij}^{[2]} \qquad \text{(III.34)}$$

$$\mu_j^{[1]} = \sum_{i=1}^{n_0} W_{ji}^{[1]} x_i \qquad (j=1,\ldots n_1) \qquad \text{(III.35)}$$

Les étapes d'apprentissage de l'algorithme de la Rétro propagation du gradient sont comme suit:

- Initialiser les poids synaptiques $W_{ji}^{[s]}$ à des valeurs aléatoires (généralement entre -0.5/fan_in et +0.5/fan_in, où le fan_in représente le nombre de neurones contenus dans la couche directement inférieure).

- Présenter un vecteur d'entrée et la sortie correspondante (désirée) et calculer les sorties actuelles de tous les neurones en utilisant les valeurs présentes $W_{ji}^{[s]}$ dans les équations (III.35), (III.28) et (III.22).

- Spécifier la sortie désirée et évaluer les erreurs locales $\delta_j^{[s]}$ pour toutes les couches en utilisant les équations (III.25), (III.32) et (III.34).

- Ajuster les poids synaptiques en accord avec la formule itérative $\Delta W_{ji}^{[s]} = \eta \delta_j^{[s]} x_i^{[s]}$ (s=3, 2,1) correspondant aux équations (III.24), (III.26) et (III.33).

- Calculer l'erreur E_p en utilisant l'équation (III.18).

- Répéter les étapes 2 à 5 avec le même vecteur d'entrée jusqu'à ce que l'erreur E_p soit très proche de l'erreur admissible.

- Répéter 2 à 6 pour chaque vecteur d'entrée X (pour chaque exemple d'apprentissage).

Tous les exemples d'apprentissage sont présentés périodiquement jusqu'à ce que les poids synaptiques soient stabilisés, i.e. jusqu'à ce que l'erreur pour tout l'ensemble complet soit acceptable et le réseau converge.
Notons aussi, qu'en plus du coefficient d'apprentissage η, on définit dans la formule de variation des poids synaptiques , ΔW, un facteur momentum α, tel que :

$$\Delta W((t+1)= W(t) + \eta\delta \ \eta\alpha\Delta$$

Le facteur momentum permet d'améliorer la vitesse de convergence tout en prévenant l'algorithme des oscillations.

III.5.Résumé des propriétés du réseau à Rétro propagation de l'erreur

- ✓ Règle d'apprentissage relativement simple si on utilise la fonction sigmoïde comme règle de transmission.
- ✓ L'apprentissage peut être fait uniquement de manière itérative.

- ✓ L'architecture implique une connaissance intrinsèque des réponses à apprendre.
- ✓ Le nombre de fois qu'un stimulus particulier est appris n'influence pas la réponse du réseau à ce stimulus.
- ✓ Les stimuli d'entrée peuvent être corrélés entre eux.
- ✓ La tâche à effectuer peut être non séparable linéairement.
- ✓ La matrice de connexions se stabilise toujours, mais pas nécessairement à un minimum global.
- ✓ Le principe de rétro propagation de l'erreur n'est pas plausible biologiquement.
- ✓ Le problème peut être affecté par les situations de sur apprentissage, ce qui le limitera dans les tâches de généralisation. Si on incorpore un autre stimulus dans la banque d'entrée, il faut recommencer.

III.6. Conclusion

Dans ce chapitre nous avons présenté une étude générale et dessellé sur l'algorithme de Retro-propagation du gradient RPG en commençant par les fondements neurobiologiques jusqu'à la formulation mathématique de l'algorithme. Cet algorithme existe sur plusieurs versions, dont chacune d'elles peut être utilisée dans une famille d'application bien déterminées.

CHAPITRE IV
CONCEPTION SOFTWARE DU CLASSIFICATEUR

IV.1 Introduction

Le but de ce travail est de concevoir un classificateur neuronal pour les trois principaux rythmes cérébraux, pour cela nous avons entamé une conception software basé sur les concept de base de développement des architecture a base de neurone moyennant l'outil Matlab, dans cette conception nous avons utilisé une base de données de signal EEG que nous avons construit nous même. L'objectif de cette partie est de définir l'architecture neuronale qui effectue l'opération de la classification avec les plus hautes performances possibles calculés a la fin de la phase de teste dans la conception.

Avant de passer à la description de l'approche suivie dans cette conception, nous rappelons que les rythmes cérébraux principales concernés par la classification sont :

1. rythme alpha **(α).**
2. rythme bêta **(β)**.
3. rythme thêta **(θ)**.

Ces rythmes sont caractérisés par leur forme de signal aléatoire, leurs amplitudes qui sont de l'ordre des μ volte et de leurs fréquences varient de 4 Hz à 30 HZ.

Dans l'approche suivie, il s'agit de réaliser un classificateur neuronale qui utilise le signal EEG comme données d'entrée pour effectuer l'opération d'apprentissage supervisé baser sur l'algorithme de la rétro propagation du gradient a fin d'effectuer la classification des rythmes cérébraux en question.

Les résultats obtenus sont résumés et interprétés à la fin de ce chapitre ou nous allons voir les points suivants :

- Les définitions des différents paramètres de performances d'un classificateur.
- la description de la construction de la base de données a utilisée dans ce travaille.
- la présentation de la programmation utilisée dans cette application a fin d'obtenir la bonne architecture pour effectuer la classification.
- Une interprétation des résultats obtenue.
- Conclusion.

IV.2 Définition des paramètres de performance d'un classificateur [9] [24]

A fin d'évaluer les performances de notre classificateur, nous avons utilisé des mesures de performances communes aux systèmes de diagnostic. En général, nous définissons les paramètres suivants :

IV.2.1 Valeur positive vraie (TP') : Une valeur positive vraie ou « true positive » **TP'** est définie par le nombre des vecteurs de tests, et pour lesquels le rythme identifiée par le système de décision coïncide avec celle identifie par le médecin.

IV.2.2 Valeur négative vraie (TN') : Une valeur négative vraie ou « true negative » **TN'** est définie par le nombre des vecteurs de tests qui présentent un rythme différent, et pour lesquels le système de décision affirme le même résultat.

IV.2.3 Valeur positive fausse (FP') : La valeur positive fausse ou « false positive » **FP'** est définie par le nombre de vecteurs de tests présentant une rythme (identifiée par le médecin) et pour lesquels le système, par erreur de classification n'identifie pas le rythme.

A partir de ces mesures nous définissons les pourcentages suivants :

$$\%CC=100*(TP'+TN')/N \qquad \text{(IV.1).}$$

Où :

N : le nombre des vecteurs de test.
%CC : le pourcentage de classification correcte.

De même nous définissons les paramètres de performances suivants :

$$\%PR=100*(TP'+FP')/N \qquad \text{(IV.2).}$$

Où :

PR : la précision du système.

IV.3 Construction de la base de données

Dans le cas de notre application concerné par la classification des différents rythmes du signale encéphalographie EEG, les bases de données sont primordiale pour la naissance et le bon déroulement du travaille, ce pendant, ce type de base de données n'ai pas tous le temps disponible pour deux principales raisons

1. l'anonymat des données et les étiques et le secret médicales qui interdissent l'échange de ces données or le corps médicale pour une raison ou une autre qu'avec une autorisation du médecin responsable.

2. les bases de données de ce genre qui sont disponible sur le net sont soit chère à payer, spécifique à des applications bien définie ou même de format particulier qui nécessite des logiciels bien spécifiques pour les manipuler.

Cette difficulté d'obtention de la base de données nous a guidais vers la construction de notre propre base d'enregistrements EEG, utilisent des enregistrements EEG réels, un PC et un logiciel de traitement qui sert a digitalisé ces enregistrements.

La construction de cette base de données a passée par les étapes suivantes :

1. effectuer un enregistrement EEG au sien de l'hôpital BEN AKKNOUN service neurologie.
2. Récupération des enregistrements sous format image .bmp.
3. Analyse des enregistrements et l'extraction des régions d'intérêt pour cette application.
4. L'échantillonnage des signaux a utilisé dans l'application et les sauvegardés sous format « .m » sous Matlab.

L'étape d'échantillonnage a été effectuée à l'aide d'un logiciel propre à ce type d'opération appliqué a des images des différents extension provenant soit d'une opération de scanne, d'une sauvegarde sous format image ou même d'un téléchargement sur web, la il s'agit du logiciel ***CurveUnscan.***

IV.3.1 CurveUnscan définition est application

CurveUnscan est un logiciel de reconnaissance de graphiques. Il permet, à partir d'un fichier graphique résultant du scannage d'une courbe (disponible initialement en général sur un support papier, d'une image quelconque enregistrer sur PC), d'identifier les coordonnées des points de cette courbe. Les données ainsi numérisées sont exportées dans un fichier texte qui est formaté afin d'être directement exploitable avec n'importe quel tableur, Excel par exemple Figure IV.1.

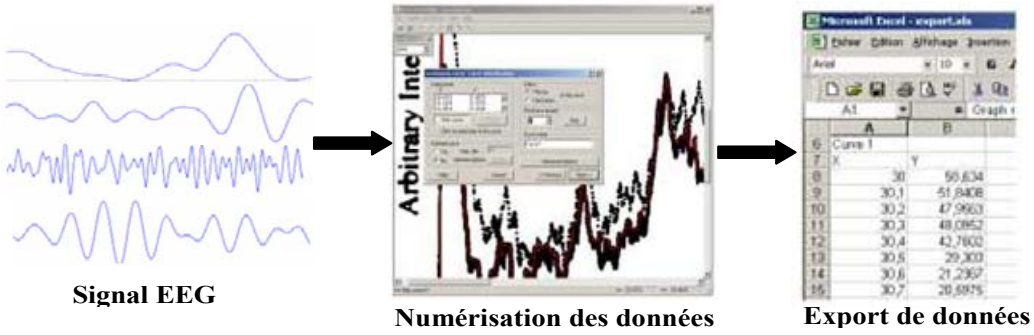


Figure IV.1 l'application de l'outil CurveUnscan

CurveUnscan décompose le processus de reconnaissance d'un graphique en trois principales étapes.

- Identification du système de coordonnées
- Identification de la courbe
- Définition de la présentation des données numérisées

Chacune de ces étapes est indépendante, du point de vue des informations traitées, ce qui permet de corriger ou modifier les données de chacune tout en impactant peu ou pas les autres étapes.

Apres avoir télécharger la courbe a digitalisé dans l'espace de travaille de CurveUnscan, la première étape a effectuer est la définition du système de coordonnées adéquat a l'opération ciblée, la deuxième étape, est de choisir le type de courbe a analysée pour le reste du traitement qu'on veut faire avec CurveUnscan. Il est a note que CurveUnscan permet d'identifier trois types de courbes : les courbes discrètes (de type nuage de points), les courbes continues et les courbes de densitomètre. Les courbes discrètes, ou nuage de points, sont composées d'une succession de points sans lien entre eux. Ces points sont individuellement identifiés par l'utilisateur. Les courbes continues peuvent faire l'objet d'une reconnaissance automatique à partir de différents éléments définis par l'utilisateur : intervalle de définition, position sur le graphique, épaisseur du trait, ...etc. Les courbes de densitométrie consistent à analyser le profil de couleurs selon un segment du graphique.

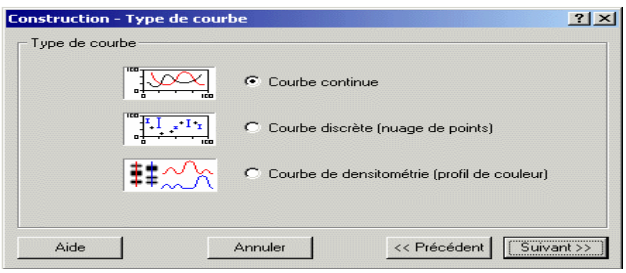


Figure IV.2 choix de la courbe a traité

Dans notre cas le type de courbe que nous utilisons sont des courbes continues a digitalisées, pour cela on passe par trois étapes essentielles pour effectuer la numérisation de la courbe en question.

1. définir les limites de l'intervalle de définition de la région d'intérêt sur la courbe. Il est possible d'effectuer cette opération soit en entrant la valeur de ces limites dans le champ correspondant, soit en cliquant sur l'endroit voulu de la courbe. Le champ correspondant est alors automatiquement remplit avec la valeur d'abscisse du point choisi.

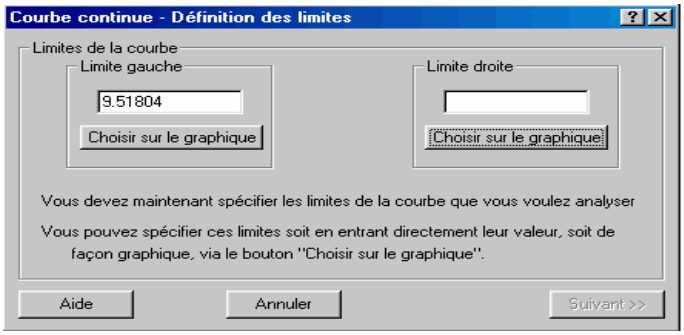


Figure IV.3 définition des limites de la courbe

2. l'identification de la courbe en cliquant sur plusieurs endroits sur la courbe pour que CurveUnscan puisse suivre l'évolution de cette dernière.

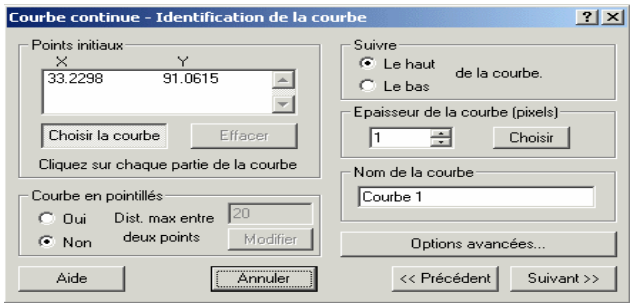


Figure IV.4 Identification de la courbe

3. la mise en forme des résultats de la numérisation et la sauvegarde, la il s'agit de faire repère à la première abscisse sur la courbe et définir la fréquence d'échantillonnage que nous voulons appliquer et puis choisir le format sous laquelle on veut enregistrer nous données.

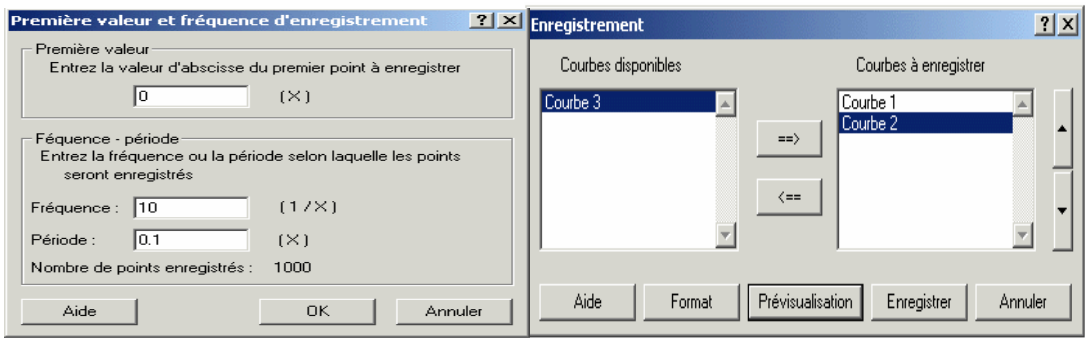


Figure IV.5 Enregistrement de la courbe échantillonnée

La manipulation et le traitement du tracé de l'enregistrement EEG que nous avons effectué au niveau de l'hôpital, nous a permet construire une base de donnée

contenant un ensemble de 60 vecteurs représentant les exemples de trois rythmes a noté le rythme alpha, le rythme bêta et le rythme thêta. Ces vecteurs d'exemples contenant chacun 82 échantillons, les 81 premiers échantillons représentent la morphologie du signale et le 82 nième échantillon représente la fréquence correspondant a l'exemple. Cette base de données regroupe les vecteurs utilisés pour l'apprentissage bien que celles utilisé pour le teste.

Table IV.1 : La base de donnée

	N d'exemples
Rythme alpha	20
Rythme beta	20
Rythme thêta	20

Apres avoir construire notre base de données, la suivante étape est de décrire l'approche de la classification que nous utilisant dans cette application, a fin de pouvoir dimensionner le réseau MLP convenable pour la classification dans ce cas.

IV.4 Programmation [27]

L'approche à suivre dans ce travaille consiste a concevoir un classificateur neuronale pour les rythmes cérébraux a noté alpha, bêta et thêta utilisant l'enregistrement EEG. L'aidé principale autour de cette application est l'utilisation de l'algorithme de la rétro propagation du gradient (RPG) applique comme algorithme d'apprentissage supervisé a un perceptron multi couche (MLP) a fin d'effectuer la classification. Cette conception nécessite le passage par une phase de conception software a fin de pouvoir bien dimensionner le réseau et tester ces performances avons de passer a la phase de la conception et l'implémentation hardware. Pour cela nous avons utilisé l'outil « **Matlab 7.0.1 » [27]**. Le choix de Matlab était basé sur l'avantage que ce dernier possède une librairie riche des fonctions prédéfinies qui facilitent la tache de programmation des fonctions complexes comme celles de l'algorithme de la rétro propagation du gradient (**RPG**). Cette librairie dite « neural net work » contient toutes les applications des réseaux de neurones et parmi laquelle on trouve les différentes fonctions prédéfinies qui nous avons utilisé dans cette application a noté :

1. *premnmx :* cette fonction prétraite la formation de réseau réglée en normalisant ses entrées et ses cibles de sorte qu'elles appartiennent à l'intervalle [-1,1], ce prétraitement se fait en appliquant la formule (IV.3) aux composants des éléments précédents.

   $$pn = 2*(p-minp)/(maxp-minp) - 1 \qquad (IV.3)$$

   Où

   Pn : est le vecteur normalisé.
   P : est le vecteur à normaliser.
   minp : le plus petit échantillon dans le vecteur P.
   maxp : le plus grand échantillon dans le vecteur P.

2. *newff :* c'est une nouvelle fonction modélisant l'algorithme Feed_Forward, à chaque appel de la fonction newff, l'outil Matlb crée un nouveau réseau avec une zone de dialogue.
3. *init :* c'est la fonction d'initialisation des différents paramètres dans la newff.

4. *sim :* simule le modèle grâce à l'outil « Simulink » en utilisant tous les arrangements de dialogue de paramètre de simulation comprenant des options de la zone de travail I/O.
5. *plot :* dessine le résultat de la fonction train dans un plan.
6. *tansig :* c'est la fonction sigmoïde définie par la forme mathématique (IV.4) suivante :

$$y = 2/ (1+\exp (-2*x))-1 \qquad \text{(IV.4)}$$

Où

y : est la sortie du bloc de la fonction d'activation.
x : est la somme pondérée présente à l'entrée du bloc de la fonction d'activation.

7. *purelin* : c'est une fonction de transfert définie par la forme mathématique (IV.5).

$$y=\text{purelin}\ (y) \qquad \text{(IV.5)}$$

Où

y : une valeur présentée à l'entrée de la fonction linéaire *purelin.*

8. *train :* forme les connexions dans le réseau selon la fonction de transferts utilisés et les performances choisies.

Dans cette librairie on trouve aussi des paramètres propres qui sont :

1. l'entrée « p » : désigne le vecteur d'entrée.
2. target « t » : désigne le vecteur cible à la sortie dans le cas d'un apprentissage supervisé.
3. le sortie « y » : désigne la sortie réelle calculée par le réseau.
4. l'erreur « e » : désigne l'erreur calculée par le réseau en appliquant la forme mathématique (IV.6) suivante.

$$e = y-t \qquad \text{(IV.6)}$$

Où

e : est l'erreur calculée.
y : la sortie réelle du réseau.
t : la sortie désirée (cible).

Il est a rappeler que l'algorithme RPG a appliqué dans cette application regroupe un ensemble de séquences d'opérations a effectuer afin d'ajuster le mieux les poids synaptique et les biais pour une application bien définie comme la classification dans notre cas. Ces opérations sont tous intégrées et prédéfini dans la librairie Neural Network de l'outil Matlab Figure IV.6.

L'algorithme RPG appliqué a la classification

1. charger les vecteurs d'apprentissage.

*2. appliquer l'algorithme (**RPG**) pour la classification.*

a. propagation de l'information vers l'avant (vers la sortie).

b. Faire la classification.

c. Calcule d'erreur.

*d. Ajustement des poids (**Wij**), les biais (**Bi**) et les autres paramètres a ajustés en propageant l'erreur calculée Vers l'entrée.*

e. Refaire les étapes a, b, c, d jusqu'à arrivé aux performances impose a obtenir.

*f. Sauvegarder les poids (**Wi**j) et les biais (**Bi**) ajustes.*

3. charger les vecteurs de tests.

4. simuler pour pouvoir tester l'efficacité de notre classificateur.

Figure IV.6 les étapes de l'algorithme RPG appliqué a la classification

Le programme que nous avons développé sous Matlab pour effectuer la classification, et qui sert à bien dimensionner l'architecture du classificateur est présenté dans la Figure IV.7

Programme sur Matlab

```
EEG = [];
t = [1 3 2 1 3 2  1 3 2   1 3 2 1 3 2 ];   /* la sortie désirée dans la phase d'apprentissage*/
EEG =EEG';
[EEGn] = premnmx (EEG);
   net = newff (minmax (EEGn),[3 13 10 9 3 1],{'tansig'  'tansig'   'purelin'});   /* la fonction  feed forward*/
   net = init (net);
 net =train (net, EEGn, t);
 y = sim (net, EEGn) ;              /*   partie de calcul des valeurs réelles  de la sortie du réseau et de
 e = y-t                            l'erreur commise sur ce calcul par rapport a celles désirées */
 e = e';
figure (1) plot(e);
w1 = net.IW {1,1}; w2 = net.LW {2,1}; w3 = net.LW {3,2}; /* partie du sauvegarde des poids et des baies */
b1=net.b {1};  b2=net.b {2};  b3=net.b {3};
 x = [];  x = x';
 [xn] = premnmx (x);       /* partie de la simulation */
 y= sim (net,xn)
```

Figure IV.7 Programme Matlab de fonctionnement du classificateur

IV.5.6 descriptions de l'approche et Dimensionnement du classificateur

L'approche que nous proposons consiste à concevoir un classificateur neuronal basé sur l'algorithme de la rétro propagation du gradient RPG, pour les rythmes cérébraux à noter le rythme alpha, le rythme Bêta et le rythme Thêta. Comme le choix d'une architecture des réseaux de neurones propre a une application bien dédier, n'ai pas une tache facile par le fait que c'est une opération qui n'obéit a aucune règle, seule l'opération de test du fonctionnement du réseaux ainsi que le test de ça repense au critère imposées peut remédier a ce problème, donc, pour un bonne choix de l'architecture il faut vérifier que le réseau présent une bonne convergence rapide toute en respectons l'erreur minimale de cette convergence figure IV.8. Le cas de cette application ou l'architecture (3 13 10 9 3 1) est adopté après une phase d'apprentissage qui a présenter :

- une convergence rapide (30 itérations),
- une erreur de convergence de l'ordre de ($\mathbf{10^{-8}}$)

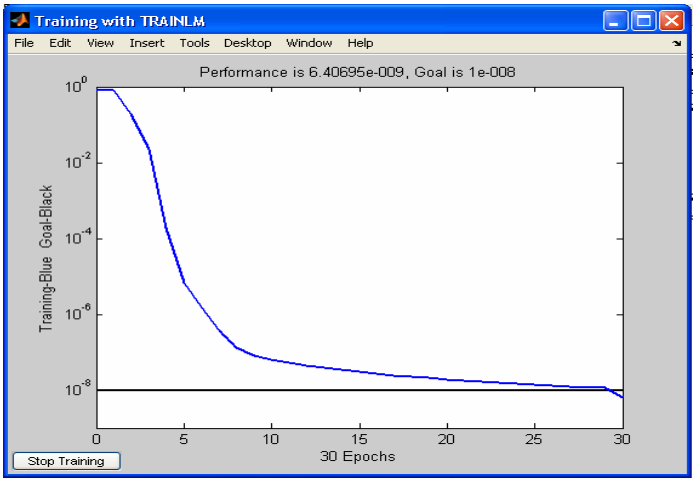


Figure IV.8 courbe de convergence du réseau (3 13 10 9 3 1)

Dans cette application, on fait apprendre à notre réseau des exemples du signal EEG représenter par la forme (morphologie) associer a la fréquence propre a chaque exemple, ces exemples sont présentés au réseau sous forme de vecteurs qui contiennent 82 échantillons les 81 premiers représentent la forme du signale et 82 nième représente la fréquence.

L'architecture (3 13 10 9 3 1) comporte trois neurones dans la couches d'entrés, le chiffre trois représente de nombre de formes d'onde qu'on veut classer (alpha, bêta et thêta), cette architecture présente aussi quatre couches cachées de tailles différentes, treize neurones, dix neurones, neuf neurones et trois neurones respectivement, et en fin elle comporte un neurone dans la couche de sortie, ce dernier donne la décision correspondante a la class du vecteur soit 1 pour le rythme alpha, 2 pour le rythme bêta et 3 pour le rythme thêta. Figure IV.9.

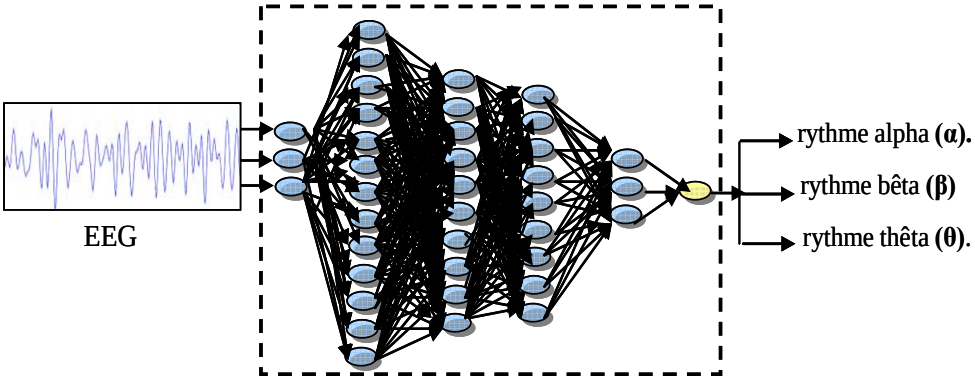


Figure IV.9 description de l'approche de classification

IV.6 Résultats de la classification

Pendant la phase de test, nous avons utilisé un ensemble de vecteur de test contenant 42 vecteurs représentant les différents rythmes à classer.

Les résultats de test de ces 42 vecteurs sont regroupés dans le Tableau (IV.2) et Ces résultats sont calculés à partir des mesures des performances (voir section (IV.2)) et des pourcentages %**CC** et %**PR** calculées par les formules.

Tableau IV.2 résultats de classification

	N	TP'	FP'	%CC	% PR
Alpha	14	9	5	64.28 %	64%
Bêta	14	12	2	85.7 %	85%
Thêta	14	13	1	92 %	92%
Totale	42	34	8	80.33 %	81%

IV.7 Interprétation des résultats

En analysant les résultats présentés dans le tableau IV.2 on remarque que les pourcentages de classification pour les différents rythmes sont de valeurs distincts, qui varient entre 64 % pour le rythme alpha et 92% pour le rythme thêta. Cette différence est tout a fait logique en regardant la régularité que présente la forme d'onde du rythme thêta par rapport a la forme d'onde que présente le rythme alpha est le rythme bêta. Chose qui nous permis de conclure que lorsque la régularité de la forme d'onde augmente plus le tau de classification et de reconnaissance de cette onde devient important.

Le taux de classification globale du classificateur qui est de 80.33% est un tau acceptable, regardant les formes d'onde qu'il classe et la difficulté de séparer les différents rythmes dans un enregistrement EEG. Néanmoins, ce taux de classification peut être amélioré en procédant à des optimisations dans la méthode de conception, en améliorant la qualité des données d'entré par des procédé de traitement de signale tels que la compression et le filtrage, et aussi améliorer l'architecture elle-même ou même adopter d'autre approche tels que concevoir un classificateur composé de plusieurs sous classificateurs, chacun d'eux est propre a la classification d'un rythme par rapport aux autres.

IV.8 Conclusion

Dans ce chapitre nous présentons l'approche que nous avons proposé pour effectuer la classification des rythmes cérébraux alpha, bêta et thêta. Ici, il s'agit de décrire les différentes étapes à suivre pour arriver à déterminer l'architecture du classificateur et évaluer ces performances dans une conception software qui a commencé par la construction de la base de données et arrivons à l'évaluation et l'analyse des résultats de classification obtenus

CHAPITRE V
L'IMPLEMENTATION HARDWARE DUCLASSIFICATEUR SUR FPGA

V.1 Introduction

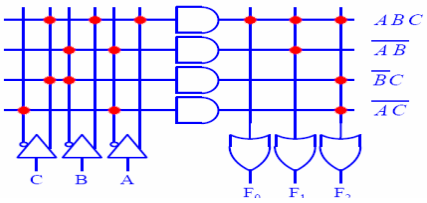
Les progrès technologiques continus dans le domaine des circuits intégrés ont permis la réduction des coûts, de la consommation, et c'est maintenant un lien commun d'affirmer que les circuits intégrés spécifiques d'une application ont permis une réduction de la taille des systèmes numériques, ainsi que la réalisation de circuits de plus en plus complexes, tout en améliorant leurs performances et leur fiabilité.

Les circuits programmables de type FPGA, apparus à la fin des années 80, prennent une part de plus en plus importante dans le marché des circuits intégrés à application, l'intégration de nouvelles fonctions (mémoires, processeurs, power PC, etc.) lui permettent de sortir de ses applications classiques de traitement d'opérations de logique booléenne et d'accéder aux applications complexes de traitement du signal et d'images.

V.2 Les circuits logiques programmables [30, 31]

Les circuits logiques programmables, ou les réseaux logiques programmables, sont des circuits logiques intégrés qui peuvent être reprogrammés à volonté en quelques dizaines de millisecondes. Ils sont flexibles, rapides et à très haute densité d'intégration (***V**ery **L**arge **S**cale **I**ntegration*). Ces circuits ont subi plusieurs évolutions, dont :

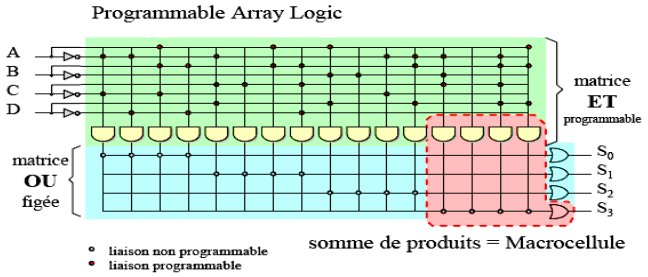
▪ **PLD** (***P**rogrammable **L**ogic **D**evice*) : Comprend une matrice de portes "AND" connectée à une matrice de portes "OR" comme présentée dans la Figure V.1.



***FigureV.1 :* Schéma d'un PLD.**

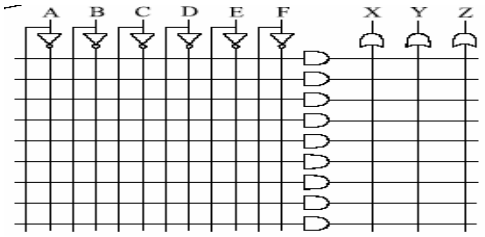
PLD est alors représenté sous forme de sommes de produits, l'on trouve aussi des **EPLD** qui sont des circuits reprogrammables effaçables par rayons ultraviolets.

▪ **PAL** (***P**rogrammable **A**rray **L**ogic*) : c'est un circuit qui présente une architecture constituée d'un plan de portes "AND" programmables suivi d'un plan de portes "OR" fixe ou figé. Figure V.2.



FigureV.2 Schéma bloc d'un PAL

▪ **PLA** (***P**rogrammable **L**ogic **A**rray*) : un circuit programmable constitué d'un plan de portes "AND" suivi d'un plan de portes "OR". Mais dans ce cas, les connexions dans les deux plans sont programmables. FigureV.3.



FigureV.3 Schéma d'un PLA

- **FPGA** (***F**ield **P**rogrammable **G**ate **A**rrays*) : ou "réseaux logiques programmables" sont des composants VLSI entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. Le progrès de ces technologies permet de faire des composants toujours plus rapides, et à plus haute densité d'intégration, qui permettent la programmation d'applications importantes.

  Les performances des FPGAs sont déterminées notamment par les délais de propagation de signaux, à travers des blocs-clés du circuit en fonction de différents facteurs, comme la température et l'alimentation. Actuellement, les délais sont voisins de la nanoseconde et les fréquences sont supérieures à 100MHz. Autant dire que ce sont des circuits très rapides.

  Depuis leur lancement, les circuits FPGAs ont révolutionné le domaine de l'électronique en apportant un plus par leur simplicité d'utilisation, en sus de leur coût sur le marché. Cette génération de circuits à logique programmable, permet d'implémenter n'importe quel algorithme en utilisant des outils de conception et un langage de programmation fourni par le constructeur. Parce qu'elle est la première à mettre ce circuit sur le marché, la société XILINX détient la plus grosse part de l'offre. Parmi les nombreux fabricants, de circuits FPGA, XILINX et ALTERA sont les plus connus **[32, 33].**

  Nous avons recouru au circuit FPGA pour implémenter notre architecture : le VIRTEX-II de XILINX **[34].**

V.3 L'architecture interne des FPGAs [35, 36]

Les circuits FPGAs sont constitués d'une matrice de blocs logiques programmables entourés de blocs d'entrées sorties programmables. L'ensemble est relié par un réseau d'interconnexions programmable Figure V.4.

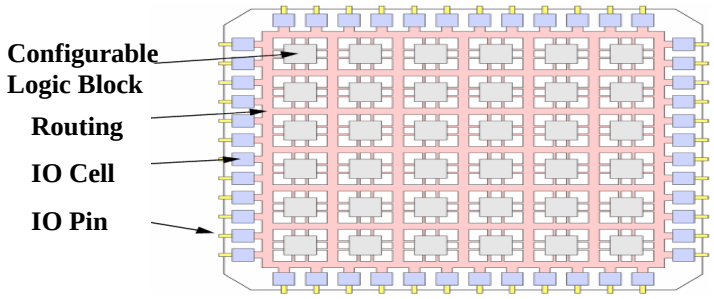


Figure V.4 Architecture interne d'un FPGA.

Distincts des autres familles de circuits programmables, les FPGAs offrent, en sus, un niveau supérieur d'intégration logique. L'architecture, retenue par XILINX, est constituée de deux couches :

- une couche appelée circuit configurable.
- une couche réseau mémoire SRAM.

La couche dite "circuit configurable" est constituée d'une matrice de blocs logiques configurables CLB permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, il y a des blocs entrées/sorties IOB dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs.

La programmation du circuit FPGA appelée aussi LCA (***L**ogic **C**ells **A**rrays*) consiste, par le biais de l'application d'un potentiel adéquat, sur la grille de certains transistors à effet de champ, à interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux. Ces potentiels sont tout simplement mémorisés dans le réseau mémoire SRAM.

V.4 Les éléments constitutifs d'un circuit FPGA [37]

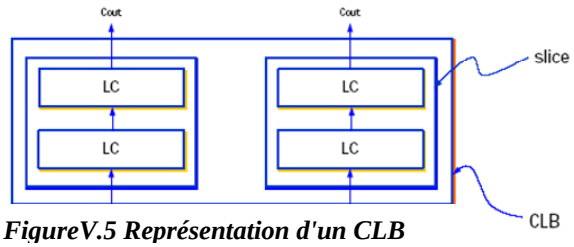
Les éléments constitutifs d'un FPGA ont les mêmes principes de fonctionnement quelle que soit l'architecture choisie. Toutefois, chaque fabricant ayant ses variantes, un circuit FPGA peut avoir des éléments différents par rapport à un autre. Un circuit FPGA comporte cinq éléments de base :

- les éléments logiques ;
- les éléments de mémorisation ;
- les éléments de routages ;
- les éléments d'entrées/sorties ;
- les éléments de contrôle et d'acheminement des horloges.

V.5 Les familles des FPGAs de Xilinx [38]

XILINX fabrique une large gamme de circuits FPGA pour diverses applications, dont : XC2000, XC4000, Spartan-II, Virtex, Virtex-E, Virtex-II, Virtex-II-Pro, Virtex 4 et virtex 5.

Les plus onéreux sont les FPGAs de la famille des Virtex. Dans cette famille l'élément logique du grain le plus fin est appelé un LC (***L**ogic **C**ell*), il est composé d'une LUT (***L**ook **U**p **T**able*), d'un registre (pour synchroniser si nécessaire la sortie) et d'une chaîne de propagation rapide de la retenue (de ce fait on peut réaliser un full adder 1 bit par LC). Les LUTs à 4 entrées peuvent réaliser n'importes quelles fonctions logiques à 1, 2, 3 ou 4 entrées. De plus les LUTs peuvent être utilisés en RAM, ROM ou registres à décalages, ce qui permet un certain nombre de fonctions réalisables par SLICE. Dans le grain logique supérieur, l'élément est appelé un SLICE, composé de deux LC. Les SLICEs sont ensuite regroupés par deux pour former un CLB comme représenté dans la Figure V.5.



FigureV.5 Représentation d'un CLB

Technologie récente des FPGA, le VIRTEX 4, qui a été dévoilé le 13 Septembre 2004, dispose de nouvelles architectures à technologie mixte, à base de FPGA et de DSP, permettant des capacités de calcul massives et comprenant des systèmes d'entrées/sorties haut débit. **[39].** Dans notre application nous étudions l'architecture détaillée du circuit VIRTEX-II.

V.6 La famille VIRTEX-II [36][38][40]

La famille VIRTEX-II est le premier jeu de plate-forme FPGA, conçue pour réaliser des conceptions à faible ou grande densité d'intégration et exige des performances élevées (jusqu'à 10 millions de portes logiques). Elle offre une densité variant de 40 000 à 8 millions portes logiques et peut atteindre une fréquence de fonctionnement de 420MHz. Son architecture est divisée en deux types de cellules de base. Figure V.6

- les cellules d'entrées/sorties appelés IOB (***I**nput **O**utput **B**loc*).
- les différentes cellules logiques appelées CLB (***C**onfigurable **L**ogic **B**loc*) sont reliées entre elles par un réseau d'interconnexions configurable.

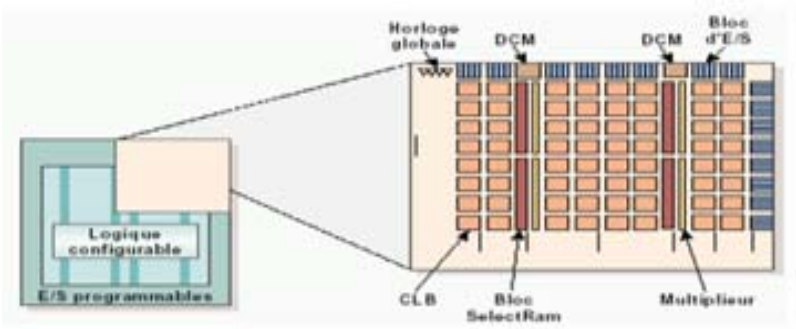


Figure V.6 Architecture interne de la famille VIRTEX-II

V.6.1 Les IOB (*Input Output Bloc*)

Les blocs d'entrées/sorties permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils 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 inutilisée (haute impédance). Figure V.7.

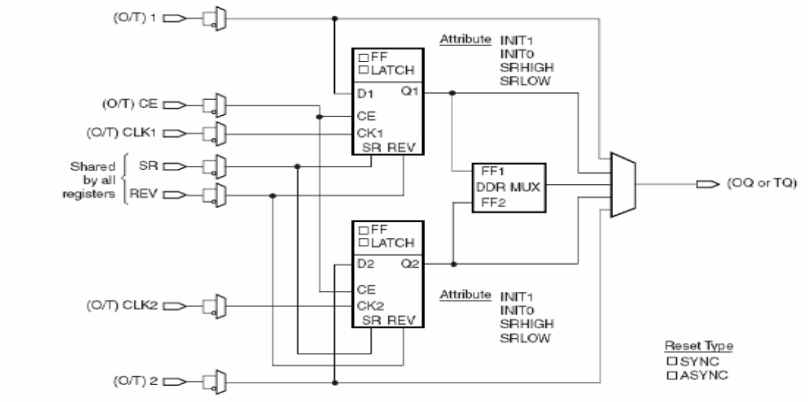


Figure V.7 Architecture interne d'un IOB

V.6.2 Les blocs logiques [38]

Les blocs logiques sont les éléments déterminants des performances du FPGA. Chaque bloc est composé de quatre cellules logiques (LC ou ***L**ogic **C**ell*) réparties en deux tranches identiques, ils servent à construire les circuits numériques implantés sur circuit FPGA. Chaque LC contient essentiellement :

V.6.2.1 Les blocs logiques configurables (CLBs) : C'est l'unité fondamentale du bloc logique qui fournit des éléments utilitaires pour la logique combinatoire et la logique synchrone, y compris les éléments du stockage de base qui sont les buffers associés à chaque CLB. Figure V.8. Les CLBs incluent deux buffers et quatre parties identiques appelées SLICE.

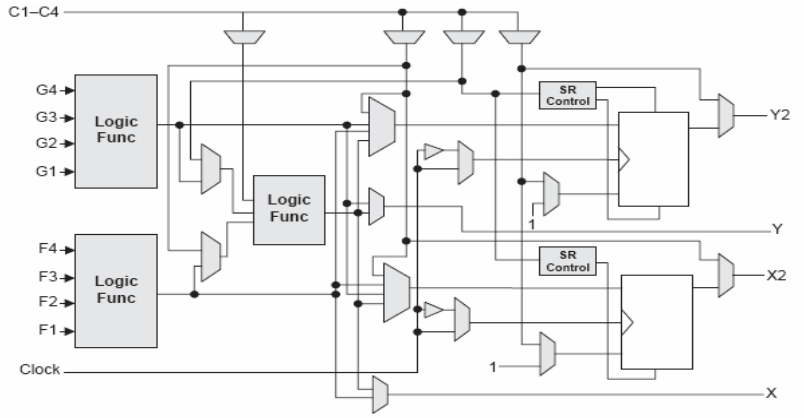


Figure V.8 Architecture d'un CLB

Chaque SLICE contient principalement :

- deux générateurs de fonction ;
- deux éléments du stockage ;
- des portes de la logique arithmétiques ;
- des multiplexeurs ;
- une chaîne de cascade horizontale (porte OU).

V.6.2.2 Les blocs mémoires (Select RAM) : possèdent des signaux d'initialisation (set et reset) synchrones ou asynchrones. Le bloc Select RAM produit de larges éléments de stockage.

V.6.2.3 Les blocs multiplieurs : chaque multiplieur peut être associé au bloc Select RAM ou peut être utilisé indépendamment **Figure V.6.**

V.6.2.4 Les blocs DCM (*Digital Clock Manager*) : le DCM produit le nouveau système d'horloges (soit intérieurement ou extérieurement au FPGA) et produit une large gamme de fréquences de l'horloge de multiplication et même de la division.

V.7 Nomenclature d'un circuit VIRTEX-II [41]

Les circuits FPGAs de XILINX sont caractérisés par une nomenclature spécifique qui définie les performances de chaque famille, cette nomenclature est illustrée dans la Figure V.9.

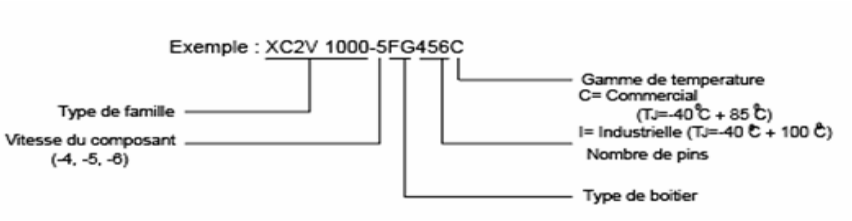


Figure V.9 La nomenclature d'un circuit Virtex-II.

Où :

- ✓ XC: **Xilinx** Circuit ;
- ✓ 2V : « **Device type** » représente le type de famille, dans ce cas c'est VIRTEX-II ;
- ✓ 1000 : c'est le nombre de portes logiques, dans ce cas il s'agit de un million de portes ;
- ✓ 5 : « **Speed Grade** » c'est la vitesse du composant selon la technologie utilisée dans la fabrication du circuit ;
- ✓ FG : « **Package type** » représente le type de boîtier ;
- ✓ 456 : « **Number of pins** » indique le nombre de pins dans le circuit ;
- ✓ C : « **Temperature Range** » donne la gamme de température tolérée.

V.8 Flot de conception des circuits FPGAs [42]

La configuration du circuit est mémorisée sur la couche réseau SRAM et stockée dans une ROM externe. Un dispositif interne permet à chaque mise sous tension de charger la SRAM interne à partir de la ROM.

Pour programmer un FPGA, il faut commencer par décrire le design en utilisant un langage de description matériel (tel que VHDL, Verilog,...). Le synthétiseur va ensuite générer la liste d'interconnexion (*netlist*) et permet de simuler le placement de tous les composants au sein du circuit FPGA, d'effectuer le routage entre les différentes cellules logiques et de calculer le délai des différents signaux. Si le routage a pu être correctement effectué, le bitstream qui sera prêt à être envoyé vers le FPGA est alors généré. C'est à ce moment là que la programmation proprement dite pourra avoir lieu. Figure V.10.

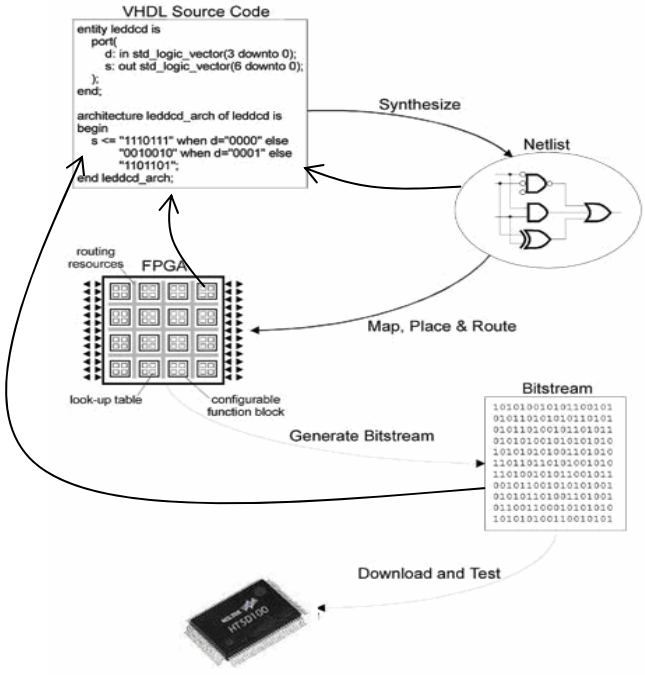


Figure V.10 Flot de conception des circuits FPGAs

V.9 Langage de description VHDL [43]

V.9.1 Bref historique

Le VHDL-VHSIC (***V**ery **H**igh **S**peed **I**ntegrated **C**ircuit*) Hardware Description Langage, a été demandé par le DOD (***D**épartement de la **D**éfense **A**méricain*) pour décrire les circuits complexes, de manière à établir un langage commun avec ses fournisseurs. C'est un langage, standard IEEE 1076 depuis 1987, qui aurait du assurer la portabilité du code pour différents outils de travail (simulation, synthèse pour tous les circuits et tous les fabricants). Malheureusement, ce n'est pour l'instant pas le cas, bien que plusieurs vendeurs aient tendance à se rapprocher du VHDL utilisé par Synopsis.

Une mise à jour du langage VHDL s'est faite en 1993 (IEEE 1164) et en 1996, la norme 1076.3 a permis de standardiser la synthèse VHDL.

V.9.2 Applications du langage VHDL [44] [45]

Le VHDL est un langage unique permettant de faire :

1. La spécification : grâce à son niveau élevé d'abstraction, le langage VHDL est très bien adapté à la modélisation des systèmes numériques complexes. La partition en plusieurs sous ensembles permet de subdiviser un modèle complexe en plusieurs éléments prêts à être développés séparément.

2. La simulation : notion du temps qui, présente dans le langage, permet son utilisation pour décrire des fichiers de simulation (Test-Bench). Le modèle comportemental avec les fichiers de simulation peuvent ensembles constituer un cahier des charges. Les fichiers de simulation peuvent également être utilisés avec un banc de tests de production.

3. La synthèse logique : logiciels de synthèse qui permettent de traduire la description VHDL en logique. Il est ainsi possible d'intégrer la description dans un composant programmable (CPLD, FPGA) ou dans un circuit ASIC.

4. La preuve formelle : langage qui permet de prouver formellement que 2 descriptions sont parfaitement identiques au niveau de leur fonctionnalité.

V.9.3 Structure d'une description VHDL [46]

La structure typique d'une description VHDL est composée de deux parties indissociables qui sont : l'entité et l'architecture.

- **L'entité :** est vue comme une boite noire avec des entrées et des sorties caractérisées par des paramètres. Elle représente une vue externe de la description.
- **L'architecture :** contient les instructions VHDL permettant de décrire et de réaliser le fonctionnement attendu. Elle représente la structure interne de la description.

Les entités et architectures sont des unités de conception dites primaires et secondaires. Un couple entité-architecture donne une description complète d'un élément ; ce couple est appelé modèle. Il prend le nom de l'entité ; Figure V.11.

Figure V.11 Modèle d'une description VHDL

Chaque modèle représente une description VHDL propre. Description VHDL d'un additionneur.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

ENTITY ADDITIONEUR IS
 port (
        A, B: in std_logic_vector (7 downto 0);
           S: out std_logic_vector (7 downto 0)
        );
END ADDITIONEUR;

ARCHITECTURE DESCRIPTION OF ADDITIONEUR1 IS
BEGIN
        S <= A + B;
END DESCRIPTION;
```

Declaration des bibliothèques

Declaration de l'entité

Fin de la description des signaux

Declaration de l'architecture

V.9.4 Les deux modes de travail en VHDL [43] [46] [47]

Le VHDL utilise deux modes de fonctionnement : le mode combinatoire (ou concurrent) et le mode séquentiel. Chacun de ces modes est utilisé dans un cas précis.

V.9.4.1 Le mode combinatoire : (ou concurrent), toutes les instructions d'une description VHDL sont évaluées et affectent les signaux de sortie en même temps, l'ordre dans lequel les instructions sont écrites n'a donc aucune importance. En effet la description génère des structures électroniques, c'est la grande différence entre une description VHDL et un langage informatique classique.

V.9.4.2 Le mode séquentiel : utilise les process dans lesquels le temps est une variable essentielle. Un process est une partie de la description d'un circuit dans lequel les instructions sont exécutées séquentiellement, c'est-à-dire les unes à la suite des autres. IL permet d'effectuer des opérations sur les signaux en utilisant les instructions standard de la programmation structurée comme dans les systèmes à base de microprocesseurs.

V.10 Architecture implémentée

Avant de passer à l'implémentation de notre architecture du classificateur des rythme cérébraux alpha, bêta, et thêta. Nous rappelons que pendant l'étude software, nous avons obtenue les poids synaptiques et les biais optimaux pour cette implémentation (off chip training). Pour cela nous n'avons besoin d'implémenter que le module Feed_Forward de l'algorithme de rétro propagation du gradient RPG qui présente la topographie (3 13 10 9 3 1) que nous avons obtenue pendant la phase d'apprentissage Figure V.12.

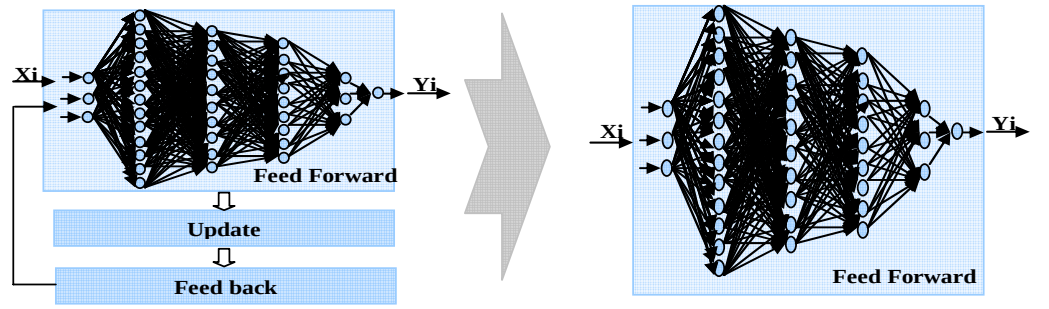


Figure V.12 description de la méthodologie de travail

Dans cette partie, nous présentons l'architecture implémentée sous forme bloc diagramme.

Cette présentation regroupe tous les blocs diagrammes des différents composants élémentaires dans la construction de l'architecture du classificateur, à noter le neurone, les différentes couches ainsi que l'architecture globale du classificateur.

V.11 Description du module Feed-forward

La partie feed-forward dans l'algorithme RPG est composée de deux unités, une unité opératrice et une autre de contrôle, Figure V.13.

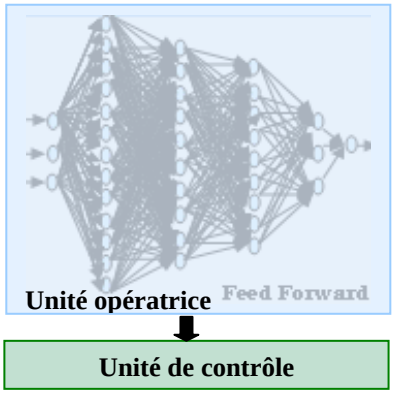


***Figure V.13description du module feed forward a* implémenté**

V.11.1 unité opératrice

Cette unité comporte l'architecture du réseau à implémenter (**3 13 10 9 3 1**). Le rôle de cette unité est de propager les exemples qui se présentent à l'entrée du réseau à travers les différentes couches jusqu'à la sortie de la partie Feed-forward.

Cette propagation est faite en calculant la valeur d'activation des neurones dans la couche d'entrée, qui sera transmise par la suite aux couches interne ou elle sera considérée comme un exemple pour ces couches. Les valeurs calculés à travers ces couches internes seront transmises en sortie à travers la dernière couche pour donner les sorties calculées du réseau. Le fonctionnement du module Feed_Forward dans notre application suit les points :

- La première couche d'entrée qui est composée de trois neurones, effectueront la somme pondérée des poids synaptiques et les valeurs d'entrées correspondants ainsi que l'addition des biais puit la génération des valeurs d'activations des neurones.
- Les couches cachées de différente taille, ont pour rôle de propager l'information à la dernière couche et pour séparer les classes non linéairement séparables dans cette classification.
- La troisième et la dernière couche effectue les opérations que la première, pour donner en fin la valeur finale du calcule effectuer par le module Feed_Forward.

V.11.1.1 Le neurone

Comme nous l'avons déjà dit, le neurone est l'élément de base dans cette conception. Le neurone est constitué de: Figure V.14

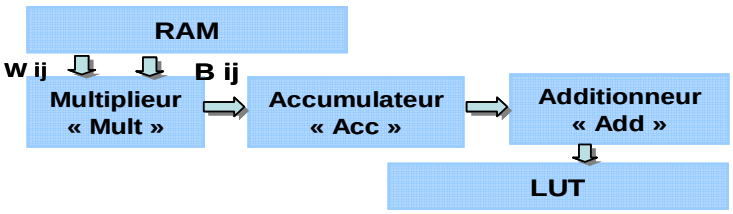


Figure V.14: architecture du neurone

- **La ROM :** utilisée pour le stockage des poids synaptiques et biais obtenus dans l'étude software. Dans cette **ROM**, les valeurs des poids synaptiques et les biais stockés sont destinées à sa sortie, après qu'elles soient sélectionnées par une entrée spécifique appelé « **addr** », ces dernières sont injectées par la suite dans le multiplieur.
- **Le multiplieur «MULT » :** il sert à faire la multiplication des poids synaptiques avec les valeurs des exemples correspondants qui sont présentés à son entrée. Cette opération est nécessaire pour le calcul de la somme pondérée. Les valeurs calculées dans ce multiplieur sont transmises de sa sortie vers l'entrée de l'accumulateur.
- **L'accumulateur « ACC » :** c'est l'élément qui effectue la somme pondérée des valeurs venant, après calcul, du multiplieur. L'accumulateur se compose d'un additionneur et d'un registre. L'opération de l'accumulation se fait par le chargement de la première valeur venant du multiplieur dans le registre puis l'additionner avec les données reçues du multiplieur. La description de cet composant est décrite en **VHDL**.
- **L'additionneur « ADD » :** cet élément reçoit à son entrée la valeur finale calculée au niveau de l'accumulateur. Il l'additionne à une constante interne qui est le biais correspondant au neurone en question, puis il transmet la

valeur calculée à l'élément suivant qui est la « **LUT** » ou la fonction d'activation.

- **La fonction d'activation « LUT »** Figure (V.15)**:** Le rôle de la fonction d'activation est de prendre la valeur du biais rajoutée à la somme pondérée calculée auparavant, et de lui appliquer la fonction sigmoïde, pour générer la valeur d'activation du neurone. Pour notre application nous avons utilisé la fonction sigmoïde définie comme suit :

$$yi = 2/ (1+\exp (-2*xi))-1 \qquad \text{(IV.1).}$$

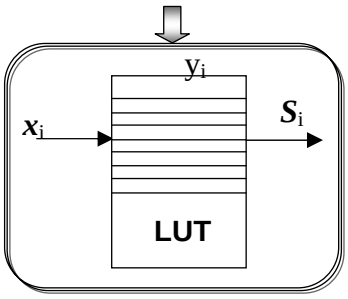


Figure V.15 Modélisation de la fonction d'activation

La modélisation de cette fonction nécessite l'implémentation des opérations de division et d'exponentielle, chacun de ces opérateurs nécessite un nombre important de ressources du circuit **FPGA**. Pour remédier à ce problème, on utilise les Look-Up-Tables (**LUTs**) du circuit FPGA pour la modélisation de cette fonction. Dans notre cas, nous utilisons ces **LUTs** en **ROM** adressée par la valeur de la somme pondérée et les valeurs d'activation seront chargées dans cette **ROM**, donc on aura la valeur de la somme pondérée en entrée et celle de l'activation en sortie.

Le neurone Figure V.14 ;du module Feed-forward précédemment décrit constitue la modélisation de la multiplication synaptique et l'accumulation afin de trouver la valeur de la somme pondérée, nous avons utilisé le multiplieur et l'accumulateur pour calculer cette valeur et le registre pour la stocker, puis nous avons utilisé l'additionneur pour ajouter à cette valeur la valeur du biais, Une fois cette opération est effectuée, nous utilisons la fonction d'activation pour le calcul de la valeur d'activation du neurone.

V.11.1.2 la couche d'entrée

La couche d'entrée dans cette architecture est composée d'un bloc principal composé lui même de trois neurones associes a trois Demultiplexeurs, un compteur qui sert a incrémenter l'adresse des donnée dans la mémoire et un bloc de contrôle qui synchronise et automatise le fonctionnement des différents blocs dans cette couche, Figure V.15. La simulation de ces éléments est présentée plus loin dans ce mémoire dans les prochains paragraphes.

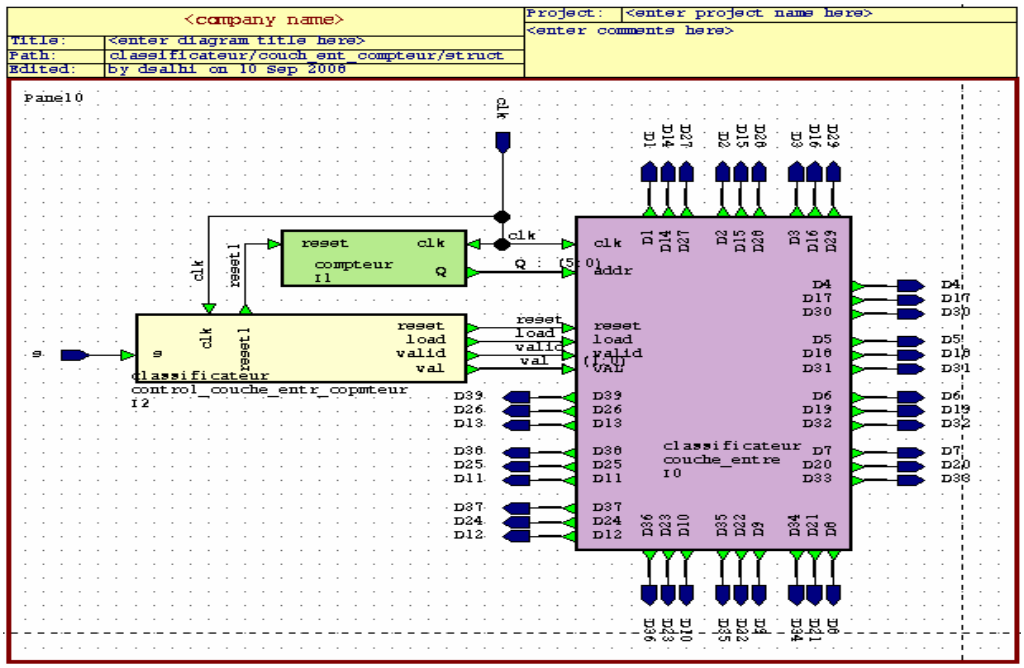


Figure V.15 aspect de la couche d'entrée

V.11.1.3 la première couche cachée

La première couche cachée représente une architecture composée d'un bloc opérateur qui contient un ensemble de 13 neurones associes a 13 multiplexeurs et 13 démultiplexeurs, une mémoire ROM qui contient les poids synaptique correspondants aux neurones dans cette couche et un bloc de contrôle qui synchronise et automatise le fonctionnement de cette couche, Figure V.16.

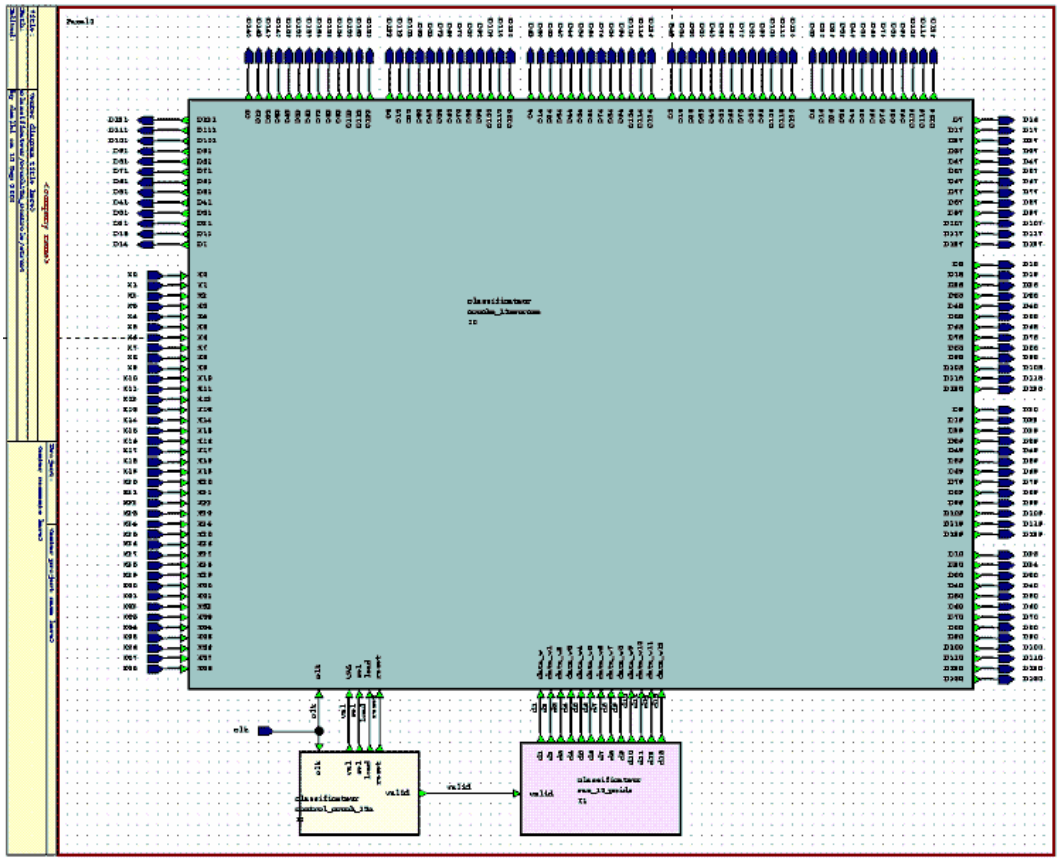


Figure V16 aspect de la première couche cachée

V.11.1.4 la deuxième couche cachée

Elle présente le même aspect que la précédente couche sauf que, la partie opératrice ici est composée de 10 neurones, 10 multiplexeurs et 10 Demultiplexeurs. Figure V.17.

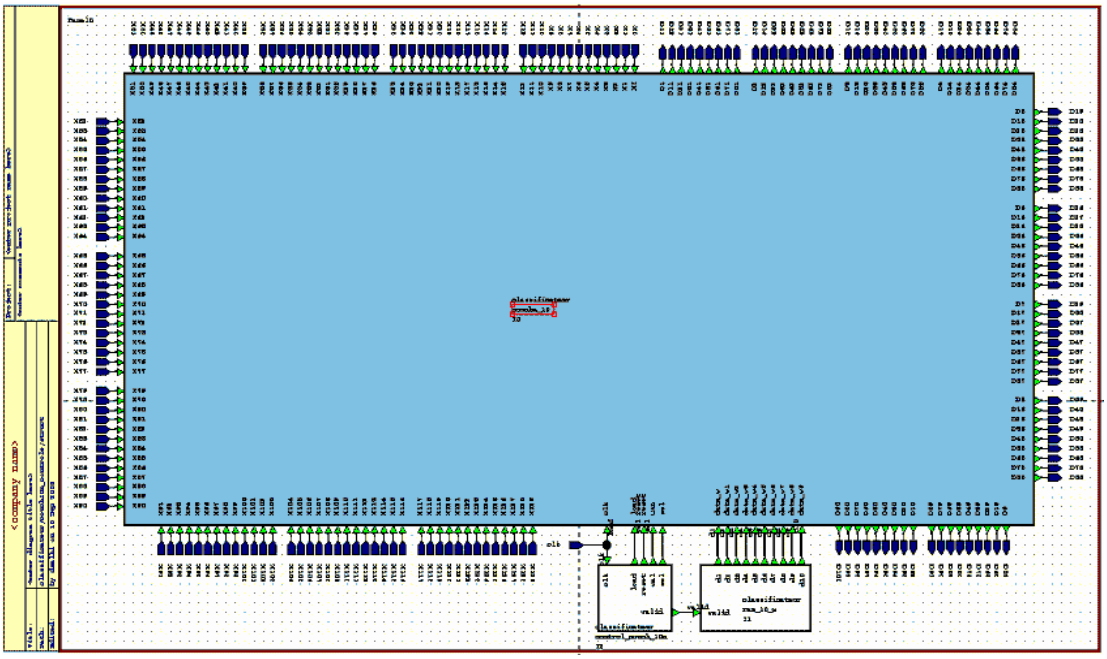


Figure V.17 aspect de la deuxième couche cachée

V.11.1.5 la troisième couche cachée

Elle présente le même aspect que les deux couches précédents, on distinguant toujours le bloc opératoire qui contient ici 9 neurones, 9 multiplexeurs et 9 demultiplexeurs. Figure V.18.

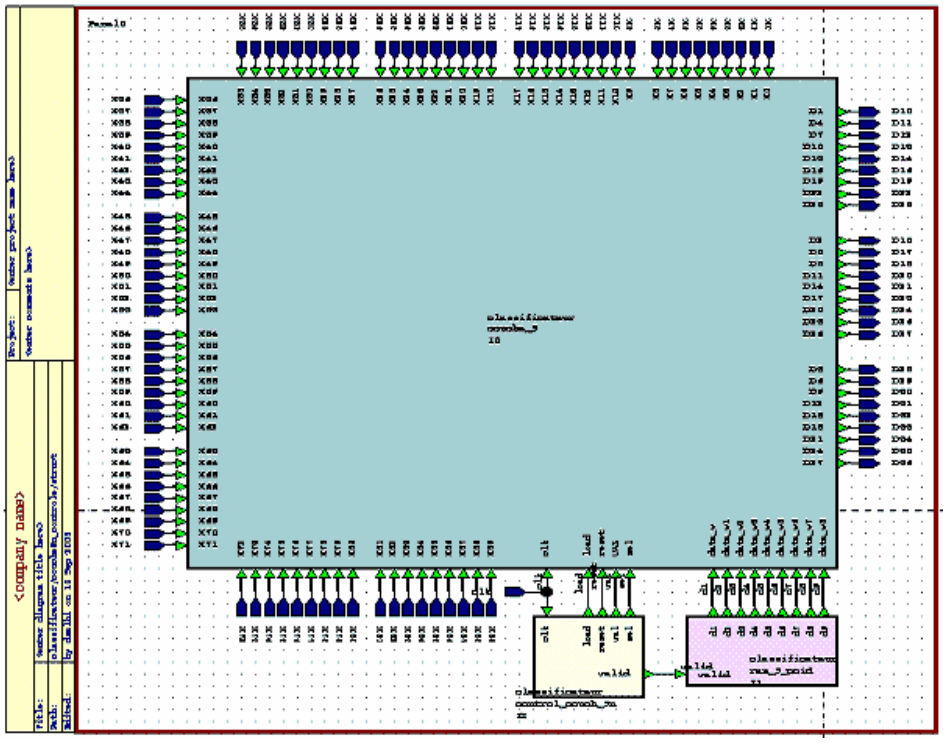


Figure V.18 aspect de la troisième couche cachée

V.11.1.6 la quatrième couche cachée

Cette couche représente la dernière couche cachée dans cette architecture, elle contient un bloc opérateurs composé de trois neurones et trois Demultiplexeurs, une mémoire rom qui stock les poids synaptique correspondants aux neurones de cette couche et un bloc de contrôle pour la synchronisation de son fonctionnement Figure V.19.

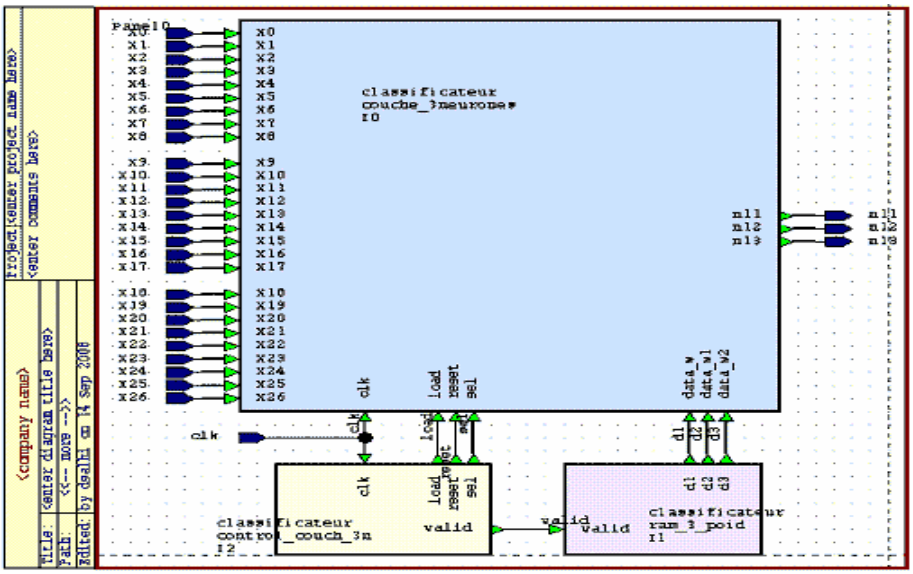


Figure V.19 aspect de la quatrième couche cachée

V.11.1.7 la couche de sortie

La couche de sortie c'est la dernière couche dans l'architecture, elle contient un neurone, une mémoire ROM et un bloc de contrôle pour la synchronisation du fonctionnement de cette couche. Figure V.20.

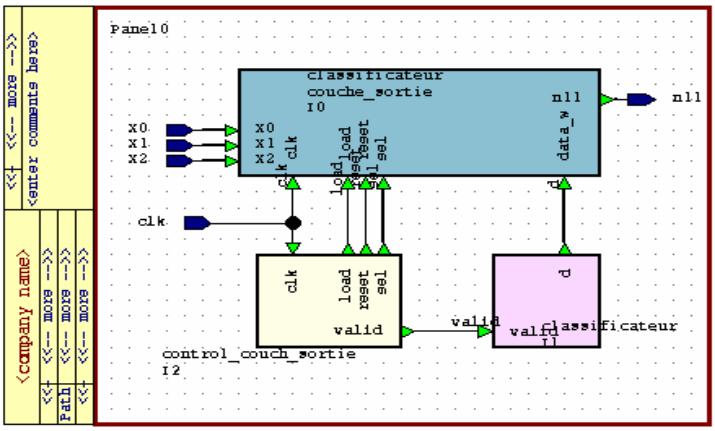


Figure V.20 aspect de la couche de sortie

Le module Feed forward qui constitue notre classificateur est composé de l'ensemble des différentes couches présentées précédemment. Il est important de rappeler que l'unité de contrôle du module Feed forward désigne l'ensemble des différents blocs de contrôle pour chaque couche dans cette architecture. Figure V.21.cette architecture qui assure les différent étapes de calcule de la somme pondérée, l'addition, la fonction d'activation et la propagation de l'information vers la couche de sortie, toute on effectuant la classification.

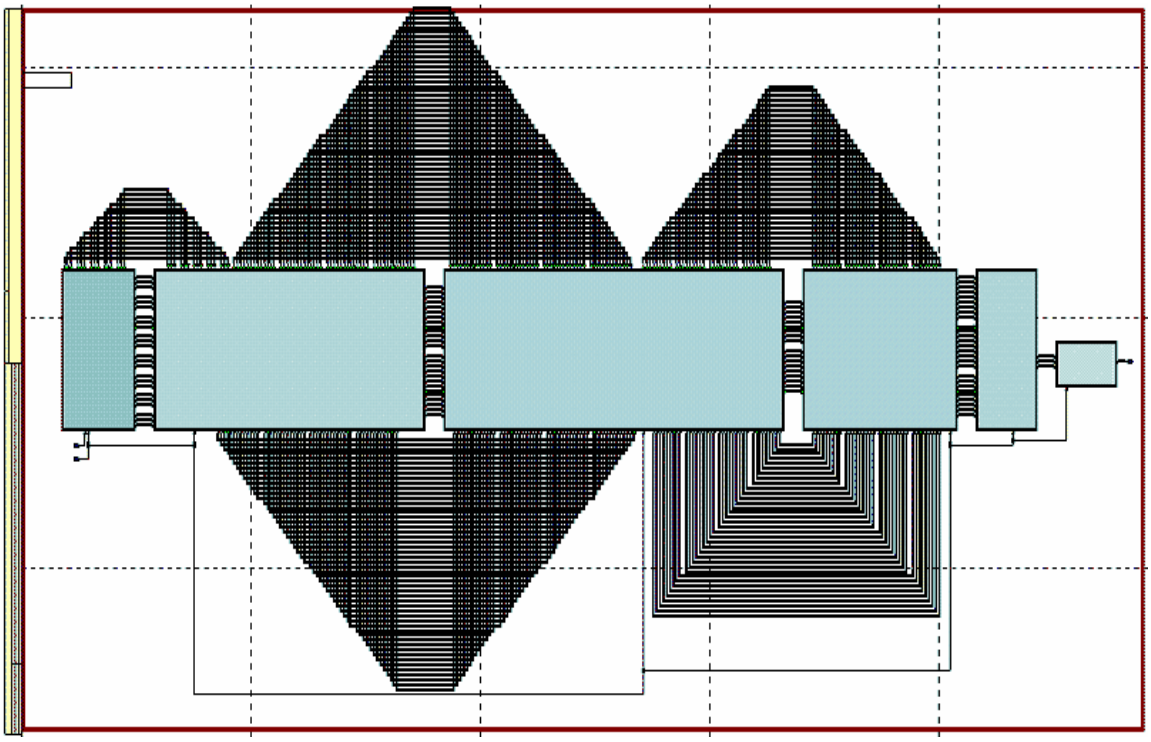


Figure V.21 l'architecture du module Feed Forward

V.11.2 Unité de contrôle

C'est une unité sous forme de diagramme d'état (de type « Moor »), qui est utilisée pour contrôler l'exécution des différentes étapes du calcul du neurone ainsi que celles des différentes couches en synchronisant les différentes composants dans le neurone pour que l'exécution se termine convenablement. Par analogie à la structure du module Feed-forward, on a généré un diagramme de contrôle composé de trois phases de contrôle : contrôle du neurone, contrôle de la couche et le contrôle du module Feed_Forward. On rappelle que Dans une machine de « Moor », les sorties ne varient que lorsque le système change d'état (Les sorties ne sont fonction que des états et non pas des états et des données d'entrée). Par contre les machines de « Mealy » produisent des sorties après la réception de données d'entrée. Dans ce qui suit, nous décrivons le control du neurone.

V.11.2.1 Control du neurone

Le control du neurone suit les étapes illustrées dans l'organigramme de la Figure V.22.

Cet organigramme montre que le fonctionnement du diagramme d'état de la partie Contrôle, passe par cinq phases qui sont : phase du départ, phase d'initialisation, phase de multiplication, d'accumulation et de stockage, phase d'addition et phase de génération de la fonction d'activation du neurone.

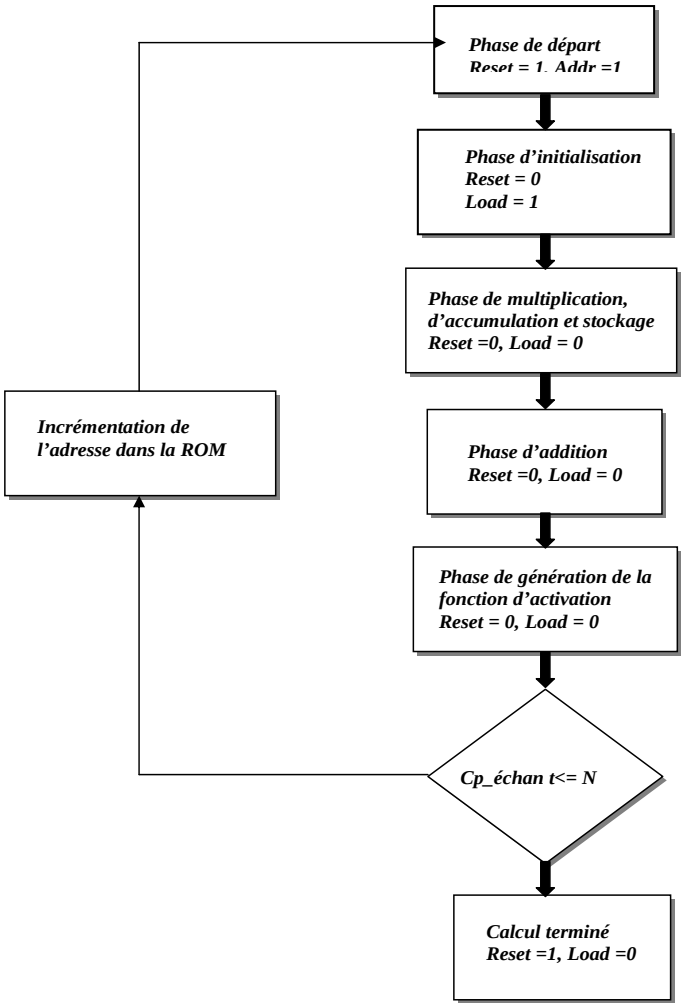


Figure V.22 Organigramme de la partie contrôle du neurone

- **Phase de départ :** Comme tous les diagrammes d'état dans l'implémentation des réseaux de neurones, ce diagramme commence par un état de **RESET** : un état durant lequel le système à contrôler reste en attente tant qu'il n'est pas en phase d'exécution, il est maintenu dans cet état par le contrôle global, de ce module. Il sert aussi à la remise à zéro des différentes variables pour entamer l'exécution d'une autre itération de ce module. Si cet état est activé par le contrôle global, le contrôle passe à la phase d'initialisation.
- **Phase d'initialisation :** Cette phase a pour rôle de charger les valeurs des poids synaptiques et des échantillons de l'exemple en question dans la ROM et d'initialiser l'accumulateur en mettant le **RESET** à la valeur **"0"** et le **LOAD** à **"1"**. Une fois cette phase terminée, on passe à la multiplication synaptique et l'accumulation.
- **Phase de la multiplication, d'accumulation et de stockage :** Cette phase est responsable du contrôle de la partie du neurone qui calcule la somme pondérée. On commence d'abord par le chargement de la valeur du poids synaptique dans le multiplieur. Après la multiplication, l'accumulateur charge cette valeur en mettant le **LOAD** à la valeur **"1"**. Une fois cette opération terminée l'adresse dans la « **ROM** » est incrémentée a l'aide d'un compteur associé à la **ROM**, et la deuxième

valeur est chargée dans le multiplieur, mais cette fois le **LOAD** est mis à zéro pour que cette valeur soit accumulée avec celle qui la précède. Ce processus est répété **N** fois (**N** : nombre des échantillons dans l'exemple). Après avoir calculé la somme pondérée, elle est stockée dans le registre dans l'accumulateur.

- **Phase d'addition :** une fois le **LOAD** à « **0** » et le **RESET** est à « **1** », l'additionneur reçoit à son entrée la dernière valeur calculée par l'accumulateur. Cette valeur est ajoutée au biais déclaré comme étant une constante à l'intérieur de l'additionneur.
- **Phase de génération de la fonction d'activation :** après avoir ajouté le biais à la somme pondérée, la valeur résultante est transmise à la « **LUT** », pour générer la fonction d'activation. Une fois que cette opération terminée, le **LOAD** est remis à « **1** » pour obtenir à la fin la combinaison (**RESET=1** et **LOAD=1**) pour remettre le tout à « **0** ».

Dans le contrôle de cette architecture, nous avons opté pour un contrôle indépendant pour chaque couche a fin de facilité le suivi du fonctionnement de cette dernière.

V.11.2.2 contrôle de la couche d'entrée

Le contrôle de la couche d'entrée repose sur le contrôle des différents pins de commande propre a chaque bloc constituant cette couche, telle que le pin « val », pour le démultiplexeur, « valid » pour la mémoire ROM, « load » et « reset » pour le neurone et « reset 1» pour la remise a zéro du compteur. Le fonctionnement de ce bloc de contrôle est cadencé par le pin « clk » d'horloge. Figure V.23 montre le diagramme d'états finis et les étapes de simulation du bloc de contrôle de cette couche.

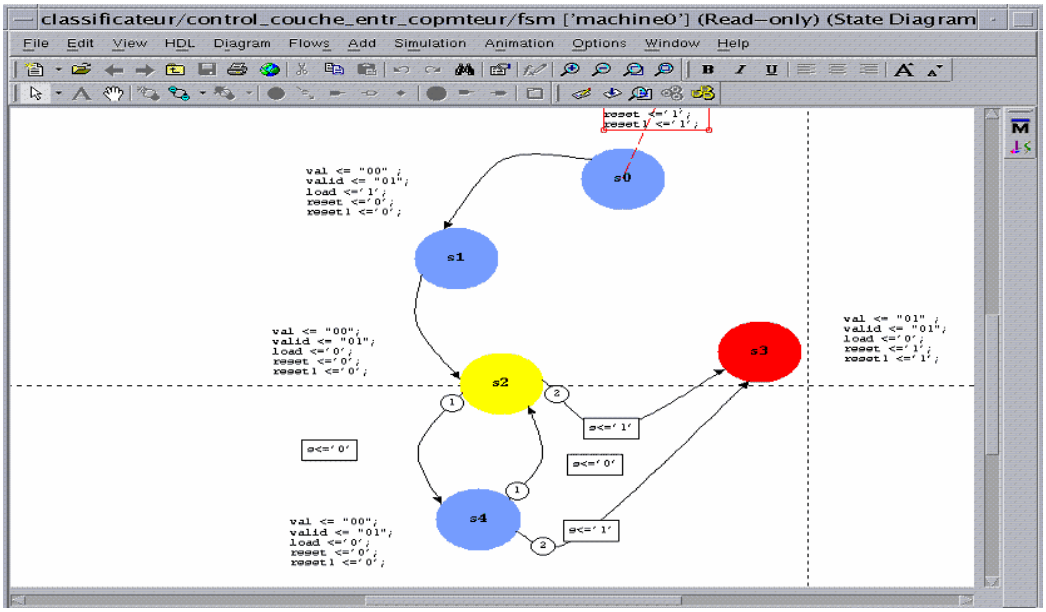


Figure V.23 diagramme d'états du contrôle de la couche d'entrée

V.11.2.3 contrôle de la première couche cachée

Dans cette couche on agis toujours sur les pins de commande a noté « val » pour le démultiplexeur, « sel » pour le démultiplexeur, « load et reset » pour le neurone, en commençons par un état initiale, suivi par une séquence d'états qui assure que cette

couche effectue correctement le calcule a ce niveau, le nombre d'état dans cette séquence correspond nombre de données reçue de la couche précédente plus un pour l'étape de l'accumulation effectué au niveau de l'accumulateur. Figure V.24.

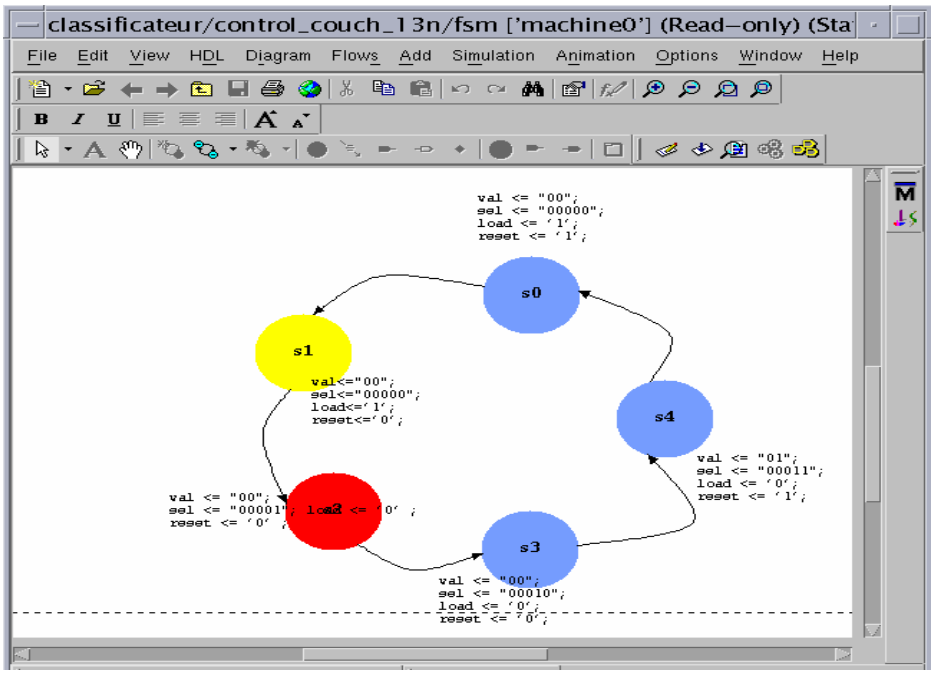


Figure V.24 diagramme d'états fini de la partie contrôle de la 1ère couche cachée

Il est a noté que toutes les couches cachées sont contrôlées avec le même principe que celle ci sauf que le nombre de séquences change d'une couche à une autre selon le nombre de donnée que reçois chacune d'elles par rapport a la précédente.

V.11.2.4 contrôle de la deuxième couche cachée

La figure V.24 présente le diagramme des états finis ainsi que les séquences de simulation du fonctionnement du bloc de contrôle de la deuxième couche cachée.

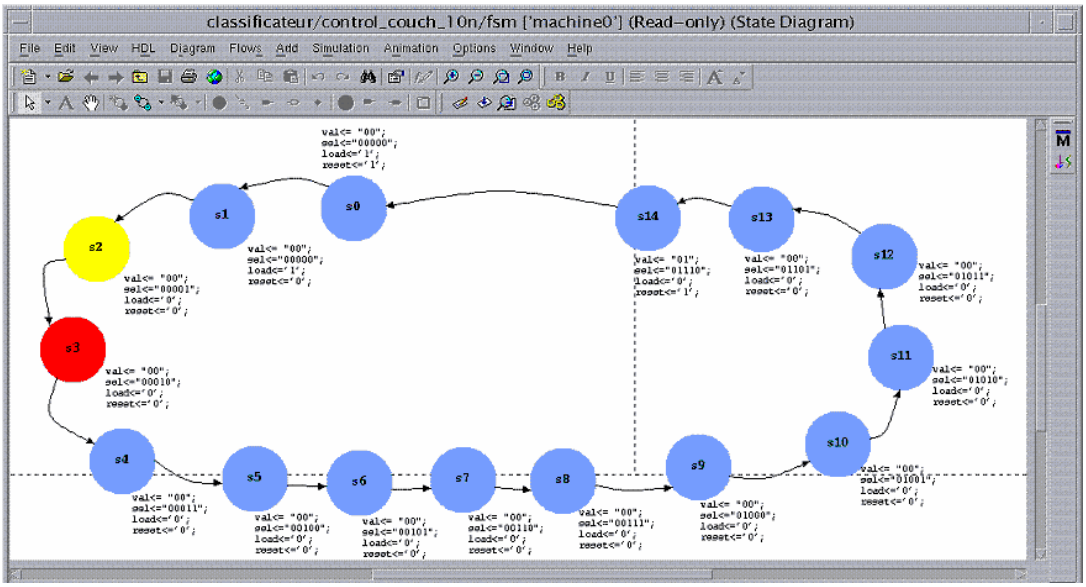


Figure V.24 diagramme d'états fini de la partie contrôle de la 2ième couche cachée

V.11.2.5 contrôle de la troisième couche cachée

La figure V.25 présente le diagramme des états finis ainsi que les séquences de simulation du fonctionnement du bloc de contrôle de la troisième couche cachée.

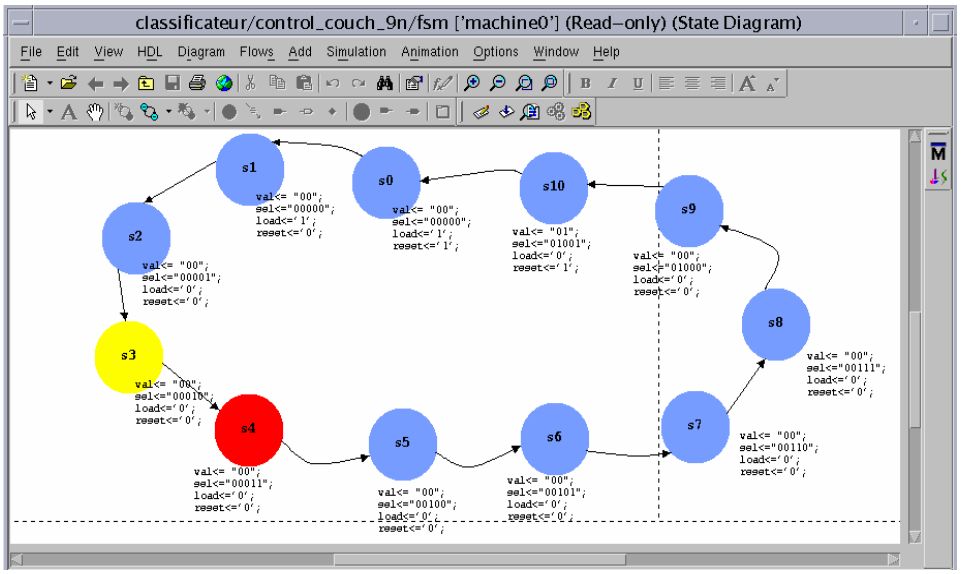


Figure V.25 diagramme d'états fini de la partie contrôle de la 3^{ième} couche cachée

V.11.2.6 contrôle de la quatrième couche cachée

La figure V.26 présente le diagramme des états finis ainsi que les séquences de simulation du fonctionnement du bloc de contrôle de la quatrième et la dernière couche cachée.

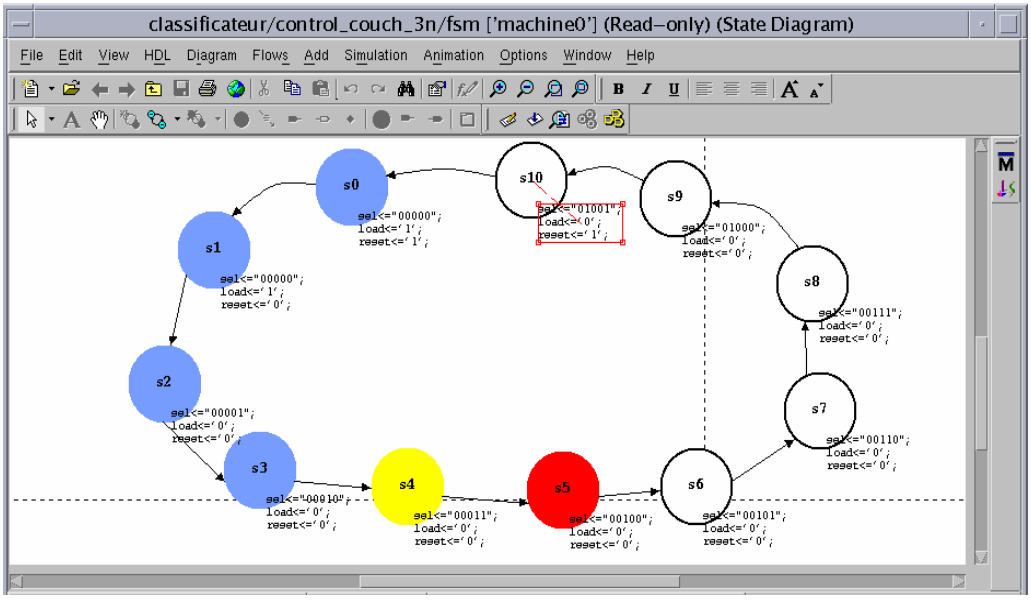


Figure V.26 diagramme d'états fini de la partie contrôle de la 4^{ième} couche cachée

V.11.2.7 contrôle de la couche de sortie

La couche de sortie est constituée d'un neurone avec la mémoire de poids synaptique propre a lui, le contrôle ici est plus simple il est résumé dans le contrôle du neurone. Figure V.27.

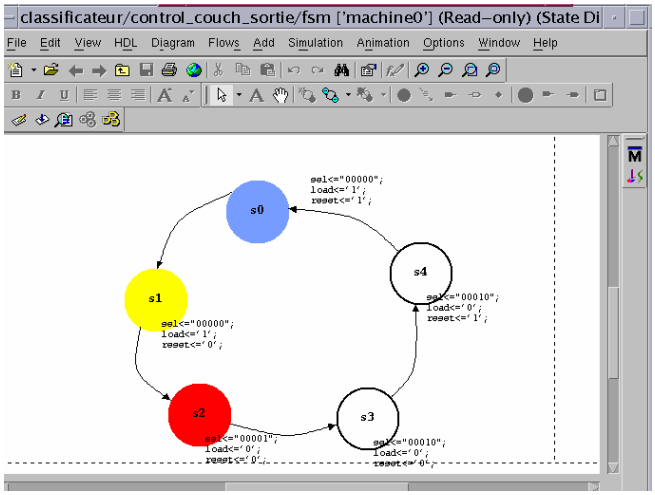


Figure V.27 diagramme d'états fini de la partie contrôle de la couche de sortie

La simulation des différents blocs de contrôles est présentée plus loin dans cette mémoire dans la partie simulation.

V.11.2.8 contrôle de l'architecture globale

Le contrôle du module feed forward est assurée par le contrôle des différentes couche commander par le pin « clk » de l'horloge globale de l'architecture et le pin « s » qui commande l'incrémentation du compteur dans la couche d'entée.

V.12 Simulation du module Feed-Forward

Dans cette partie, nous présentons les résultats de simulation des différents composants dans l'architecture du classificateur que nous avons implémenté, commençons par les blocs élémentaire constituent le neurone, passons par les différents couches et arrivons a la simulation de l'architecture globale du module Feed_Forward.

V.12.1 codage des données

Les résultats obtenus dans l'étude software moyennant l'outil « **MATLAB** », ont une précision de six (6) chiffres après la virgule. Ces résultats qui sont les poids synaptiques et les biais, sont codés pour la conception hardware sur **24** bits, le **24^{ième}** bit est réservé pour le signe.

Le mapping de notre classificateur a été fait sur un circuit **FPGA**, plus exactement sur le circuit « **XC2V1000_4FG 456** » de la famille «**VIRTEX-II** ». Nous avons utilisé l'outil « **FPGA ADVANTAGVE**» de « **MENTOR GRAPHICS**» qui fonctionne sous environnement « **UNIX** ». L'outil « **FPGA ADVANTAGE** » intègre deux modules, un pour la synthèse qui est le « **LEONARDO SPECTROME** », et l'autre pour la simulation qui est le « **MODEL SIM** », pour le placement et le routage comme étant les phases finales dans l'implémentation ainsi que la génération du fichier .bit qui permis la configuration de l'FPGA ; sont effectués avec l'outil ISE Fondation [48] [49].

V.12.2 Simulation du neurone

Dans ce paragraphe, nous présentons les résultats de simulation de chaque composant du neurone jusqu'à arriver aux résultats de simulation du

neurone, et à la fin, le résultats de simulation du fonctionnement du module Feed_Forward.

Le neurone est composé de Figure V.28 une mémoire **ROM**, un multiplieur « **MULT** », un accumulateur « **ACC** », un additionneur « **ADD** » et un bloc de la fonction d'activation du neurone « **LUT** ».

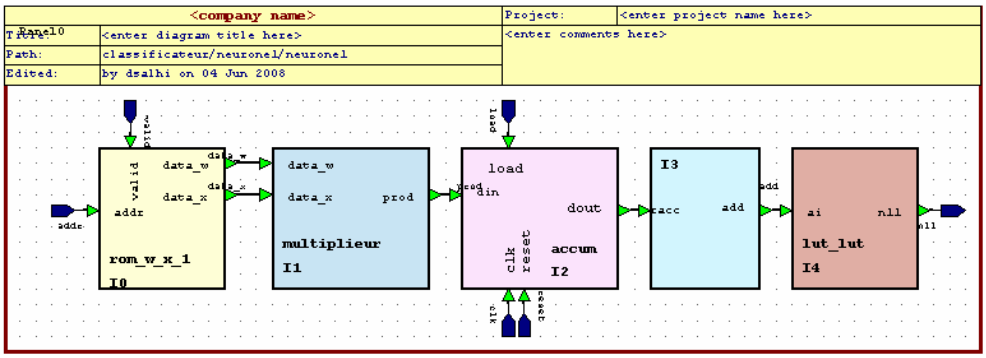


Figure V.28 Architecture du neurone

1. **La « ROM » :** c'est une mémoire de **256** mots de **8** bits d'adressage, elle possède deux entrées, chacune est codée sur **24** bits, un sélecteur d'adressage « **ADRR** »codé sur **6** bits incrémenté à l'aide d'un compteur associé à cette **ROM** et à la fin deux sorties codées sur **24** bits. Figure V.19

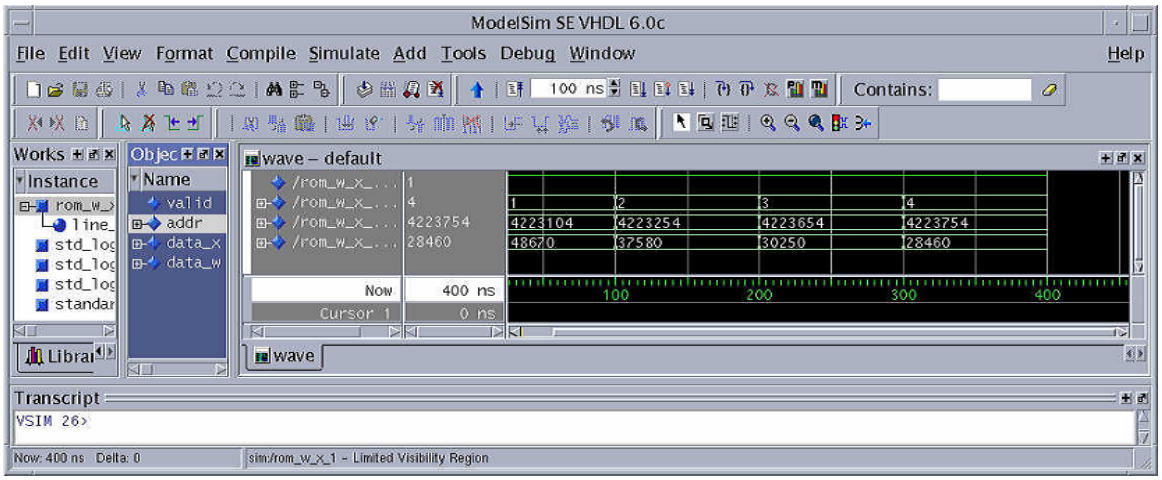
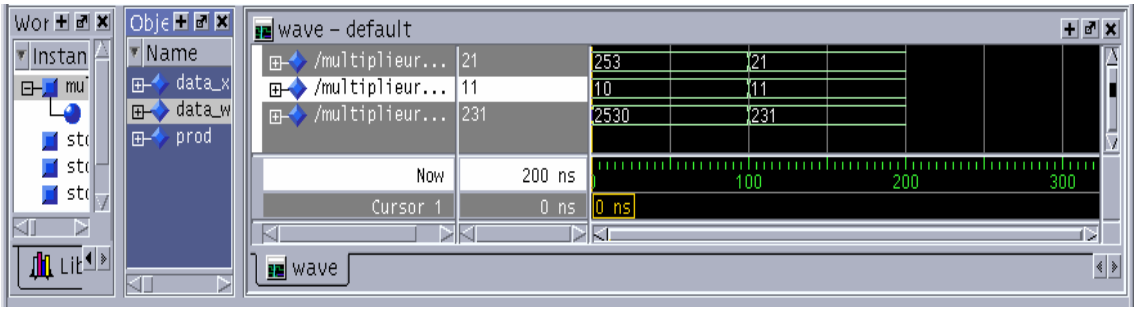


Figure V.29 Chronogramme de simulation de la ROM.

2. **le multiplieur « MULT » :** il est composé d'une entrée et une sortie, codées sur **24** bits. Figure V.20



FigureV.30 Chronogramme de simulation du multiplieur.

3. **L'accumulateur « ACC » :** il est composé d'une entrée, une sortie codées toutes les deux sur **24** bits, plus les signaux de commandes qui sont : l'horloge « **CLK** », la remise à zéro « **RESET** » et la commande de chargement « **LOAD** ». Figure V.21.

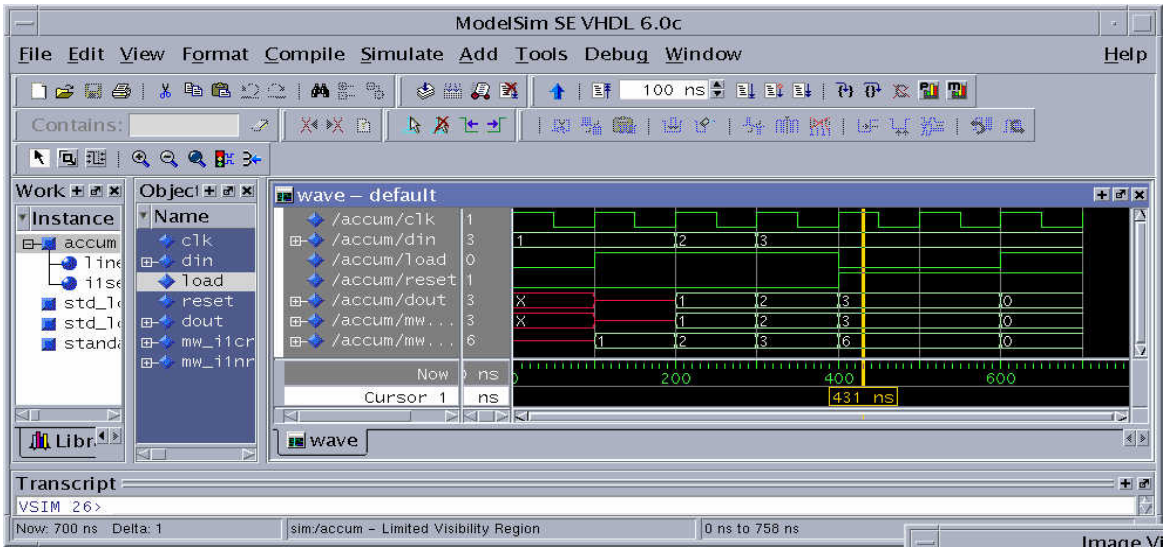


Figure V.31 Chronogramme de simulation de l'accumulateur

4. **L'additionneur « ADD » :** il possède une entrée et une sortie codées sur **24** bits. La valeur du biais est déclarée comme étant une constante car elle est fixe dans cette application. Figure V.22.

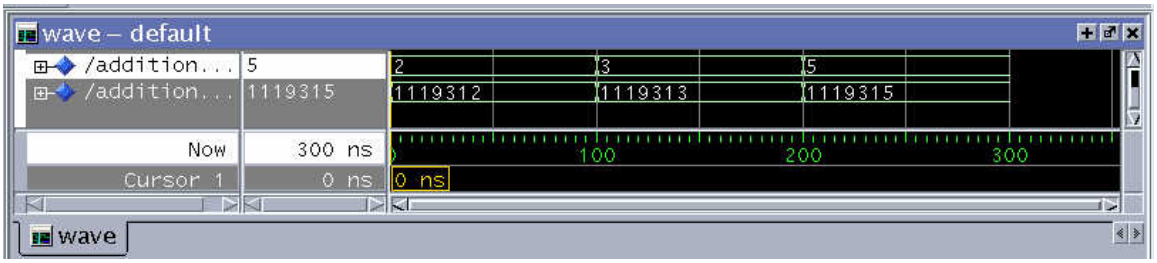


Figure V.32 Chronogramme de simulation de l'additionneur

5. **Le bloc de la fonction d'activation « LUT » :** il est composé d'une entrée et une sortie, codées toutes les deux sur **24** bits. Figure V.23

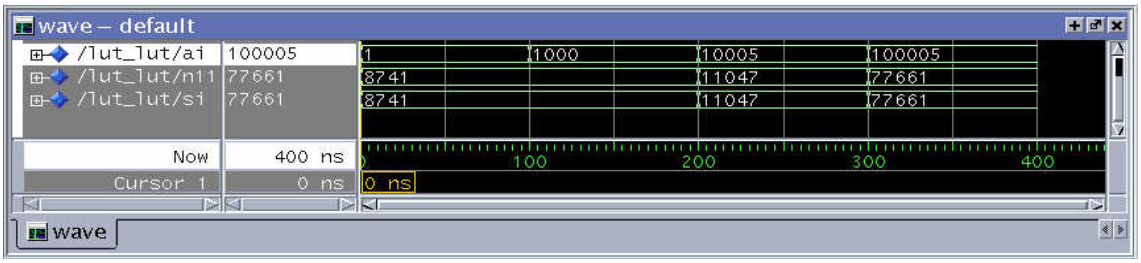


Figure V.33 Chronogramme de simulation de la « LUT »

Toutes les figures de la simulation précédente montrent le bon fonctionnement de tous les composants qui forment le neurone du module Feed_Forward, d'où la simulation de ce neurone donne le chronogramme de simulation du neurone sur Figure V.24.

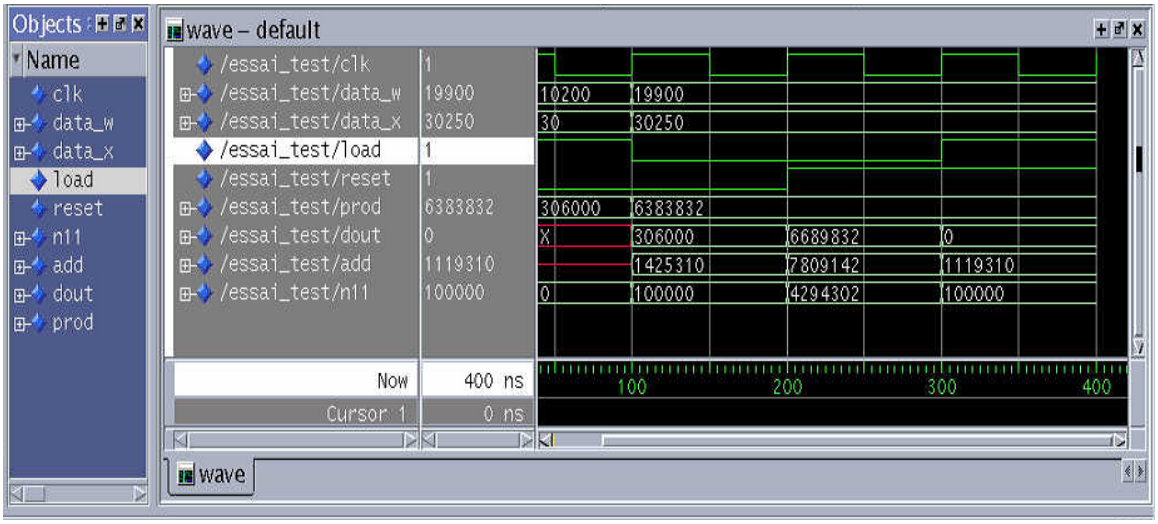


Figure V.34 Chronogramme de simulation du neurone

V.12.3 simulation de la couche d'entrée :

La première couche est constituée de trois neurones et trois demultiplexeurs. Il est a rappeler aussi que dans cette couche l'incrémentation de l'adresse des données dans la mémoire et assurée par un compteur. Figure V.25.

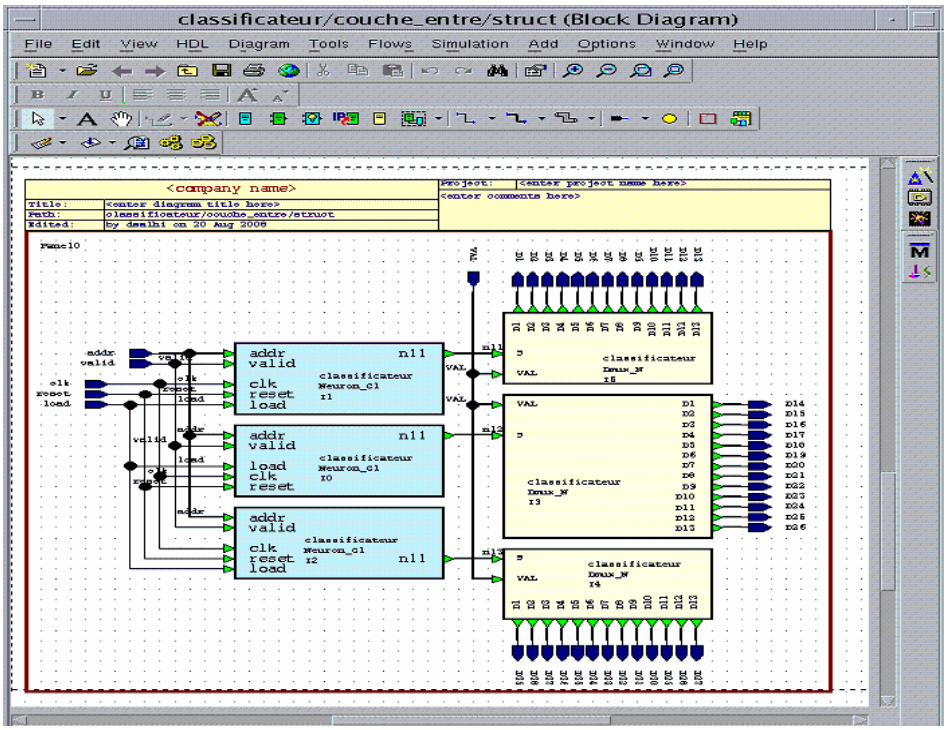


Figure V.35 composition interne de la couche d'entrée

V.12.3.1 Le Demultiplexeur « Dmux » : ce démultiplexeur est commandé par le pin de commande « Val », lorsque « val » est active le « Dmux » envoie a ça sortie plusieurs copies de la donnée qu'il a reçue a l'entrée. Figure V.36.

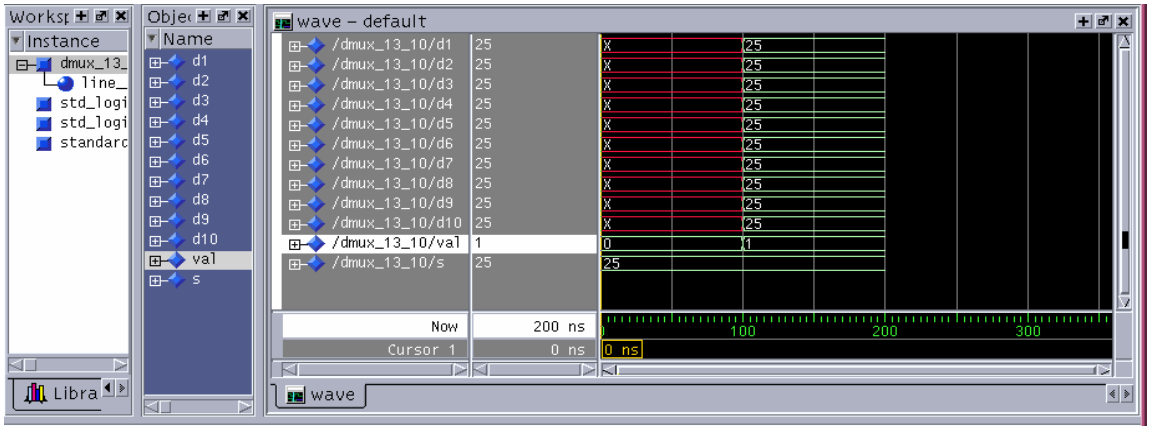


Figure V.36 chronogramme de la simulation du demultiplexeur

V.12.3.2 le compteur

Cet élément est responsable de l'incrémentation de l'adresse dans la mémoire qui stock les poids synaptique et les valeurs des échantillons dans la première couche. Figure V.37 présente le chronogramme de fonctionnement du compteur.

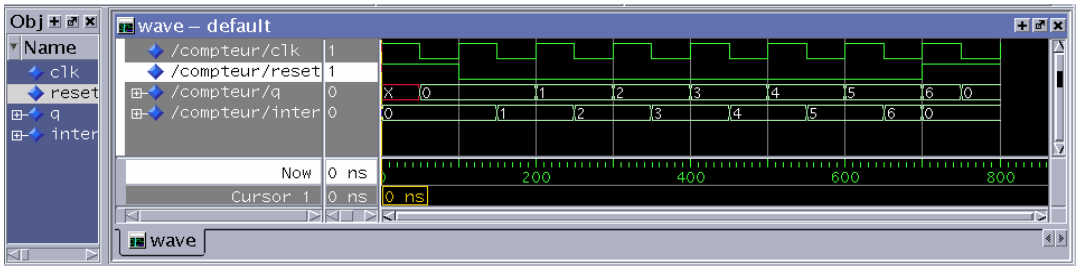


Figure V.37 chronogramme de simulation du compteur

Le fonctionnement de la première couche est présenté par le chronogramme de simulation sous Modelsim. Figure V.38

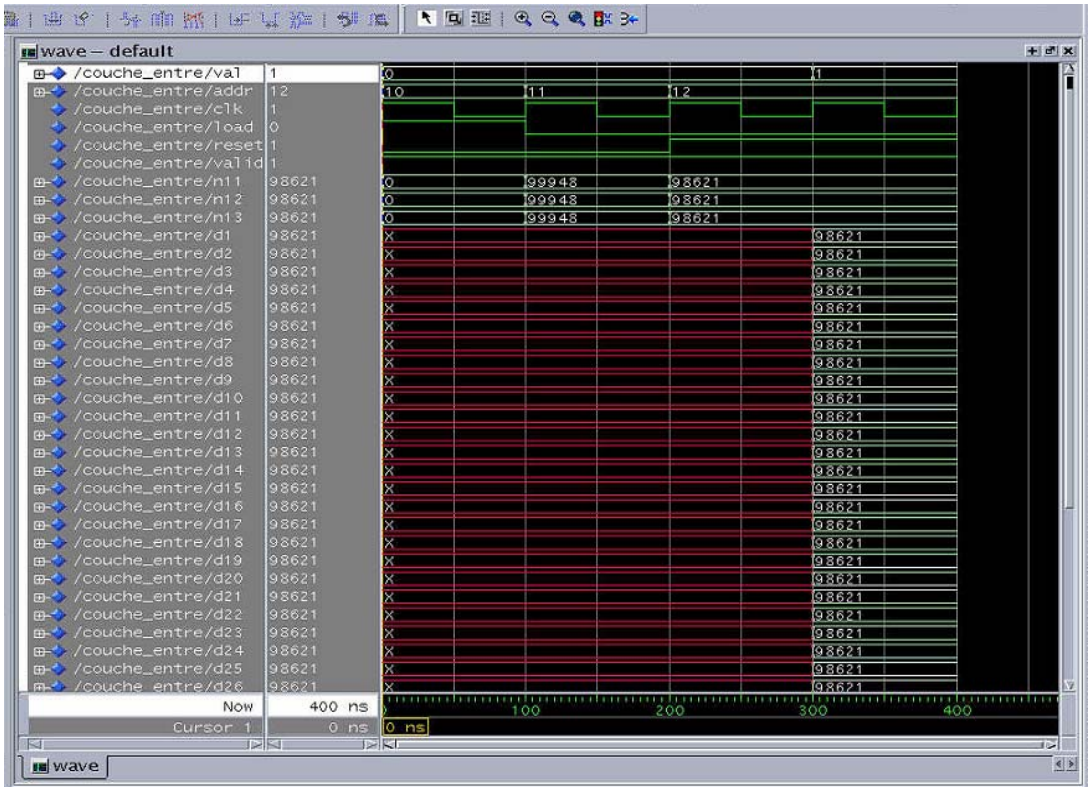


Figure V.38 chronogramme de simulation de la couche d'entrée

V.12.4 simulation de la première couche cachée (13 neurones)

Cette dernière est composée de 13 neurones précédés par 13 multiplexeurs « Mux » et suivis par 13 démultiplexeurs « Dmux », Figure V.39.

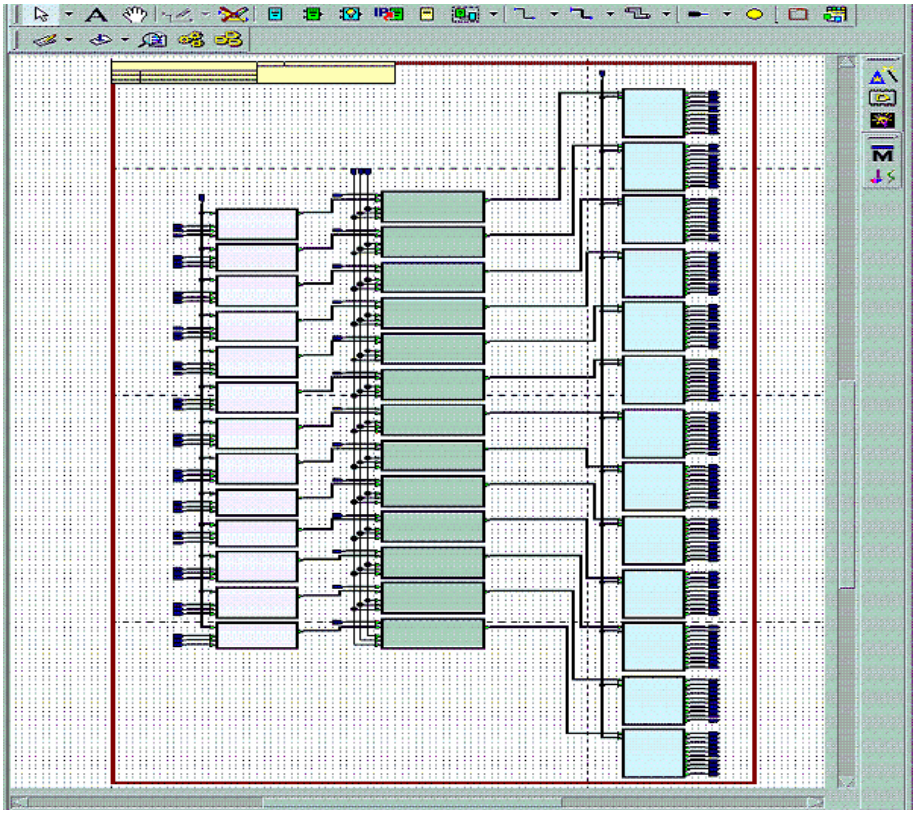


Figure V.39 structure interne de la première couche cachée

V.12.4.1 Le multiplexeur « Mux »

Le multiplexeur Mux reçois a son entrée des données en parallèle qu'il transmet a ça sortie séquentiellement .Figure V.40.

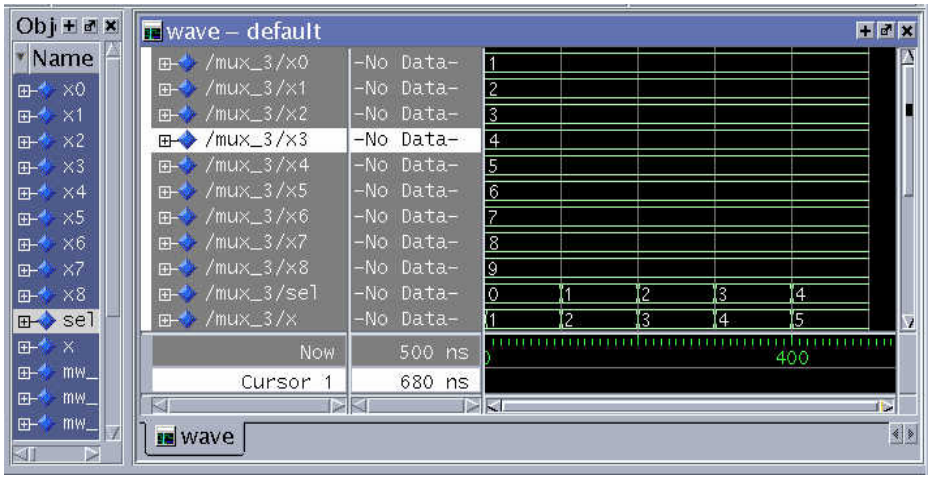


Figure V.40 chronogramme de simulation du multiplexeur Mux

V.12.4.2 simulation du bloc de contrôle

Figure V.41 présente le chronogramme de fonctionnement du bloc de contrôle de la couche d'entrée.

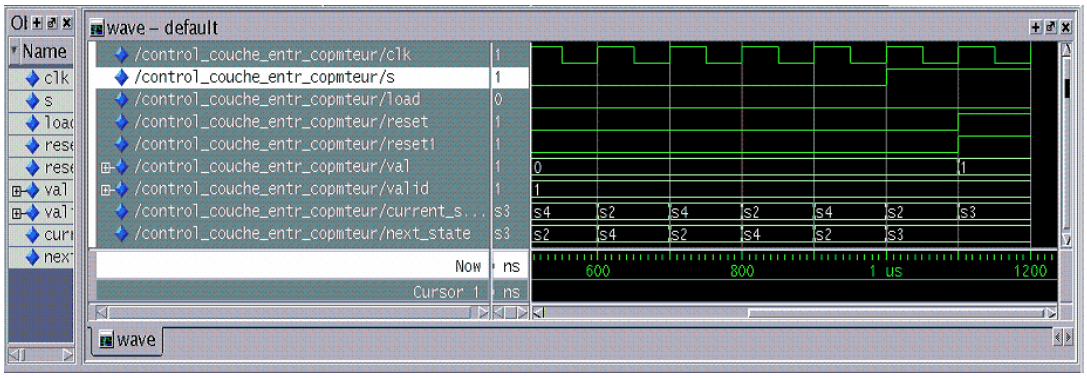


Figure V.41 chronogramme de simulation du bloc de contrôle de la couche d'entrée

Le fonctionnement de la première couche cachée est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.41.

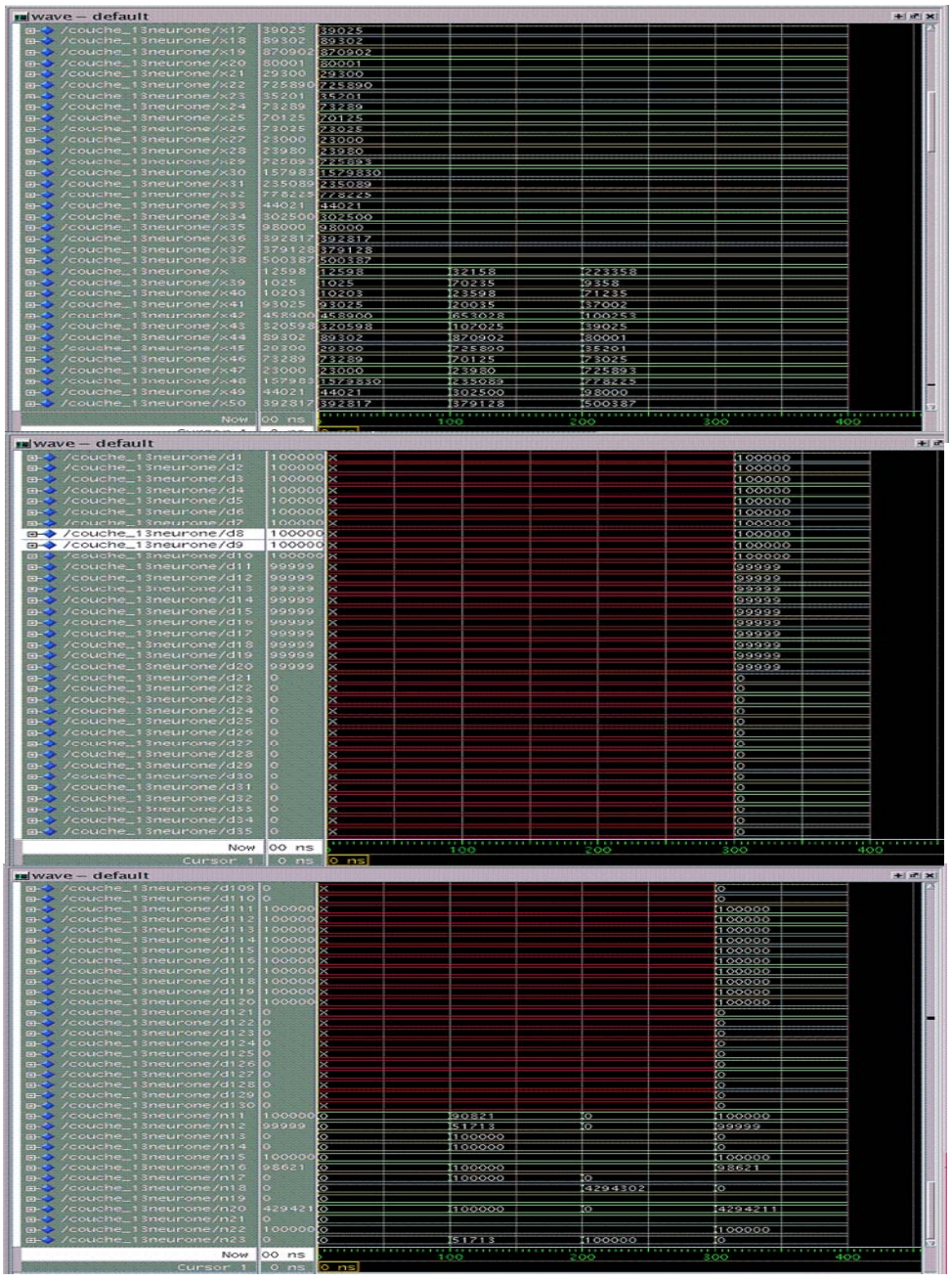


Figure V.41 chronogramme de simulation de la 1ère couche cachée

V.12.5 simulation de la deuxième couche cachée (10 neurones)

Cette dernière est composée de 10 neurones précédés par 10 multiplexeurs « Mux » et suivis par 10 démultiplexeurs « Dmux », Figure V.42.

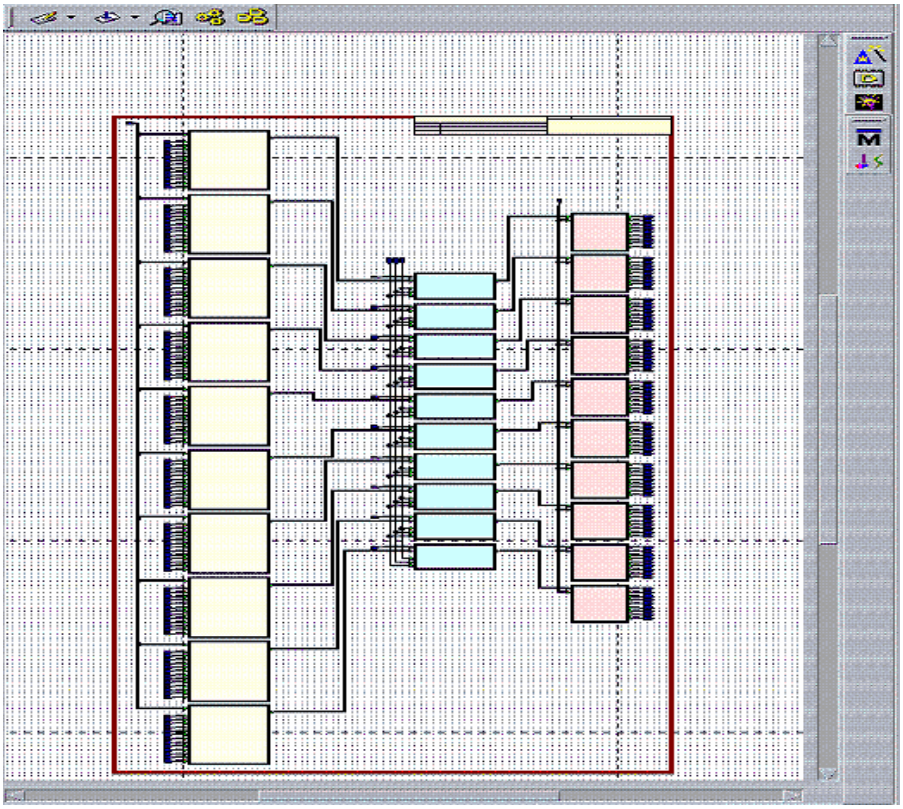


Figure V.42 structure interne de la deuxième couche cachée

V.12.5.1 simulation du bloc de contrôle

Figure V.43 présente le chronogramme de fonctionnement du bloc de contrôle de la deuxième couche cachée.

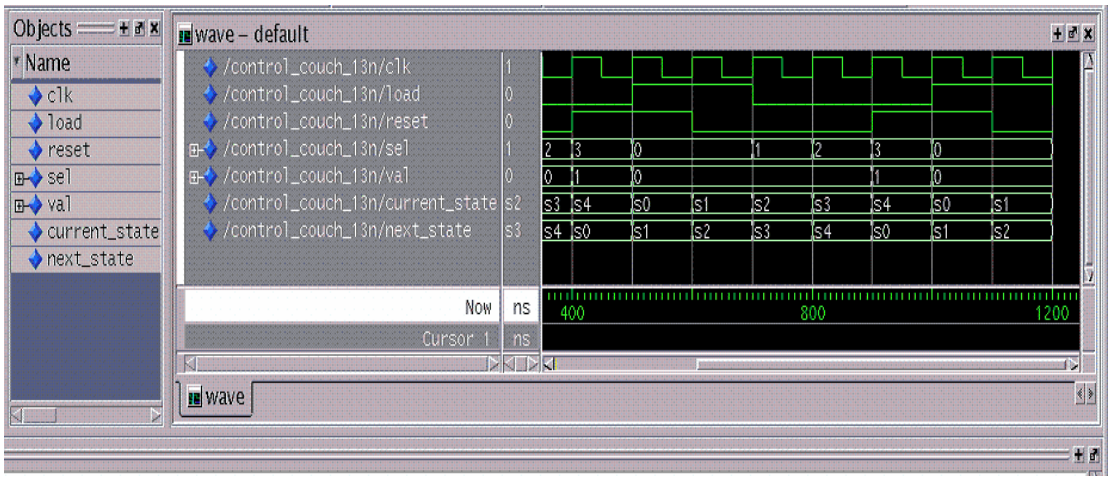


Figure V.43 chronogramme de simulation du bloc de contrôle de la deuxième couche cachée

Le fonctionnement de la deuxième couche cachée est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.44.

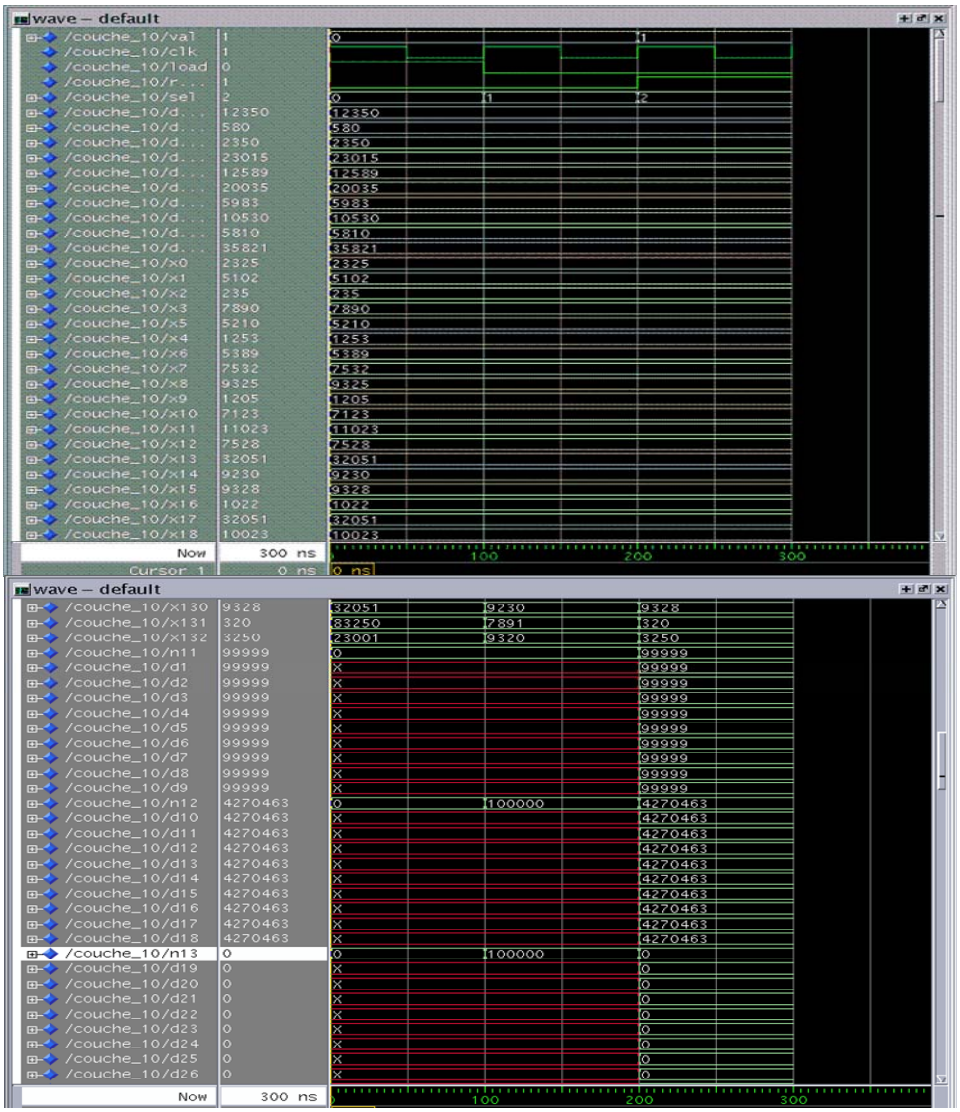


Figure V.44 chronogramme de simulation de la 2^{ième} couche cachée

V.12.6 simulation de la troisième couche cachée (9 neurones)

Cette dernière est composée de 9 neurones précédés par 9 multiplexeurs « Mux » et suivis par 9 démultiplexeurs « Dmux », Figure V.45.

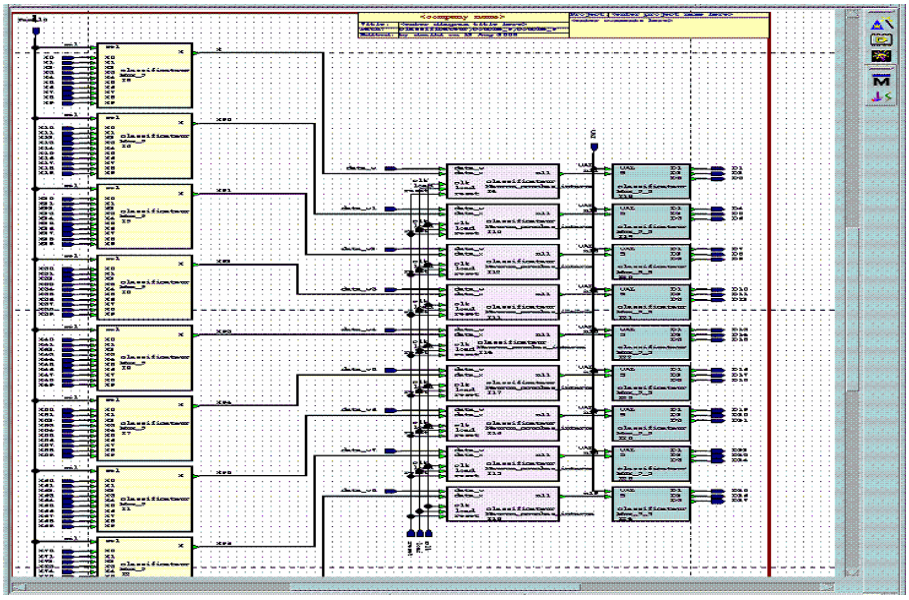


Figure V.45 structure interne de la troisième couche cachée

V.12.6.1 simulation du bloc de contrôle

Figure V.46 présente le chronogramme de fonctionnement du bloc de contrôle de la troisième couche cachée.

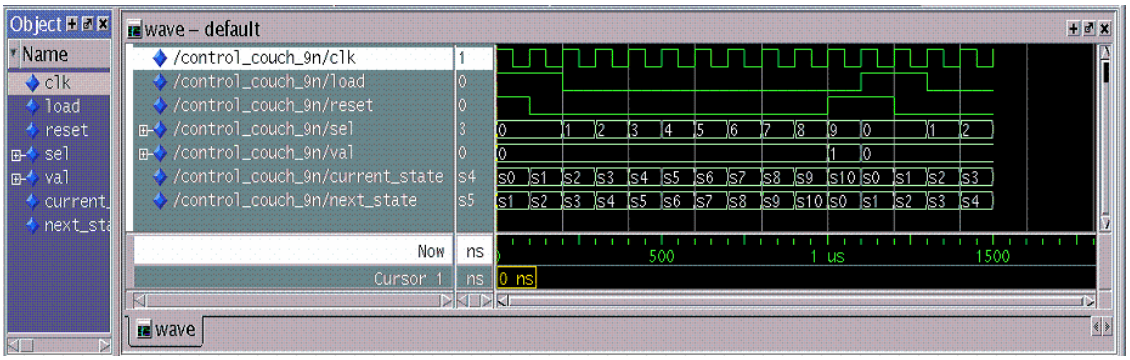


Figure V.46 chronogramme de simulation du bloc de contrôle de la troisième couche cachée

Le fonctionnement de la troisième couche cachée est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.47.

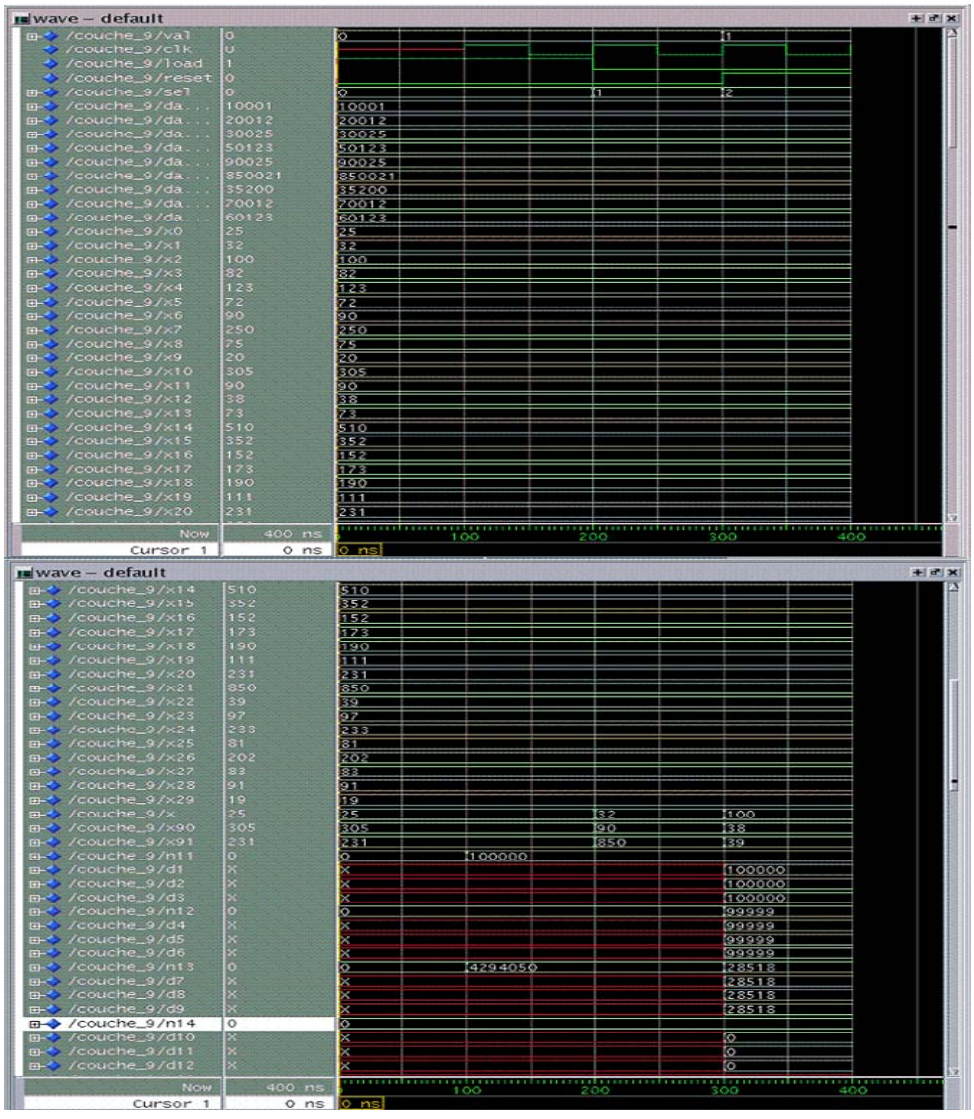


Figure V.47 chronogramme de simulation de la 3^{ième} couche cachée

V.12.7 simulation de la quatrième couche cachée (3 neurones)

Cette dernière est composée de 3 neurones précédés par 3 multiplexeurs « Mux » Figure V.48.

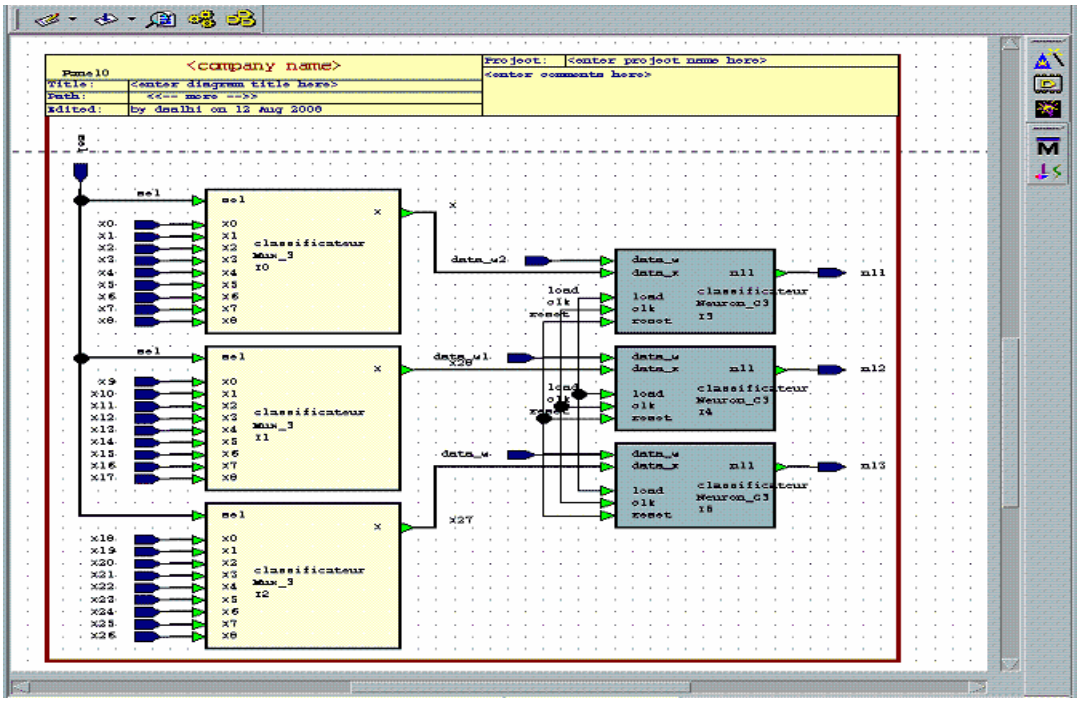


Figure V.48 structure interne de la quatrième couche cachée

V.12.7.1 simulation du bloc de contrôle

Figure V.49 présente le chronogramme de fonctionnement du bloc de contrôle de la quatrième couche cachée.

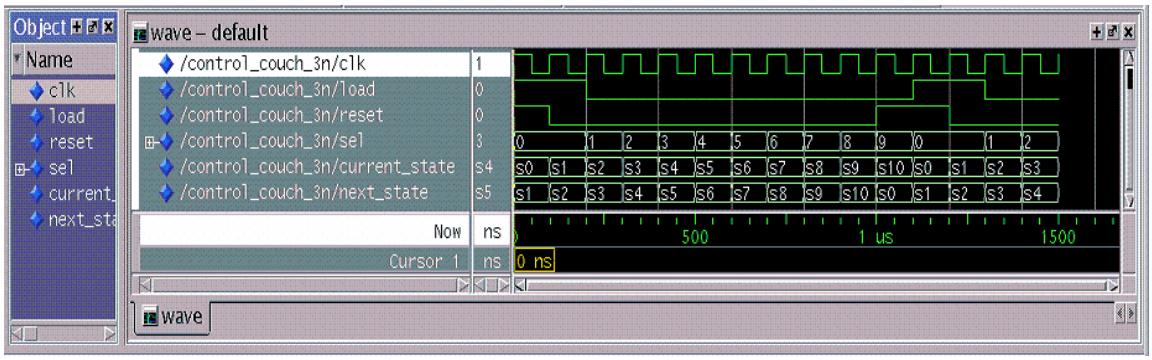


Figure V.49 chronogramme de simulation du bloc de contrôle de la quatrième couche cachée

Le fonctionnement de la quatrième couche cachée est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.50.

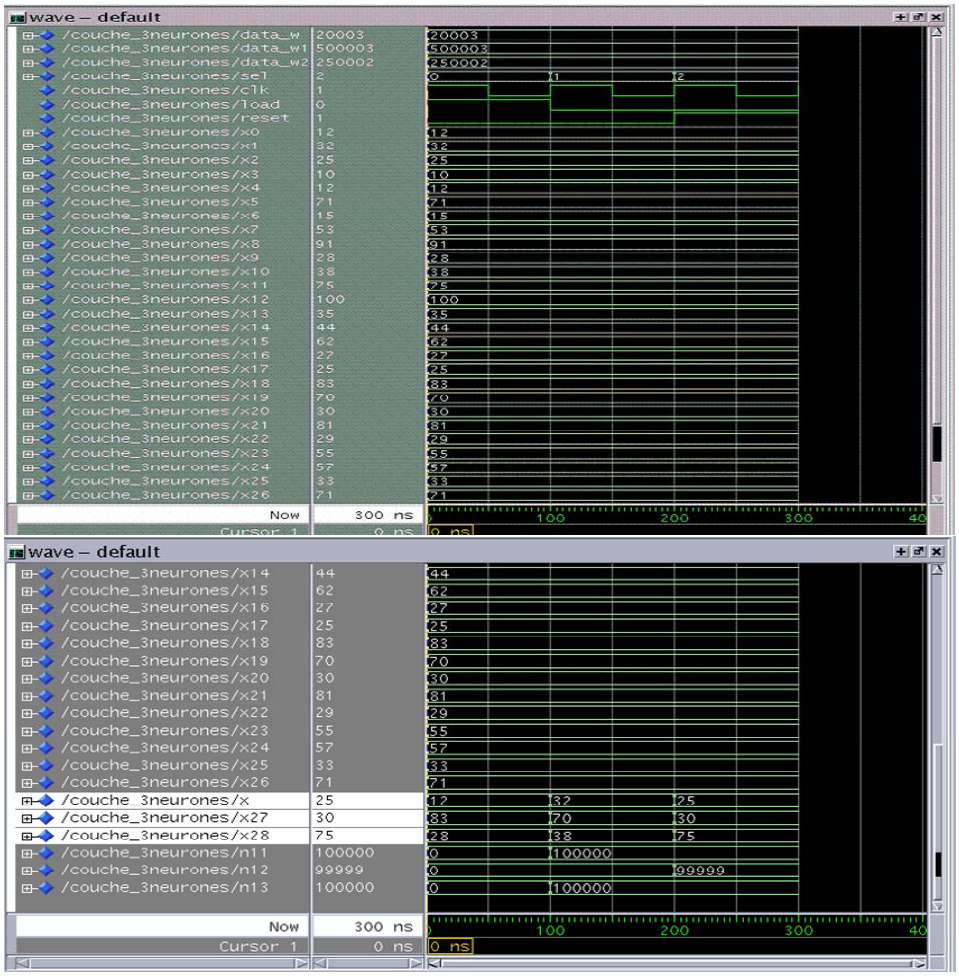


Figure V.50 chronogramme de simulation de la 4^{ième} couche cachée

V.12.8 simulation de la couche de sortie (1 neurones)

Cette dernière est composée d'un neurone précédé par un multiplexeur « Mux » Figure V.51.

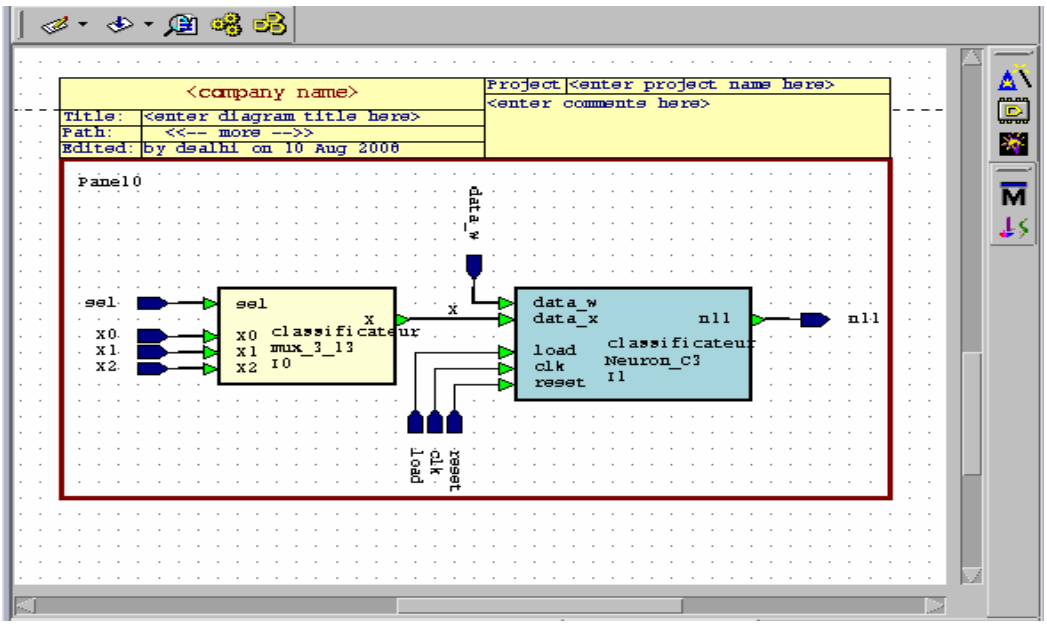


Figure V.51 structure interne de la couche de sortie

V.12.8.1 simulation du bloc de contrôle

Figure V.52 présente le chronogramme de fonctionnement du bloc de contrôle de la couche de sortie.

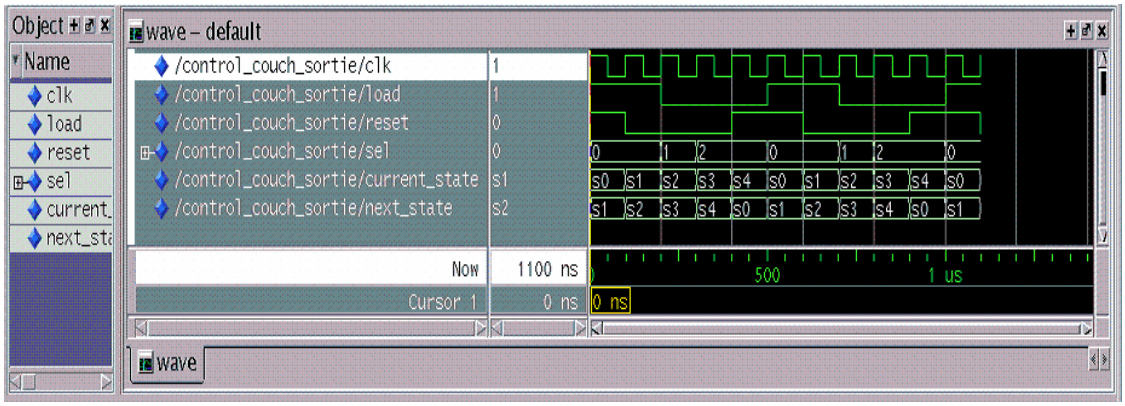


Figure V.52 chronogramme de simulation du bloc de contrôle de la couche de sortie

Le fonctionnement de la couche de sortie est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.53.

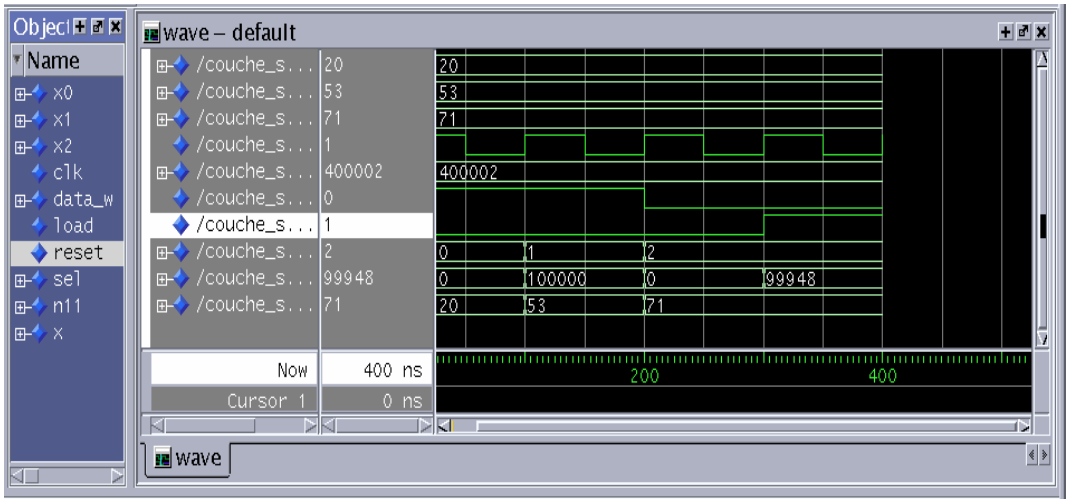


Figure V.53 chronogramme de simulation de la couche de sortie

V.11.8 simulation du module Feed Forward

Le fonctionnement du module Feed Forward est présenté par le chronogramme de la simulation de ce bloc avec l'outil Modelsim. Figure V.54.

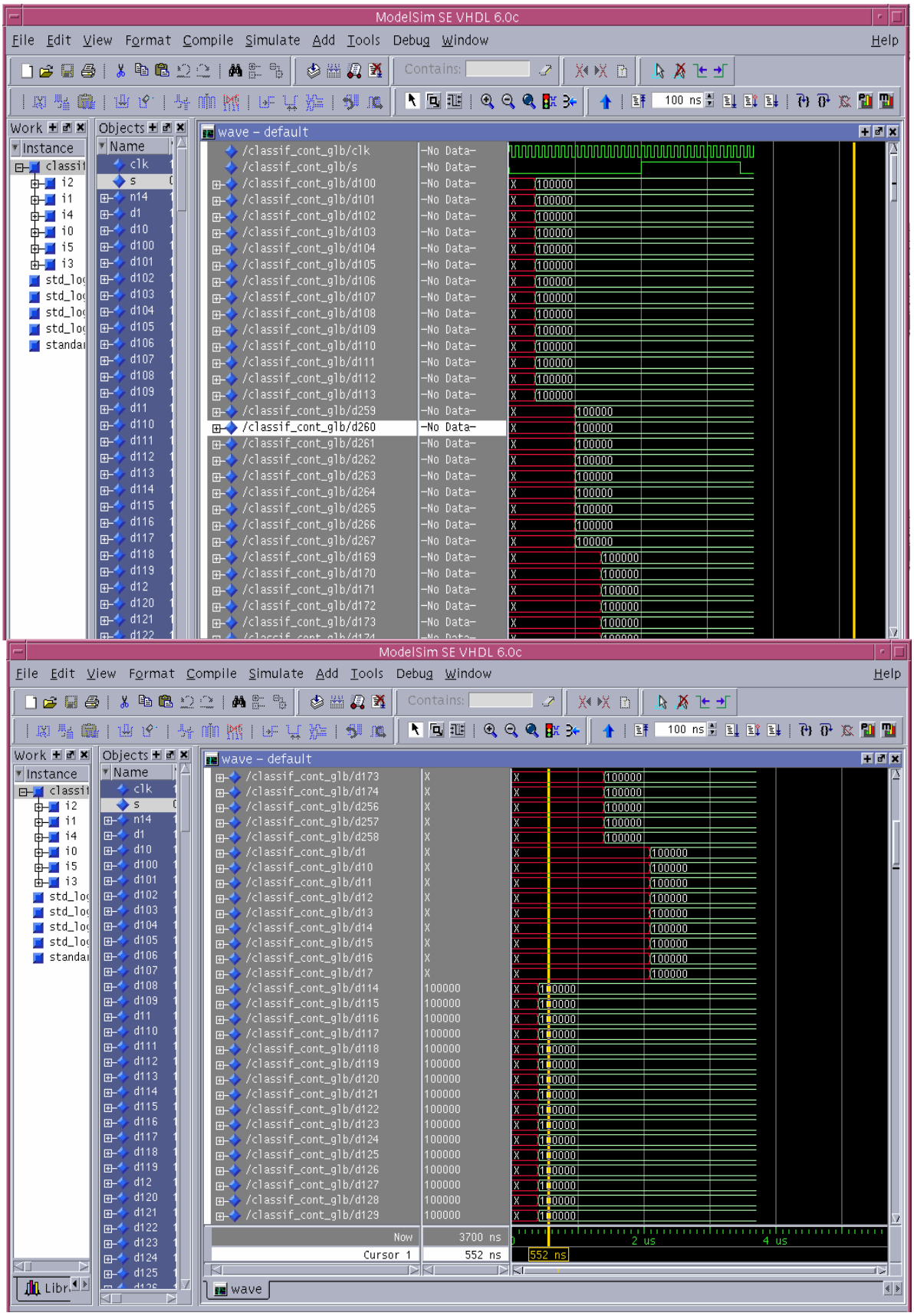


Figure V.54 une partie d u chronogramme de la simulation du classificateur.

Il faut mettre en évidence que chaque composant est précédemment décrit est aussi décrit en **VHDL**, et les programmes correspondants sont dans **l'annexe**.

Après avoir présenté les chronogrammes du bon fonctionnement des différentes composants dans l'architecture de notre classification, nous passons a la synthèse et le placement routage.

V.12. synthèse et placement & routage

V.12.1 Phase de synthèse : permet de passer d'un niveau d'abstraction élevé ' VHDL' vers un niveau d'abstraction plus bas qui est le niveau (RTL : Registre Translate Level) adapté au langage machine. Le résultat de la synthèse est un fichier dont l'extention '**.edif** '. qui permet a l'outile le placement & routage de l'architecture.

V.12.2 Phase de placement & routage : effectuer avec l'outil **ISE FONDATION** cette phase et composée de trois partie son :

- **La translation** : permet de convertir le fichier '**.edif** 'vers un fichier d'extention ' **NGD** dont il est comprhénsible par l'outil de placement routage .
- **Le mapping** : planification de l'implémentation des différentes parties de l'architecture vis-à-vis des ressources des circuits FPGA que nous ciblons dans notre application .
- **Le placement & routage**: il s'agit d'établir les différentes interconnexions entre les composants integrés dans le circuit.Figure V.55

- **Génération du fichier ' . bit '** : après le placement routage, l'outil **ISE Fondation** nous génére un fichier dont l'extention ' **.bit** ',un fichier binaire, téléchargeable, qui permet de configurer le circuit FPGA.

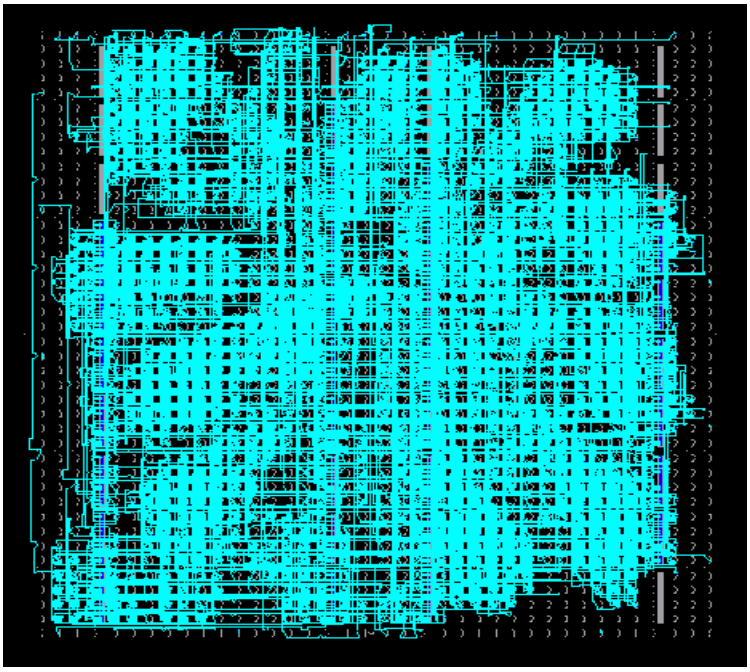


Figure V.55 placement & routage de l'architecture sur le circuit FPGA

IV.9 Résultats de l'implémentation

Pendant la phase de synthèse avec l'outil Léonardo Spectrum de Mentor graphics l'outil affiche dans le rapport de synthèse, que l'architecture neuronal (3 13 10 9 3 1), que nous avons développer a consommer tout les ressources du circuit FPGA VIRTEX-II XC2V1000 et que l'outil ne peut pas aller loin dans l'implémentation. C'est un résultat très logique on regardant la taille de l'architecture qui nécessite des mémoires et des multiplieurs ainsi que des multiplieurs qui vont être consommés directement des ressources interne du circuit FPGA. Cependant, nous avons basculé vers un autre circuit de la même famille c'est le circuit FPGA VIRTEX-II XC2V3000-4fb 967, ce circuit est 3 fois plus puissant par rapport a XC2V1000-4XC 456. Après l'étape de placement & routage l'outil ISE Fondation nous fournis un rapport sur la consommation des ressources de l'FPGA, correspondante a l'architecture implémentée. Tableau V.I.

Après la phase du placement & routage l'outil nous génère un fichier binaire dont l'extension est .bit, qui permet la configuration ou la programmation du circuit FPGA.

Tableau V.I rapport de la consommation des ressources sur FPGA après placement routage

```
**************************************************
Device Utilization for 2V3000bf957
**************************************************
Resource              Used    Avail   Utilization
--------------------------------------------------
IOs                     25      684       3.65%
Function Generators     23774   28672    82.92%
CLB Slices              11887   14336    82.92%
Dffs or Latches         3186    30724    10.37%
Block RAMs              0       96        0.00%
Block Multipliers       42      96       43.75%

--------------------------------------------------
Using wire table: xcv2-3000-4_wc

                        Clock Frequency Report

      Clock                : Frequency
      ----------------------------------
      Clk                  : 35.2 MHz
```

IV.10 Conclusion

Cette implémentation à démontrer que les ressources du circuit FPGA VIRTEX-II XC2V1000 ne sont pas suffisantes pour implémenter notre classificateur ; cependant. Un autre circuit de la même famille VIRTEX-II qui est le XC2V3000 a supporter cette architecture avec une consommation de 83%de la surface du circuit et horloge de fonctionnement de 35.2 MHz.

CONCLUSION GENERALE

Le travail présenté dans le cadre de ce mémoire, se rapporte à la conception et l'implémentation d'un classificateur neuronal basé sur l'algorithme de la rétro propagation du gradient RPG des rythmes cérébraux sur un circuit FPGA. Ce travail nous a permis d'acquérir un savoir faire très précieux dans le domaine de la conception, en particulier dédiée aux architectures neuronales. Il nous a permis aussi de nous familiariser avec les circuits FPGAs, le cycle de la conception et les différents outils de conception aussi bien software que hardware a noté l'outil Matlab en software, les outils FPGA Adventage, Leonardo Spectrum, Modelsim et IES Fondation en hardware. Cette étude nous a permis, aussi de nous initier dans le domaine de la microélectronique dédiée aux applications médicales.

L'architecture que nous avons conçue n'a pas pû être implémentée sur le circuit FPGA, XC2V1000 que nous avons ciblé au début de cette conception. Néanmoins, l'implémentation de notre classificateur a été réalisée avec le circuit FPGA XC2V3000 qui appartient à la même famille des circuits FPGA (VIRTEX-II de XILLINX), avec une consommation de 83% de la surface du circuit et une horloge de fonctionnement de 35.2 MHz. cela nous a permis d'évaluer la capacité d'intégration des différents circuit FPGAs dans la famille VIRTEX-II de XILLNX.

Pendant la réalisation de ce travail, nous avons rencontré un certain nombre de problèmes techniques tel que la non disponibilité des bases de données nécessaire à ce type d'application, c'est la raison pour laquelle nous avons conçu notre propre base de données.

La perspective que nous pouvons envisager sur ce travail est la construction des différents bases de données telles que les bases de données des signaux EEG, ECG, EMG ou autres nécessaires pour le développement de telles applications dans toutes les disciplines, ainsi que le développement de nouvelles méthodes et algorithmes qui permettent une meilleure exploitation de l'intelligence artificielle présentés par les réseaux de neurone et les mettre en oeuvre dans des systèmes fonctionnels commercialisés. D'autre pat nous envisageons aussi une optimisation de l'architecture pour aboutir a des performances de fonctionnement aussi proches de 100%, et de généraliser ce genre d'approche pour d'autres types de modification pathologique sur le signale EEG dans le but de concevoir une plateforme intelligente pour la classification des différents état sur le signale EEG et les employés dans des systèmes plus complexes telles que les stimulateurs neurologique utilisés actuellement comme thérapie pour plusieurs types de trouble de fonctionnement neurologique chez l'homme.

BIBLIOGRAPHIES

[1] Torsten Felzer ; « On the possibility of Developing a brain-computer interface (BCI)"technical university of Darmstadt; Department of computer Science; technical Report; 2001; fleze&informatik.tu- Darmstadt.de.

[2] Adrien Depeursinge ; Touradj Ebrahimi; Gary N. Garcia "Nonlinear EEG analysis and classification" école polytechnique Fédérale de LAUSANNE ; rapport technique ;fevrier 2004.

[3] «Virtex-II Platform FPGAs: Complete Data Sheet» www.xilinx.com

[4] A.C. Guyton ; « Précis de Physiologie médicale ». PICCIN 1996.

[5] Gilles Bourbonnais Cégep de Sainte-Foy ; « Introduction au système nerveux ».

[6] Claude TOUZET ; « Réseaux de neurones », 1992.

[7] SALOMON, E.P. et DAVIS, P.W ; « Développement neurobiologique. Description du système nerveux ». (Adaptation française de CHOLETTE C). "*Anatomie et physiologie humaines*", McGraw-Hill 1981.

[8] P.Cesaro, Y. Keravel, H. Ollat, M. Peschanski, M. Sindou NeuroAnatomie fonctionnelle : de la cellule aux comportements. Vol.1 Le neurone.. ANPP.

[9] Gilles Bourbonnais « Neurophysiologie « chapitre 5 » » Université Laval.

[10] Germàn G´omez Herrero « Mismatch Negativity Detection in EEG recordings using Wavelets », June 2003.

[11] « Electroencéphalographie Module de base Physiologie », Faculté de Médecine

Montpellier-Nîmes, Novemnbre 2004.

[12] M. Teplan Fundamentals of EEG Measurement, Measurement Science Review, Volume 2, Section 2, 2002

[13] Neuro-anatomie fonctionnelle Professeur OUTREQUIN.

[14] Vadim V. Nikouline "Magnetoencephalographic and electroencephalographic studies of Spontaneous activity and evoked responses in the sensorimotor system", University of Helsinki 2001.

[15] R.D. Bickford. In: Adelman G. ed. Encyclopedia of Neuroscience, Birkhauser, Cambridge 1987 (USA), 371-373.

[16] Brunel, Philippe Maillet « LE CERVEAU ET L'ACQUISITION DES CONNAISSANCES » LA GESTION DES CONNAISSANCES –NOVEMBRE 2003

[17] Marc Parizeau « RESEAUX DE NEURONES » Université de LAVAL, automne 2004, GIF-21140 et GIF-64326.

[18] Fabien Moutarde « INTRODUCTION AUX RESEAUX DE NEURONES », Ecole des Mines de Paris, Mars 2007.

[19] Fabien Lotte, Anatole Lécuyer, Yann Renard ; Fabrice Lamarche Bruno Arnaldi, « Classifcation de Données Cérébrales par Système d'Inférence Flou pour l'Utilisation d'Interfaces Cerveau-Ordinateur en Réalité Virtuelle IRISA-INRIA Rennes IRISA-INSA Rennes 1.

[20] M.T. Hagan, H.B. Demuth, M. Beale, «Neural Network Design», PWS Publishing Compagny, 1995.

[21] J.C. Principe, N.R. Euliano, W.C. Lefebvre, «Neural and Adaptive Systems: Fundamentals through Simulations», Wiley, 2000.

[22] Simon Haykin, «Neural Networks: A Comprehensive Foundation», IEEE Press, 1994.

[23] J.A. Freeman, D.M. Skapura, «Neural Networks: Algorithms, Applications, and Programming Techniques», Addison-Wesley, 1992.

[24] R.P. Lippmann, « An Introduction to Computing with Neural Nets», IEEE ASSP Magazine, pp. 4-22, avril 1987.

[25] R. Krishnapuram, J.M. Keller, «A Possibilistic Approach to Clustering», IEEE Transactions on Fuzzy Systems, vol. no. 2, p. 98-110, 1993.

[26] Bernd Fritzke, «A Growing Neural Gas Network Learns Topologies», Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky et T.K. Leen (´editeurs), MIT Press, Cambridge MA, 1995.

[27] Jean-Pierre Rospars le neurone biologique et sa modélisation, Traitement de l'information par les neurones Physiologie de l'insecte, INRA Versailles& Mathématiqueset Informatiqua Appliquées, INRA Jouy, INA-PG, 14 novembre2005

[28] I. rivals, l. personnaz, G. dreyfus « modélisation, classification et commande par réseaux de neurones : principes fondamentaux, méthodologie de conception et illustrations industrielles » Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris Laboratoire d'Electronique 10, rue Vauquelin 75005 PARIS

[29] http://www.curveunscan.com/fr/features.htm.

[30] S.D.Brown, R.J.Francis, J. Rose, S.G.Vranesic «Field-Programmable Gate Arrays» Kluwer Academic Publishers, 1992

[31] Etienne Messerli, Yves Meyer «Electronique Numérique, 1er tome, Systèmes combinatoires» l'école d'ingénieurs de l'arc jurassien Version 1.0 décembre 2004

[32] S.Brown and J.Pose «FPGA and CPLD Architectures». A tutorial IEEE Design and Test of computers summer 1996.

[33] T.A. York, «Survey of field programmable logic devices, Microprocessors and Microsystems». 7 (17) 1993 371-381.

[34] «Virtex-II Platform FPGAs : Detailed Description» Xilinx Mai 2003. URL

http ://direct.xilinx.com/bvdocs/publications/ds031-2.pdf

[35] Jonathan Rose, Abbas El Gamal, Alberto Sangiovanni-Vincentelli «Architecture of Field-Programmable Gate Arrays» Proceedings of the IEEE, Volume: 81, Issue: 7, Pages: 1013-1029, July 1993.

[36] S. Brown, « FPGA architectural research: a survey» Design & Test of Computers, IEEE Volume 13, Issue 4, Page(s):9 - 15, winter 1996

[37] J. S. Rose and S. Brown « Flexibility of Interconnection Structures for Field Programmable Gate Arrays» EEE JSSC, Vol. 26, NO. 3, pp. 277-282, March 1991

[38] François Verdier «Systèmes Intelligents et Communicants » Les circuits FPGA Concepts de base, architecture et applications Cours Master Recherche. Université de Cergy-Pontoise Laboratoire ETIS - UMR CNRS 8051. Www-etis.ensea.fr/~verdier/

[39] Xilinx, Un article de Wikipédia, l'encyclopédie libre.13 juillet 2007

http://www.xilinx.com/products/silicom_solutions/fpgas/virtex4/index.html

[40] «The programmable logic Data Book» VHDL and Verilog Simulation for Work stations, V-system/plus work station, User's Manual. Version 6.0 – PLDshell Plus/PLDasm User's Guide V4.1, Intel. Xilinx 1994

[41] V. Betz, J. Rose and A. Marquardt «Architecture Deep-Submicron FPGAs»

Kluwer Academic Publishers, 1999

[42] Philipe Larcher «Introduction à la synthèse logique»Eyrolles, 1997.

[43] «VHDL du langage au circuit, du circuit au langage», J. Weber & M. Meaudre, Masson, 1997.

[44] «VHDL du langage à la modélisation», Airiau & Berge & Olive & Rouillard.

[45] «IEEE, Standard VHDL Langage Référence Manual», ANSI/ IEEE std 1076-1993.

[46] « IEEE Standard multivalue Logic Système For VHDL Model Introperability» IEEE std 1164-1993.

[47] Patrick Cohen. « Elément D'analyses Et De Synthèses En Electronique Numérique».

[48] data sheet Mentor graphics.

[49] www.Xillinx.com.

ANNEXE I

Etat de l'art :

Afin de comparer entre les réalisations possibles plusieurs critères sont établies dans la littérature, tels que : la représentation des données, la stratégie d'interconnexion, la technologie, mesure de performance et la précision arithmétique.

- **Stratégie d'interconnexion :**

  Quelques conceptions consistent à traiter des nœuds divisés sur un bus. Les machines SIMD (Single Instruction Multiple Data) GF11 et CM-2 comptent surtout sur des connexions entre des unités adjacentes, tandis que les MIMD (Single Instruction Multiple Data) iPSC/860 ont une connectivité d'hyper-cube. D'autres conceptions font simplement les connexions entre les unités de traitement quand c'est nécessaire.

- **Représentation des données :**

  Les valeurs des poids synaptiques, des entrées et des sorties peuvent être encodées soit en analogique soit en numérique. Les conceptions qui utilisent la représentation numérique diffère selon que ces valeurs soient en virgule flottante, en série ou en parallèle.

- **Technologie :**

  Les architectures des réseaux de neurones sont réalisées sur des puces VLSI. Mais en combinaison avec les puces DSP ou FPGA, plusieurs implémentations utilisent les processeurs MIMD ou SIMD précédemment conçu.

- **Performance :**

  Les unités de performance employée pour la comparaison entre les différentes architectures sont : MCPS (des millions de connexions par seconde, pour le module Feed-forward seulement) et MCIJPS (des millions de connexion de mises à jour par seconde, pour la Feed-forward et la Rétro-propagation). Cette mesure de performance dépend de la taille du réseau. L'efficacité de mesure est calculée en divisant le nombre MCUPS par nombre de poids dans le système.

- **Précision arithmétique :**

  La plupart des algorithmes connectioniste sont basés sur des variables continues. L'émulation par des logiciels emploie typiquement une représentation en virgule flottante. Pour l'implémentation hardware, et vu le coût élevé d'arithmétique de virgule flottante, le concepteur est contraint d'utiliser une représentation en point fixe comme compromis entre la précision et le coût (dans le cas de valeurs digitales). En ce qui concerne les valeurs analogues, elles ne sont pas spécifiquement quantifiées, mais la précision de telles valeurs est typiquement de précision limitée. Pour des buts de comparaison, ces valeurs peuvent être converties à une précision de bit équivalente. La précision exigée pour les différentes couches d'une architecture

connectioniste peut varier largement : beaucoup d'architectures profitent de ce fait pour économiser des ressources en réduisant la précision quand c'est possible. Cependant des études récentes ont montré qu'une précision de 12 bits et plus des poids synaptiques sont nécessaires pendant la phase d'apprentissage. Le tableau I.I Montre les différentes réalisations avec leurs critères de performances.

Dans la littérature, nous avons recensé un certain nombre de réalisation d'implémentation de l'algorithme RPG. Ces réalisations peuvent être groupées en deux grandes classes : les implémentations à base de multiprocesseur et les implémentations hardware dédiées.

- **Implémentation multiprocesseur :**

Les implémentations à base de multiprocesseur sont basées sur l'utilisation de plusieurs matrices de processeurs qui permettent de réaliser des supercalculateurs. L'obtention de hautes performances sur de telles machines, nécessite la connaissance approfondie de l'architecture de la machine et le réglage détaillé du logiciel. Les réalisations les plus connues sont les suivantes :

CM-2 :

La Machine connexion CM-2 est un ordinateur parallèle SIMD ayant des processeurs de 64Ko, qui fonctionnent sur des données à 1 bit. Un CM-2 avec des processeurs de 4 Koctets a été employé pour simuler l'algorithme de la Rétro-propagation du gradient, avec une performance de 2.5 MCUPS. Les auteurs estiment que sur un CM-2 avec des processeurs de 64 Koctets, l'exécution serait 40 MCUPS. Le calcul de la phase Feed-forword se fait comme suit :

1. Chaque processeur reçoit une valeur d'entrée pour son unité d'entrée.
2. Chaque processeur multiplie la valeur d'entrée par le poids approprié; le produit est accumulé dans le nœud caché.
3. Les valeurs d'entrée tournent à travers des processeurs : chaque processeur envoie l'entrée l'estiment reçu à son voisin gauche; le processeur extrême gauche envoie sa valeur au processeur extrême droit.
4. Répéter des pas 2 et 3 avant que chaque processeur n'ait vus chaque valeur d'entrée; cela achève le calcul de couche cachée.
5. Répéter les pas 2, 3 et 4, des valeurs de couche de production de calcul au lieu des valeurs de couche cachée.

IBM GF11 :

IBM GF11 est un ordinateur expérimental SIMD avec 566 processeurs réalisant une exécution maximale de 11 Giga flops. Chaque processeur est capable d'exécuter 20 millions d'action de flottant ou des opérations de point fixées par seconde. Les processeurs contiennent les mots de 16 Koctets de RAM statique et les mots de 512 Koctets de mémoire dynamique. Les processeurs sont connectés via un réseau reconfigurable Benès. L'implémentation d'algorithme de la Rétro

propagation reproduit le réseau connectioniste pour être simulé à travers tous les processeurs fonctionnants de la machine; en présentant un sous-ensemble différent d'exemples recevant une formation à chaque processeur. À des intervalles réguliers des processeurs se communiquent leurs mises à jour de poids accumulées et calculent de nouveaux poids : ainsi le groupe des variables de la Rétro propagation est imité. La configuration de la carte choisie est ici appropriée au matériel dont les équipements de connexion qui ont moins de performances par rapport au modèle de traitement qui a une fréquence d'exécution de haute performance.

Le GF11 a été évalué sur le NETtalk jeu de formation, en utilisation le réseau 203-60-26 réseaux. Avec 356 processeurs opérationnels, le GF11 réalise 901 MCUPS, où chaque calcul d'un compte de changement de poids comme une mise à jour de poids (cela fait la comparaison de sur-lire et le groupe de variable de la Rétro propagation possible). Les auteurs estiment qu'avec 566 processeurs opérationnels, 1231 MCUPS seraient réalisés.

iPSC/860 :

L'iPSC/860 est un système hyper-cube de base MIMD fait par la Division de Systèmes de Super-ordinateur d'Intel Corporation l'iPSC/860 peut être configuré avec jusqu'à 128 processeurs; les programmes peuvent être écrits dans un surensemble du langage C qui soutient la communication d'inter-processeur et d'autres fonctions nécessaires. Chaque processeur dans le iPSC/860 a sa propre mémoire de données et ses propres instructions. La communication entre des processeurs est implantée en passant des messages, ou les blocs de données.

Pour des buts d'évaluation de performance, les données employées dans le projet de NETtalk ont été implantées sur l'iPSC/860, employant des différents nombres de processeurs. IPSC/860 à 32 processeurs a réalisé 12 MCTIPS. Pour la comparaison, iPSC/860 à 1 processeur a réalisé 1 MCUPS.

➢ Implémentation hardware dédiée :

Dans cette section, on décrit trois architectures proposées pour l'émulation d'algorithmes connexionnistes.

SPERT :

Les auteurs proposent de construire un super-computer neuronal universel, avec une performance maximale de 100 MCUPS. Comme un premier arrêt, une tradition VLSI "neuro-microprocessor", SPERT, a été le contrôle de programme conçu sous SPERT exécute quelques opérations généralement nécessaires dans des architectures connectionistes, avec l'accent spécial sur des réseaux à deux couches Rétro-propagation. Les programmes sont écrits dans l'assembleur. Les précisions des poids et des activations sont fixées à 16 et 8 bits, respectivement : les concepteurs ont empiriquement décidé que cette précision était adéquate pour réaliser des résultats comparables avec

une simulation qui a employé des valeurs de virgule flottante 32 bits (dans une application de classification de discours). L'architecture SPERT emploie un bus à 128 bits pour la communication avec la mémoire externe, où les instructions et les valeurs de poids sont stockées. Un petit bus en série est employé pour transmettre les valeurs du modèle (l'entrée et la sortie désirée) à tous les processeurs. La performance de SPERT simulant la variante en ligne de la Rétro-propagation a estimé une performance maximale d'environ 100 MCUPS est réalisés pour de très grands réseaux (par exemple 2048 l'entrée, caché et des unités de production). Forward propagation a une performance de propagation d'environ 350 MCPS pour le même réseau.

CNAPS :

L'architecture CNAPS est construite à 64 unités de traitement ″processingunit chip″ capables d'implanter beaucoup de modèles connectionistes. Les processeurs partagent à 8 bits les bus d'entrées et de production pour la transmission de valeurs de données. Les systèmes sont construits de quelques chips et un ou plus de chips de séquenceur qui émettent des commandes à toutes les chips via un autobus d'instruction à 31 bits.

La configuration de modèles connectioniste à l'architecture CNAPS est flexible : les unités de réseau peuvent être imitées par des unités de traitement simples, des unités de traitement multiples peuvent imiter des unités de réseau simples et des unités de réseau multiples peuvent être imitées dans des unités de traitement simples. Ainsi l'utilisateur du système peut faire du commerce du nombre de processeurs contre le temps d'exécution, et de très grands réseaux peuvent être implantés dans de petits systèmes.

La performance pour un réseau avec des entrées 1900, 500 unités cachées et 12 productions sont estimées dans un système proposé à huit chips (512 unités de traitement) réalise 2379 MCUPS avec l'utilisation d'un taux d'horloge de 25 MHz.

ANNEXE II

Description VHDL de la Ram

```
        LIBRARY ieee;
        USE ieee.std_logic_1164.all;
        use ieee.numeric_std.all;
        USE ieee.std_logic_unsigned.all;

entity romw11 is
    Port ( valid   : IN     std_logic;
           addr    : IN     std_logic_vector (5 DOWNTO 0);
    data_x,data_w : OUT    std_logic_vector (22 DOWNTO 0)
   );
end romw11;

architecture Behavioral of Rom is

subtype word is std_logic_vector (22 downto 0) ;
type romx is array ( 49 downto 0) of std_logic_vector (22
downto 0);
type romw is array ( 49 downto 0) of std_logic_vector (22
downto 0);

constant romx_array : romx := ( "10000001000001011011100",
     "10000001000000101111110",
     "10000001000000100011010",…
);
constant romw_array : romw := ( "10000001101100110110010",
     "10000001010101110100100",
     "10000010011110111011010",…
     );
BEGIN
process ( addr, valid)
begin
if valid = '1'then data_x <= romx_array(conv_integer(addr) ) ;
                   data_w <= romw_array(conv_integer(addr) ) ;
else
data_x <= ( others => '0');
data_w <= ( others => '0');
    end if;
    end process;
    end Behavioral;
```

Description VHDL du Multiplieur

```
        LIBRARY ieee;
        USE ieee.std_logic_1164.all;
        USE ieee.std_logic_arith.all;
        use IEEE.STD_LOGIC_UNSIGNED.ALL;

 entity MULW11 is
     Port ( data_x : in std_logic_vector(22 downto 0);
            data_w : in std_logic_vector(22 downto 0);
            prod : out std_logic_vector(22 downto 0));
 end MULW11;

 architecture Behavioral of MULW11 is

 BEGIN

  I0combo: PROCESS (data_x, data_w)
  VARIABLE dtemp : std_logic_vector(45 DOWNTO 0);
  BEGIN
       dtemp := (unsigned(data_x) * unsigned(data_w));
       prod <= dtemp(22 DOWNTO 0);
  END PROCESS I0combo;
end Behavioral;
```

Description VHDL de l'Accumulateur

```
          library IEEE;
         use IEEE.STD_LOGIC_1164.ALL;
         use IEEE.STD_LOGIC_ARITH.ALL;
         use IEEE.STD_LOGIC_UNSIGNED.ALL;

    entity accw11 is
       PORT(
      clk     : IN     std_logic;
      rprod     : IN     std_logic_vector (22 DOWNTO 0);
      load    : IN     std_logic;
      reset   : IN     std_logic;
      acc    : OUT    std_logic_vector (22 DOWNTO 0)
   );

   end accw11;

architecture Behavioral of accw11 is

.
SIGNAL MW_I0cregl, MW_I0nregl : std_logic_vector(22 DOWNTO 0);
   BEGIN
   acc <= MW_I0cregl;
I0seq: PROCESS (clk, reset, MW_I0nregl,rprod, MW_I0cregl, load)
VARIABLE dtemp, ntemp, ctemp : std_logic_vector(23 DOWNTO 0);
    begin
      IF (clk'EVENT AND clk='1') THEN
            MW_I0cregl <= MW_I0nregl;
         IF (reset = '1' OR reset = 'H') THEN
           MW_I0cregl <= MW_I0cregl;
        END IF;
     END IF;
      ctemp := '0' & MW_I0cregl;
      dtemp := '0' & rprod;
      ntemp := (std_logic_vector(unsigned'(unsigned(ctemp)
                                + unsigned(dtemp))));

      IF (load = '1' OR load = 'H') THEN
         MW_I0nregl <= ntemp(22 DOWNTO 0);
      ELSIF (load = '0' and reset ='0') THEN
         MW_I0nregl <= rprod;
     elsif (load ='1' and reset ='1') then
         MW_I0nregl <= "00000000000000000000000";
         MW_I0cregl <= "00000000000000000000000";

        END IF;
    END PROCESS I0seq;
      end Behavioral;
```

Description VHDL de l'Additionneur

```
      LIBRARY ieee;
      USE ieee.std_logic_1164.all;
      USE ieee.std_logic_arith.all;

     use IEEE.STD_LOGIC_SIGNED.ALL;

    entity addw11 is
         PORT(
         racc : IN     std_logic_vector ( 22 DOWNTO 0);
         add : OUT    std_logic_vector ( 22 DOWNTO 0)

          );
     end addw11;
   architecture Behavioral of addw11 is
      constant b11: STD_LOGIC_VECTOR (22 downto 0) :=
"10000001101011110000010";
       BEGIN
       add <= signed(b11) + signed (racc);
       end Behavioral;
```

Description VHDL de la Lut

```
     library IEEE;

     entity lut is
    Port ( Ai : in std_logic_vector(22 downto 0);
           N11 : out std_logic_vector(22 downto 0));
end lut;

architecture Behavioral of lut is
begin
 if Ai>=0       then

if    ((Ai>=00001)   and (Ai<=10000))   then
                    Si <= "00000000010001000100101"; --0.08741
elsif ((Ai>=10001)   and (Ai<=20000))   then
                    Si <= "00000000010101100100111"; --0.11047
elsif (Ai>=644001)                      then
                    Si <= "00000011000011010100000"; --1.0000
           else                    Si<= (OTHERS => 'X');

end if;
 else
if    ((Ai>=-2300000)and (Ai<=-650000)) then
                   Si <= "10000011000011010100000";-- -1
elsif ((Ai>=-649999) and (Ai<=-550000)) then
                   Si <= "10000011000011010011110"; -- -0.99998
elsif ((Ai>=-3999)   and (Ai<=00000))   then
                   Si <= "10000000000001000101111"; -- -0.00559
        else       Si <= (OTHERS => 'X');
   end if;
   end if;
   end process;
       end Behavioral;
```

Description VHDL du multiplexeur

```
      ENTITY MUX IS
     PORT(
     X0 : IN     std_logic_vector (15 DOWNTO 0);
     X7 : IN     std_logic_vector (15 DOWNTO 0);
     X : OUT    std_logic_vector (15 DOWNTO 0)
     );
     END MUX ;

ARCHITECTURE MUX_8 OF MUX_8 IS
  SIGNAL MW_I0din0l : std_logic_vector(15 DOWNTO 0);
  SIGNAL MW_I0din7l : std_logic_vector(15 DOWNTO 0);
BEGIN

PROCESS (MW_I0din0l, MW_I0din1l,MW_I0din3l
,MW_I0din2l,MW_I0din4l,MW_I0din5l,MW_I0din6l,MW_I0din7l, sel)
  VARIABLE dtemp : std_logic_vector(15 DOWNTO 0);
  BEGIN

     CASE sel IS
     WHEN "000" => dtemp := MW_I0din0l;
     WHEN "001" => dtemp := MW_I0din1l;
     WHEN "111" => dtemp := MW_I0din7l;

    WHEN OTHERS => dtemp := (OTHERS => 'X');
     END CASE;

     X <= dtemp;
  END PROCESS ;
  MW_I0din0l <= X0;
  MW_I0din7l <= X7;
    END MUX;
```

Description VHDL du neurone

```
    entity neurone1 is

  PORT(
      addr  : IN     std_logic_vector (5 DOWNTO 0);
      N11   : OUT    std_logic_vector (22 DOWNTO 0)
      );
end neurone1;
architecture Behavioral of neurone1 is

COMPONENT LUT
   PORT (
      Ai  : IN     std_logic_vector (22 DOWNTO 0);
      N11 : OUT    std_logic_vector (22 DOWNTO 0)
   );
   END COMPONENT;

COMPONENT mulw11
   PORT (
      data_x, data_w : IN     std_logic_vector (22 DOWNTO 0);
      prod     : OUT    std_logic_vector (22 DOWNTO 0)
  );
   END COMPONENT;

COMPONENT accw11
   PORT (
      rprod : IN     std_logic_vector (22 DOWNTO 0);
      acc   : OUT    std_logic_vector (22 DOWNTO 0)
   );
   END COMPONENT;

COMPONENT romw11
   Port ( addr    : IN     std_logic_vector (5 DOWNTO 0);
          data_x, data_w : OUT    std_logic_vector (22 DOWNTO 0)

   );
   END COMPONENT;

  COMPONENT addw11
   PORT (
      racc : IN     std_logic_vector (22 DOWNTO 0);
      add  : OUT    std_logic_vector (22 DOWNTO 0)
    );
   END COMPONENT;

        begin
        I2 : LUT
         PORT MAP (
         Ai=> S0 ,
         N11 => N11
         );

        I4 : addw11
        PORT MAP (
        racc => S1,
        add => S0);

         I3 : accw11
         PORT MAP (
           acc => S1
          );
          I1 : mulw11
          PORT MAP (
          prod => S2
          I0 : romw11
          PORT MAP (
          data_x =>S3, data_w =>S4, );
          end Behavioral;
```

ملخص

هدا العمل الذي عرض في هذا البيان الموجز، يتعلق بتصميم وتنفيذ جهاز لتصنيف الإيقاعات في الدماغ على رقاقة "اف.بي.جي.ا" باستعمال خوارزميه تتركز على أساس انتشار عكس اتجاه الانحدار "ار.بي.جي" و استعمال المسح الدماغية "او.او.جي", البناء الشامل للمصنف لدينا تم تنفيذه على رقاقة من النوع التي استهلكت مواردها الداخلية بنسبة 38 بالمائة و أعطى ساعة للتشغيل تقدر ب 35.2 ميغاهرتز

Abstract

In this work thesis, we describe was concerned by the design of a neural network classifier based on RPG algorithm of cerebral rhythms using EEG record. The principal axis of this application is the use of the back propagation of gradient algorithm as the main supervised learning algorithm, applied to a multi-layer perceptron neural network. The whole design of this classifier was successfully implemented on the XC2V300 FPGA VIRTEX-II circuit, achieving an area consummation of 83% of FPGA resources and a clock of the order of 35.2 MHz.

Résumé

Le travail présenté dans ce mémoire, consiste à concevoir et implémenter un classificateur neuronal basé sur l'algorithme de la rétro propagation du gradient RPG, des rythmes cérébraux en utilisant l'enregistrement EEG. L'architecture globale de notre classificateur est implémentée sur le circuit VIRTEX-II XC2V3000 de XILINX, cette architecture a consommé 83% des ressources internes du circuit FPGA avec un fonctionnement piloté par une horloge de l'ordre de 35 MHz.