Cellules analogiques CMOS pour réseaux de neurones. Application à la classification des cellules cancéreuses dans le sein

Hassan Jouni

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Cellules analogiques CMOS pour réseaux de neurones : Application à la classification des cellules cancéreuses dans le sein

Hassan JOUNI

Polytech'Lab

Présentée en vue de l'obtention
du grade de docteur en Electronique
d'Université Côte d'Azur
et de Lebanese International University
Dirigée par : Gilles Jacquemod
Co-dirigée par : Adnan Harb
Co-encadrée par : Yves Leduc, Professeur associé, Université Côte d'Azur
Soutenue le : 14 Décembre 2018

Devant le jury, composé de :
Yann Deval, Professeur, INP Bordeaux
Mohamad Sawan, Professeur, Ecole Polytechnique Montréal
Luc Hébrard, Professeur, Université Strasbourg
Yves Leduc, Professeur associé, Université Côte d'Azur
Adnan Harb, Professeur, LIU Beyrouth
Gilles Jacquemod, Professeur, Université Côte d'Azur

POLYTECH NICE-SOPHIA
POLYTECH'LAB

Cellules analogiques CMOS pour réseaux de neurones : Application à la classification des cellules cancéreuses dans le sein

Jury :

Rapporteurs
Yann Deval, Professeur, INP Bordeaux
Mohamad Sawan, Professeur, Ecole Polytechnique Montréal, Canada

Examinateurs
Luc Hébrard, Professeur, Université Strasbourg

Directeur
Gilles Jacquemod, Professeur, Université Côte d'Azur

Co-Directeur
Adnan Harb, Professeur, Lebanese International University, Liban

Co-Encadrant
Yves Leduc, Professeur associé, Université Côte d'Azur

Résumés et mots clés

Titre Cellules analogiques CMOS pour réseaux de neurones : Application à la classification des cellules cancéreuses dans le sein

Résumé Les réseaux de neurones artificiels sont particulièrement intéressants pour les implémentations CMOS VLSI (Very Large Scale Integration Complementary Metal-Oxide Semiconductor) car chaque élément parallèle (neurone ou synapse) est relativement simple, permettant l'intégration complète de grands réseaux sur une seule puce. Les multiplieurs, la fonction non-linéaire et sa dérivée sont des éléments clés essentiels dans le traitement du signal analogique et notamment dans la mise en œuvre VLSI analogique de réseaux neuronaux artificiels. Les principales conditions de ce type de circuits sont les suivantes : une faible surface de Silicium et une faible consommation électrique. Pour valider notre approche, nous avons choisi comme type d'application, la classification de cellules cancéreuses (malignes ou bénignes) du sein. Il existe de nombreux types de réseaux de neurones: réseau de neurones à réaction avec rétro-propagation (MLP), réseau de base radiale (RBN), réseau de neurones récurrents (RNN) et autres. Le réseau de neurones étudié dans cette thèse est basé sur l'architecture Multi-Layer Perceptron, formé par la rétro-propagation.
L'objectif principal est de trouver les meilleurs compromis et optimisations pour réaliser des circuits avec une technologie mature HCMOS9A 130 nm de STMicroelectronics alimentés sous ±900mV afin d'avoir le coût le plus faible possible. Après avoir choisi le meilleur algorithme (le plus simple et efficace) pour une implémentation VLSI simple, nous avons défini une architecture analogique efficiente. Enfin les briques de base ont été conçues et réalisées avant l'intégration finale sur une faible surface de silicium et une faible consommation de puissance. Pour vérifier et valider le projet de la puce VLSI avant fabrication, une méthodologie de vérification a été proposée dans cette thèse. Elle nous a également permis de définir le cahier des charges de la puce, ainsi que celui des blocs de base.

Mots clés : Réseaux de neurones, Cellules analogiques CMOS, MLP, Cellules cancéreuses dans le sein

Title Analog CMOS cells for artificial neural networks: application in the classification of breast cancer cells

Abstract The artificial neural networks are particularly interesting for CMOS VLSI (Very Large Scale Integration Complementary Metal-Oxide Semiconductor) implementations because every parallel element (neuron or synapsis) is relatively simple, allowing the complete integration of big networks on a single chip). Multipliers, non-linear function and its derivative are essential key elements in the analog signal processing in particular for analog VLSI implementation of artificial neuronal networks. The main conditions of this kind of circuits are the following ones: a low surface of Silicon and a low electric consumption. To validate our approach, we chose as type of application, the classification of cancer cells (malignant or benign) of the breast. There are many types of neural networks: Feed-forward neural network with back propagation (MLP), Radial Basis Network (RBN), Recurrent Neural Network (RNN) and other. The neural network studied in this thesis is based on Multi-Layer Perceptron with back-propagation (MLP).
The main objective is to find the best compromises and the optimizations to realize circuits in a mature STMicroelectronics HCMOS9A 130nm technology and supplied with ± 900mV to

have the lowest cost. Having chosen the best algorithm (the simplest and most effective) as a simple VLSI implementation, we defined efficient analog architecture. Finally building blocks were designed and realized before the final integration on a low surface of silicon and low power consumption. To verify and validate the project of the VLSI chip before manufacturing, a methodology of check was proposed in this thesis. It also allowed us to define the specifications of the full chip, as well as that of the building blocks.

Keywords : Artificial neural networks, Analog CMOS, MLP, Breast Cancer

Avant-propos

Ma gratitude, mon profond respect et mes remerciements à l'ensemble des membres de mon jury qui m'ont fait l'honneur de participer à mon jury de thèse, les Professeurs: Gilles Jacquemod, Adnan Harb, Yann Deval, Mohamad Sawan, Yves Leduc et Luc Hébrard.

Je voudrais tout d'abord remercier grandement mon directeur de thèse, Monsieur Gilles Jacquemod, Professeur à l'université Nice Sophia Antipolis, pour toute son aide. Je suis ravi d'avoir travaillé en sa compagnie car outre son appui scientifique, il a toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse. Cette section de remerciements est à mon avis la plus satisfaisante à écrire de toute ma thèse. J'ai beaucoup de reconnaissance et d'admiration à témoigner d'abord à mon cher directeur de recherche, le Professeur Gilles Jacquemod, qui est un modèle professionnel pour sa gestion calme de son temps pourtant si chargé de responsabilités importantes et bien sûr pour sa maîtrise remarquable de notre domaine, à la fois comme ingénieur et comme professeur, pédagogue et comme chercheur.

Merci pour ton ouverture à nos caractères et à notre touche propres en recherche, merci pour la liberté que tu nous laisses en nous témoignant ta confiance dans notre travail. Ce sont cette confiance et cette liberté qui, à mon avis, nous laissent spontanément faire ressortir le meilleur de nous-même dans notre travail et comme personnes. Merci de reconnaître nos efforts et nos difficultés. Merci pour la relation que nous avons pu développer depuis mon entrée à la chaire comme stagiaire, merci aussi de m'avoir dirigé et encouragé à passer au doctorat et d'avoir ainsi joué un rôle majeur dans ma découverte de possibilités de carrière qui me motivent vraiment; merci pour ton efficacité qui facilite toujours le lourd processus de la publication d'articles! Je te remercie également pour ton accueil chaleureux à chaque fois que j'ai sollicité ton aide, ainsi que pour tes multiples encouragements, notamment lors de ma première communication à l'université Nice Sophia Antipolis.

Je tiens à remercier mon co-directeur de thèse, Monsieur Adnan Harb, professeur à l'université Lebanese International University pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses multiples conseils et pour toutes les heures qu'il a consacrées à diriger cette recherche. J'aimerais également lui dire à quel point j'ai apprécié sa grande disponibilité et son respect sans faille des délais serrés de relecture des documents que je lui ai adressés. Enfin, j'ai été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail doctoral. C'est grâce à lui que j'ai pu concilier avec bonheur recherche théorique et appliquée pendant cette thèse. Je remercie encore Professeur Adnan Harb pour son aide précieuse pour ma recherche bibliographique. Elle a toujours fait tout son possible pour m'aider.

Messieurs Yann Deval et Mohamad Sawan Professeurs à l'université INP Bordeaux et à l'université Ecole Polytechnique Montréal respectivement m'ont fait l'honneur d'être rapporteurs de ma thèse, ils ont pris le temps de m'écouter et de discuter avec moi. Leurs remarques m'ont permis d'envisager mon travail sous un autre angle. Pour tout cela je les remercie.

Je tiens à remercier Professeur Yann Deval pour avoir accepté de participer à mon jury de thèse et pour sa participation scientifique ainsi que le temps qu'il a consacré à ma recherche.

Merci aussi au Professeur Mohamad Sawan, c'est un honneur de vous avoir comme membre externe sur ce jury, merci pour vos visites, pour votre intérêt envers ma recherche et pour les relations amicales que vous entretenez avec notre chaire de recherche, Je souhaiterais remercier mes rapporteurs pour le temps qu'ils ont accordé à la lecture de cette thèse et à l'élaboration de leur rapport.

Je tiens à remercier Monsieur Yves Leduc, Professeur à l'Université de Nice Sophia Antipolis, qui m'a encadré tout au long de cette thèse et qui m'a fait partager ses brillantes intuitions. Qu'il soit aussi remercié pour sa gentillesse, sa disponibilité permanente et pour les nombreux encouragements qu'il m'a prodigués.

J'exprime ma gratitude à Monsieur Luc Hébrard, Professeur à l'Université de Strasbourg, qui a bien voulu être examinateur, et Je le remercie également Professeur Luc Hébrard d'avoir accepté avec beaucoup d'élan d'être Président de mon jury de thèse.

Ce travail n'aurait pas été possible sans le soutien de l'Université Lebanese International University et de l'université Nice Sophia Antipolis, qui m'ont permis, grâce à une allocation de recherches et diverses aides financières, de me consacrer sereinement à l'élaboration de ma thèse.

Je remercie tout particulièrement mon laboratoire d'accueil, Polytech'Lab et LIU Lab, ainsi que ses responsables qui m'ont permis de m'intégrer rapidement et de réaliser mon projet.

Mes derniers remerciements et non les moindres, s'adressent à ma femme Professeur Sarya Chahine et à ma petite fille Farah, qui, pour mon plus grand bonheur partage ma vie et mes expérience professionnelles depuis leurs origines. Elle a su, tout au long de cette thèse, et m'encourager dans ma voie. Elle est la clef de ma réussite, sans elle à mes côtés, cette réalisation n'aurait pas la même saveur.

Table des Matières

Glossaire

Pour faciliter la lecture du manuscrit, nous rappelons dans le tableau ci-dessous les acronymes les plus utilisés. Ils sont également décrits lors de leur première utilisation dans ce mémoire.

ANN	Artificial Neural Network ou **RNA** : Réseau de Neurones Artificiels
AE	Amplitude Error ou **EA** : Erreur Amplitude
AVNN	Analog VLSI Neural Network
BCDB	Breast Cancer Data Base
BP	Back Propagation
DE	Destination Error ou **ED** : Erreur de Destination
DVNN	Digital VLSI Neural Network
IA	Artificial Intelligence ou **IA** : Intelligence Artificielle
I-V	I-V Converter : Convertisseur Courant-Tension
LTM	Long Term Memory
MCPS	Mega Connections Per Second
MLP	Multi-Layer Perceptron
UCI	University of California at Irvine
WBC	Wisconsin Breast Cancer
WDBC	Wisconsin Diagnostic Breast Cancer
WE	Wrights Error ou **EP** : Erreur du poids

Introduction générale

1. Introduction

Les réseaux de neurones artificiels (RNA ou ANN : Artificial Neural Network) sont particulièrement intéressants pour les implémentations CMOS VLSI car chaque élément parallèle (neurone ou synapse) est relativement simple, permettant l'intégration complète de grands réseaux sur une seule puce. Les multiplieurs, la fonction non-linéaire et sa dérivée sont des éléments clés essentiels dans le traitement du signal analogique et notamment dans la mise en œuvre VLSI analogique de réseaux neuronaux artificiels. Afin d'implanter un nombre très élevé de neurones sur une même puce, les principales contraintes de ce circuit sont les suivantes : une faible surface de Silicium et une faible consommation électrique.

Pour valider notre approche, nous avons choisi comme type d'application, la classification de cellules cancéreuses (malignes ou bénignes) du sein. Le travail s'appuie sur de nombreux travaux présentés dans la littérature. L'objectif principal est de trouver les meilleurs compromis et optimisations pour réaliser des circuits dans une technologie mature CMOS 130 nm afin d'avoir le coût le plus faible possible. Le réseau de neurones étudié dans cette thèse est basé sur l'architecture MLP (Multi-Layer Perceptron) [1], [2], [3], dont la phase d'apprentissage utilise la technique de rétro-propagation (BP) [3], [4], [5]. Nous détaillerons ce processus dans le second chapitre.

Ce mémoire décrit mon activité et mon travail pendant le doctorat dans le Laboratoire Polytech'Lab de l'Université de Nice Sophia Antipolis, France et à l'Université Internationale du Liban. L'étude, l'analyse et la conception de la mise en œuvre VLSI analogique sont les principaux objectifs de cette thèse. Une implémentation de puce VLSI du réseau neuronal est étudiée et conçue. Les trois étapes suivantes sont utilisées :

1) Choisir le meilleur algorithme (le plus simple et efficace) pour une implémentation VLSI simple ;
2) Définir une architecture analogique efficiente ;
3) Concevoir et réaliser les briques de base, puis le circuit analogique final sur une faible surface de silicium et une faible consommation de puissance.

Pour vérifier et valider le projet de la puce VLSI avant fabrication, une méthodologie de vérification est proposée dans cette thèse. De plus, des simulations au niveau des transistors sont également présentées. Des polarisations de transistors en faible inversion seront privilégiées afin de garantir

une basse tension d'alimentation, à savoir 1,8V (±0,9V) et une dynamique suffisante. La conception sera ainsi réalisée dans un objectif de faible consommation et basse tension.

2. Objectifs et motivation

Les avancées des performances réalisables par Circuits Intégrés (IC) ont suscité un grand intérêt dans la mise en œuvre VLSI de Réseau de Neurones Artificiels (ANNs). Selon le style de conception du circuit, les réseaux neuronaux VLSI (VNNs) pourraient être subdivisés en deux catégories principales : numérique et analogique.

Les NN numériques VLSI (DVNN) utilisent et fonctionnent sur des signaux numériques à intervalles de temps discrets. Dans le cas des DVNN, l'apprentissage en temps réel sur puce n'est pas nécessaire. En pratique, les contraintes de surface et de vitesse du silicium limitent la faisabilité des DVNN. Leur implémentation est souvent réalisée sur des circuits de types FPGA ou mixtes FPGA et microcontrôleurs, nécessitant alors un partitionnement matériel-logiciel. Les architectures neuronales sans apprentissage semblent être les applications les mieux adaptées aux DVNN.

Les NN analogiques VLSI (AVNN) utilisent des signaux classés dans leur état et continus dans le temps. Entre ces deux extrêmes, de nombreuses implémentations de NNS existent. Certains utilisent les deux mécanismes pour le codage de l'information en utilisant des circuits qui fonctionnent sur des variables électriques numériques (par exemple pour le stockage des poids synaptiques), d'autres fonctionnent sur des variables électriques analogiques (par exemple pour les calculs anticipés).

Les AVNN permettent de concevoir des systèmes neuronaux structurés inspirés par la biologie mais motivés par le désir de rendre efficaces les moteurs de calcul analogiques. L'intérêt pour les AVNN a trois motivations majeures :

1) La puissance de calcul des NNs biologiques dérive non seulement du parallélisme massif mais aussi du traitement analogique [6];
2) Le plein potentiel de la technologie du silicium peut être mieux exploité en utilisant la physique des dispositifs pour faire le calcul [7];
3) La possibilité d'imiter les fonctions des neurones biologiques et des réseaux [8], [9], [10].

Par exemple, R.R. Coggins et al. ont conçu une classification d'électrocardiogrammes inter-cardiaques de basse puissance en utilisant des circuits VLSI analogiques ayant dix entrées et trois sorties, correspondantes à trois classes [11]. R.L. Yaeger a conçu un réseau de neurones pour la reconnaissance en ligne de l'écriture à la main [12]. Un état de l'art sur l'implémentation matérielle des réseaux neuronaux est présenté au chapitre 2.

Parmi les implémentations matérielles possibles de réseaux de neurones (circuits dédiés, numériques ou analogiques), certains auteurs [6], [13], [14] ont mis en évidence les avantages (i.e. faible consommation d'énergie, etc.) de l'approche VLSI analogique et la possibilité de mettre en œuvre un système suffisamment grand, efficace et rapide pour être utilisé dans des applications réelles. En effet, l'approche VLSI analogique offre des circuits simples et efficaces. Par exemple, la somme des signaux est obtenue en exploitant la loi de courant de Kirchhoff (KCL), la multiplication de deux signaux est réalisée en utilisant un simple OTA avec une cellule de polarisation, le coefficient de pondération (poids) peut être mémorisé dans un simple condensateur, etc.

L'apprentissage peut être réalisé à l'intérieur de la puce (apprentissage sur puce) ou en dehors de la puce (apprentissage hors puce). Dans le premier cas, nous devons implémenter le calcul anticipé et le calcul rétrograde, dans le dernier cas nous avons seulement besoin d'implémenter les calculs anticipés. L'avantage de la deuxième approche est de diminuer la taille du circuit (i.e. la surface de Silicium), mais son principal inconvénient est la complexité de réaliser l'apprentissage par un ordinateur hôte qui calcule les poids. Un autre inconvénient réside dans le fait que le réseau mis en œuvre par ordinateur n'est pas similaire à celui sur silicium et que, par conséquent, les pondérations n'incluent pas les effets non idéaux des différents circuits. Certains auteurs proposent de faire l'apprentissage de la puce dans la boucle, ce qui signifie que le calcul anticipé est exécuté par la puce pendant que l'ordinateur hôte exécute le calcul arrière. L'avantage de cette méthode est de réduire l'effet négatif obtenu par l'apprentissage hors puce. Toutefois les exigences de l'ordinateur hôte et du système complet limitent la faisabilité de cette méthode.

Sur la base des inconvénients présentés par les différentes techniques d'apprentissage, celui sur puce semble être la meilleure méthode à suivre. L'inconvénient principal est l'augmentation de la surface de silicium. Les avantages de l'apprentissage sur puce sont d'obtenir un apprentissage plus rapide [15], [16] et de simplifier le système qui le gère par rapport aux deux autres méthodes. Nous avons donc retenu cette méthode pour notre approche. Les principaux objectifs de la thèse sont énumérés ci-dessous :

1) Exploitation des principales règles à l'intérieur du paradigme neuronal computationnel ;
2) Définitions, modèles et analyses des primitives de calcul neuronales ;
3) Evaluation et choix de l'algorithme de classification des cellules cancéreuses ;
4) Étude de l'algorithme de propagation arrière et de certaines variétés pour trouver le plus approprié pour la mise en œuvre du matériel ;
5) Etude, conception et implémentation VLSI analogique des systèmes neuronaux ;
6) Etude et conception des briques de base ;

7) Optimisation des cellules élémentaires avec étude des transistors en faible inversion pour obtenir une faible consommation d'énergie et réaliser des fonctions plus complexes ;
8) Conception des circuits VLSI avec une faible surface de silicium pour implémenter un grand réseau de neurones dans une puce et réduire le coût ;
9) Conception d'une puce VLSI pour la mise en œuvre d'un algorithme MLP « formé » (ou phase d'apprentissage) par BP.

La conception des différentes briques de base et du circuit final doit viser également les objectifs suivants :

1) Implémenter le plus d'opérations requises dans un système de réseau neuronal artificiel ;
2) Implémenter les modules de neurones dans le perceptron multicouche formé par l'algorithme de propagation arrière en utilisant la fonction non linéaire proposée et ses dérivées ;
3) Implémenter les modules de synapses dans le perceptron multicouche formé par l'algorithme de propagation arrière en utilisant les circuits multiplieurs ;
4) Implémenter exactement (plus en détails) les opérations d'avance et de retour dans les neurones et les modules de synapses ;
5) Concevoir une puce VLSI en utilisant les modules de neurones et de synapses ;
6) Utiliser la puce dans des applications réelles telles que les réseaux de neurones artificiels pour différentes applications autres que la classification des cellules cancéreuses, comme les problèmes de vision, la reconnaissance de caractères (lettres, chiffres ...), le traitement d'image ou du son, ….

Les objectifs et le but principal du projet de recherche est d'obtenir un microprocesseur analogique au lieu d'un microprocesseur numérique afin de résoudre de vrais problèmes de vision, de reconnaissance de formes et d'autres problèmes de résolution d'intelligence artificielle. Dans notre recherche, le perceptron multicouche formé par l'algorithme de propagation arrière (ou rétro-propagation, BP) sera pris en considération, puis nous devons:

1) Obtenir des fonctions non linéaires comme la fonction tangente hyperbolique ou la fonction sigmoïde (ou logsigmoid) qui implémente les opérations du module neuronal requises par le système de réseau neuronal ;
2) Obtenir la dérivée des fonctions non linéaires qui implémente la dérivée du module neuronal pour calculer l'erreur ;
3) Préparer le module pour la conception de micro-puces qui contient le système de réseau neuronal ;
4) Valider le système complet dans un cas particulier, à savoir la classification de cellules cancéreuses ou non dans le sein.

3. Travaux de Thèse

Le présent mémoire est composé de 5 chapitres encadrés par une introduction et une conclusion générale présentant également quelques perspectives, ainsi des annexes. Dans le premier chapitre sont présentés quelques éléments sur le cancer du sein, avant de décrire la base de données qui nous a servi de référence. En particulier, la comparaison des caractéristiques entre une tumeur bénigne et une tumeur maligne permettra d'en réaliser la classification. Cette classification est réalisée à partir de 9 attributs extraits des images (biopsies) de mammographies, décrits dans la base de données Wisconsin. La seconde partie de ce chapitre dresse un état de l'art de l'implémentation matérielle des réseaux de neurones. Enfin, nous proposons une méthodologie pour réaliser notre travail de thèse, à savoir la conception et la réalisation d'un réseau de neurones à partir de cellules CMOS analogiques. Cette étude nous permet de définir les briques de base pour réaliser un neurone, à savoir un multiplieur, une fonction d'activation non-linéaire et sa dérivée. Le cahier des charges du circuit final et de chaque élément sera réalisé au chapitre 3, par une analyse système de l'algorithme retenu et de sa robustesse aux erreurs de précision, de bruit, …

Le second chapitre est dédié à l'analyse des réseaux de neurones artificiels avec dans un premier temps sa comparaison avec le cerveau humain. Nous rappellerons ainsi quelques définitions et propriétés des RNA, ainsi que leurs domaines d'applications. Nous détaillerons l'architecture du réseau de neurones multicouches Perceptron (MLP) avec propagation avant (pour la classification) et rétro-propagation (pour la phase d'apprentissage). Les différents algorithmes (calculs pondérés) sont également présentés. La problématique de ce travail de thèse est ainsi posée, il convient maintenant de définir l'architecture de RNA la plus efficiente pour notre application.

Dans le troisième chapitre, une étude sous Matlab nous a permis de définir d'une part l'architecture du RNA et d'autre part ses spécifications. Plusieurs algorithmes ont été étudiés dans le cadre de notre application. La base de données Wisconsin a servi de références pour évaluer les performances de ces différents algorithmes. Le RNA retenu comprend 9 neurones d'entrée, 10 dans la couche cachée et 2 en sortie (noté 9-10-2). Les fonctions d'activation tangente hyperbolique ou surtout logsigmoid sont retenues. En outre, il semblerait que le fait de supprimer le biais ne diminue pas la qualité du système de classification, voire l'améliore légèrement. Enfin, nous avons évalué, toujours sous Matlab, les spécifications des différents blocs, à savoir le multiplieur et la fonction d'activation. Nous avons ainsi déterminé leur dynamique et leur précision (bruit ou erreur maximale). Ces spécifications serviront de cahier des charges pour dimensionner les blocs de base du RNA.

Le quatrième chapitre est dédié à la conception et à l'optimisation des blocs de base du RNA, à savoir le multiplieur, la fonction d'activation (tangente hyperbolique ou logsigmoid) et sa dérivée. Une part importante de ce chapitre concerne le multiplieur, bloc sensible du circuit final. Deux topologies sont retenues, et seront utilisées en fonction de leurs dynamiques d'entrées respectives afin d'optimiser la consommation totale du RNA. Le multiplieur à large bande, ayant une consommation plus importante, sera utilisé uniquement lorsqu'une des entrées correspond aux poids (W_{ij}) ou coefficients. Dans les autres cas, nous utiliserons une cellule de Gilbert. Suivant les topologies retenues pour ces différents blocs de base, les transistors seront polarisés en faible ou forte inversion afin de réaliser les fonctions désirées. Enfin, le dimensionnement des transistors permettra d'atteindre les spécifications de chaque bloc tout en optimisant leur consommation respective.

Le cinquième et dernier chapitre concerne la mise en œuvre du circuit RNA complet. La première étape consiste à réaliser la mémorisation et la mise à jour des poids, ainsi que la réalisation du bloc Delta qui calcule l'erreur entre la sortie d'un neurone et la valeur désirée. Cette phase correspond au processus d'apprentissage qui fixe la valeur des poids en utilisation l'algorithme de rétro-propagation. La topologie du RNA final est alors réalisée à partir des différents blocs de base puis validée par des simulations Spice au niveau transistor. Les résultats montrent que le taux d'erreur global atteint une valeur de 2,4% pour un RNA sans biais avec une fonction d'activation Logsigmoid, équivalent à celui obtenu sous Matlab qui nous a servi de valeur de référence (avec la fonction d'activation SOFTMAX), et plus faible que celui obtenu toujours sous Matlab (2,6%, avec une fonction d'activation Logsigmoid). Ces résultats permettent de valider le choix du RNA, son cahier des charges, ainsi que les spécifications et le dimensionnement des briques de base.

Enfin, lors de la conclusion générale, nous proposons quelques perspectives de ce travail, à commencer par la réalisation du dessin des masques du circuit final, en vue de sa fabrication et de son test. L'étape suivante consisterait à proposer une meilleure solution pour la sauvegarde et la mise à jour des poids par un circuit à capacités commutées plus performant ou l'utilisation de mémoires non-volatiles. L'étape ultime serait également de proposer une solution reconfigurable permettant de s'adapter au problème à résoudre, autre que celui de la classification de cellules cancéreuses.

4. Bibliographie

[1] J. Hertz, A. Kough and R. Palmer, "Introduction to the Theory of Neural Computation," in *Addison-Wesley, Reading*, 1991.

[2] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine,* pp. 4-22, ,April,1987.

[3] D. E. Rumelhart and J. L. McClelland, “Parallel distribuited processing,” in *MIT Press*, Cambridge, 1986.

[4] M. Valle, D. D. Caviglia, G. Donzellini, A. Mussi, F. Oddone and G. M. Bisio, “A neural computer based on an analog VLSI neural network,” *Proceedings of ICANN'94, Sorrento, Italy,* vol. 2, pp. 1339-1342, 26-29 May 1994.

[5] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink and D. L. Alkon, “Accelerating the convergence of the back-propagation method,” *Biological Cybernetics,* pp. 257-263, 1988.

[6] C. A. Mead, “Analog VLSI and Neural Systems,” in *Addison-Wesley, Reading,*, 1989.

[7] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE,* vol. 78 (No10), 1990.

[8] C. A. Mead and M. A. Mahowald, “A silicon model of early visual processing,” *Neural Networks,* vol. 1, pp. 91-97, 1988.

[9] M. Mahowald and R. Douglas, “A silicon neuron,” *Nature,* vol. 354, pp. 19-26, Dec 1991.

[10] A. G. Andreou, “Electronic arts imitate life,” *Nature,* vol. 354, pp. 19-26, Dec. 1991.

[11] R. R. Coggins, M. Jabri, B. Flower and S. Pickard, “Low-power intercardiac electogram classification using analog VLSI,” *Proceedings of MicroNeuro'94,* pp. 376-382, 1994.

[12] R. L. a. L. Yaeger, “On-line hand-printing recognition with neural networks,” *MicroNeuro'96,* pp. 201-212, 1996.

[13] Y. P. Tsividis, “Analog MOS integrated circuits - certain new ideas, trends and obstacles,” *IEEE Journal of Solid State Circuits,* Vols. SC-22 (No. 3), pp. 312-317, 1987.

[14] E. A. Vittoz, “Analog VLSI implementations of neural networks,” *Proceedings of ISCAS'90,* pp. 2524-2527, 1990.

[15] G. A. Cairns and L. Tarassenko, “Implementation issues for on-chip learning with analogue VLSI MLPs,” *Artificial Neural Networks, Conference Publication No. 409, IEE,* pp. 465-470, 26-28 June 1995.

[16] M. Valle, D. D. Caviglia and G. M. Bisio, “An experimental analog VLSI neural network with on-chip back-propagation learning,” *Analog Integrated Circuits & Signals Processing, Kluwer Academic Publishers,Boston,* vol. 9, pp. 231-245, 1996.

Chapitre I – Cancer du sein et Réseau de neurones artificiel

1. Introduction

Dans ce chapitre sont présentés quelques éléments sur le cancer du sein, avant de décrire la base de données qui nous a servi de référence. En particulier, la comparaison des caractéristiques entre une tumeur bénigne et une tumeur maligne permettra d'en réaliser la classification. La seconde partie de ce chapitre dresse un état de l'art de l'implémentation matérielle des réseaux de neurones. Enfin, nous proposons une méthodologie pour réaliser notre travail de thèse, à savoir la conception et la réalisation d'un réseau de neurones à partir de cellules CMOS analogiques.

2. Cancer du sein

2.1. Introduction

Dans le sein d'une femme, il y a 15 à 20 sections appelées lobes, comme le montre la figure 1.1 [1]. Chaque lobe contient de nombreuses petites sections appelées lobules. Ce sont des groupes de minuscules glandes produisant le lait maternel, qui coule à travers de minces tubes, appelés conduits, au mamelon. La graisse et les autres tissus remplissent les espaces entre les lobules et les canaux. Les seins contiennent également des vaisseaux lymphatiques, qui sont connectés à de petites masses rondes de tissus appelées ganglions lymphatiques. Les ganglions lymphatiques produisent des cellules qui aident le corps à combattre l'infection. Des groupes de ganglions lymphatiques se trouvent près des seins dans les aisselles, au-dessus de la clavicule et dans la poitrine, derrière le sternum.

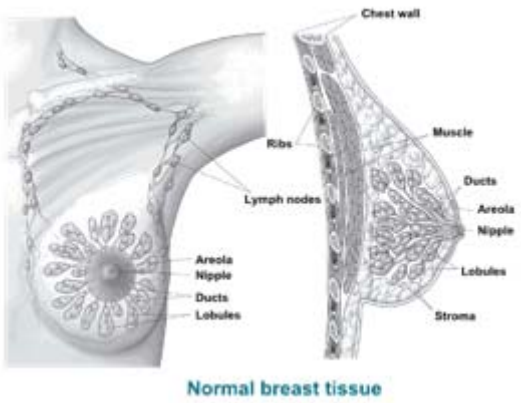


Figure 1.1 : Vue de face et de profil su sein

Les lobes et les canaux des seins, et les ganglions lymphatiques voisins (cf. Figure 1.1) sont des zones que le cancer peut attaquer.

2.2. Cellules cancéreuses

2.2.1. Introduction

Les cellules se développent et se divisent pour former de nouvelles cellules. Lorsque les cellules normales vieillissent ou sont endommagées, elles meurent et de nouvelles cellules prennent leur place. Parfois, de nouvelles cellules se forment lorsque le corps n'en a pas besoin, et les cellules anciennes ou endommagées ne meurent pas comme elles le devraient. Les cellules supplémentaires forment souvent une masse de tissu appelée masse, croissance ou tumeur. Les tumeurs du sein peuvent être bénignes (non cancéreuses) ou malignes (cancer) [1].

2.2.2. Dépistage du cancer

Le cancer du sein commence lorsque les cellules du sein commencent à échapper à tout contrôle. Ces cellules forment généralement une tumeur qui peut souvent être vue sur une radiographie ou ressentie comme une masse [2]. La tumeur est maligne (cancer) si les cellules peuvent se développer (envahir) les tissus environnants ou se propager (métastases) à des zones éloignées du corps. Le cancer du sein survient presque entièrement chez les femmes, mais les hommes peuvent aussi contracter le cancer du sein.

Pour éviter cette maladie qui coûte tant de vies, les applications informatiques diagnostiques sont largement utilisées. En outre, les systèmes experts sont présents dans les salles de traitement intensif. À son tour, l'utilisation d'un autre aspect de l'intelligence artificielle pour le diagnostic du cancer du sein constitue une réelle avancée. Il est rapporté que la maladie du cancer du sein est le deuxième cancer le plus commun qui affecte les femmes, et était le cancer prévalent dans le monde en l'an 2002 [3]. Macmillan Cancer Support à Londres rapporte qu'au Royaume-Uni, le cancer du sein affecte un nombre important de femmes et un petit nombre d'hommes. Aux États-Unis, environ une femme sur huit au cours de sa vie a un risque de développer un cancer du sein [4]. Le dépistage, le plus en amont possible, reste le meilleur rempart contre cette maladie.

Le cancer du sein commence par la division incontrôlée d'une cellule à l'intérieur du sein et aboutit à une masse visible, appelée tumeur. La tumeur peut être bénigne ou maligne. Le diagnostic précis pour déterminer si la tumeur est bénigne ou maligne peut sauver des vies. Par conséquent, la nécessité d'une classification précise au sein de la clinique est une source de grande préoccupation pour les spécialistes et les médecins. L'importance de l'intelligence artificielle a été motivée au cours

des 25 dernières années, lorsque les scientifiques ont commencé à prendre conscience de la complexité de prendre certaines décisions pour traiter certaines maladies. L'utilisation de l'apprentissage et de l'exploration automatique de données comme outils dans le diagnostic médical devient très efficace. La tâche de classification joue un rôle essentiel pour le diagnostic du cancer du sein. Par conséquent, les techniques d'apprentissage automatique peuvent aider les médecins à faire un diagnostic précis du cancer du sein et à faire la classification correcte d'une tumeur bénigne ou maligne. Il ne fait aucun doute que l'évaluation des données du patient et les décisions des médecins et des spécialistes sont les facteurs les plus importants dans le diagnostic, mais les systèmes experts et les techniques d'intelligence artificielle telles que l'apprentissage automatique pour les tâches de classification aident grandement les médecins et les spécialistes dans leur décision finale.

2.2.3. Où commence le cancer du sein ?

Les cancers du sein peuvent commencer à partir de différentes parties du sein [2]. La plupart des cancers du sein se développent dans les canaux qui transportent le lait vers le mamelon (cancers canalaires). Certaines tumeurs naissent dans les glandes qui produisent le lait maternel (cancers lobulaires). Il existe également d'autres types de cancer du sein moins fréquents. Un petit nombre de cancers commencent dans d'autres tissus du sein. Ces cancers sont appelés sarcomes et lymphomes et ne sont pas vraiment considérés comme des cancers du sein.

Bien que de nombreux types de cancer du sein puissent causer une grosseur dans le sein, tous ne le font pas. De nombreux cancers du sein sont détectés lors des mammographies de dépistage qui permettent de les détecter à un stade précoce, souvent avant qu'ils ne puissent être ressentis, et avant que les symptômes ne se développent. Il y a d'autres symptômes du cancer du sein que les patientes doivent surveiller et signaler à leur médecin.

Une mastose est une pathologie bénigne (et donc non cancéreuse ou maligne) du sein fréquente puisqu'elle est observée chez 50% à 80% des femmes âgées de 30 à 50 ans. Il s'agit d'une affection du sein provoquant différents types de lésions : kystes ou éléments de fibroses. Les tumeurs mammaires non cancéreuses sont des tumeurs anormales, mais elles ne se propagent pas à l'extérieur du sein et ne mettent pas la vie en danger. Toutefois, certaines tumeurs, bien que bénignes, peuvent augmenter le risque d'avoir un cancer du sein. Toute grosseur ou modification du sein doit être vérifiée par un professionnel de la santé afin de déterminer si elle est bénigne ou maligne (cancer) et si elle pourrait affecter votre futur risque de cancer.

Ainsi, les tumeurs bénignes sont rarement une menace pour la vie, peuvent être retirées et ne repoussent généralement pas. Elles n’envahissent pas les tissus autour d'elles et ne se propagent pas

à d'autres parties du corps [1]. A l'inverse, les tumeurs malignes peuvent être une menace pour la vie ; elles peuvent souvent être retirées, mais repoussent parfois. Elles peuvent envahir et endommager les organes et les tissus avoisinants (comme la paroi thoracique) [1] et donc peuvent se propager à d'autres parties du corps.

Les cellules cancéreuses du sein peuvent se détacher de la tumeur initiale et pénétrer dans les vaisseaux sanguins ou les vaisseaux lymphatiques qui se ramifient dans tous les tissus du corps. Les cellules cancéreuses peuvent se propager aux ganglions lymphatiques près du sein, ou elles peuvent se fixer à d'autres tissus, se développant en de nouvelles tumeurs dommageables.

2.2.4. Propagation du cancer du sein

Le système lymphatique est un réseau de vaisseaux lymphatiques situés dans tout le corps qui relie les ganglions lymphatiques (petites collections de cellules immunitaires en forme de haricot) [2]. Le liquide clair à l'intérieur des vaisseaux lymphatiques, appelé lymphe, circule dans tout le corps pour éliminer les déchets, les bactéries et d'autres substances des tissus. Les vaisseaux lymphatiques transportent le liquide lymphatique loin du sein. Dans le cas du cancer du sein, les cellules cancéreuses peuvent pénétrer dans ces vaisseaux lymphatiques et commencer à se développer dans les ganglions lymphatiques. La plupart des vaisseaux lymphatiques du sein se drainent dans:

• Les ganglions lymphatiques sous le bras (ganglions axillaires)

• Les ganglions lymphatiques autour de la clavicule (nœuds lymphatiques supra-claviculaires [au-dessus de la clavicule] et infra-claviculaires [sous l'os de la clavicule])

• Les ganglions lymphatiques à l'intérieur de la poitrine près de l'os de la poitrine (ganglions lymphatiques mammaires internes).

2.2.5. Facteurs de risque et symptômes [1]

Personne ne sait ce qui cause le cancer du sein. Les facteurs de risque de cancer du sein comprennent l'âge, les antécédents personnels et familiaux, les modifications génétiques, la radiothérapie antérieure, les antécédents génésiques et menstruels, la race, la densité mammaire, le surpoids et l'obésité, l'inactivité physique et la consommation d'alcool. Vous pouvez éviter certains facteurs de risque, tels que la consommation d'alcool. Avoir un facteur de risque ne signifie pas que vous aurez un cancer du sein. La plupart des femmes présentant des facteurs de risque ne développent, fort heureusement, jamais de cancer du sein.

Le cancer du sein précoce ne provoque habituellement pas de symptômes. Mais au fur et à mesure que la tumeur se développe, elle peut changer la façon dont le sein ressemble ou se sent (lors d'une palpation par exemple), y compris :

- Une grosseur ou un épaississement dans ou près de la poitrine ou des aisselles
- Un changement dans la taille ou la forme du sein.
- Meulage ou plissement de la peau de la poitrine. La peau peut être striée ou piquée comme une orange
- Un mamelon tourné vers l'intérieur dans le sein
- Décharge liquide du mamelon, surtout s'il est sanglant
- Peau squameuse, rouge ou enflée sur le sein, le mamelon ou l'aréole (la zone sombre de la peau au centre de la poitrine).

2.3. Base de données de Wisconsin sur le cancer du sein

2.3.1. Introduction

La base de données originale de Wisconsin sur le cancer du sein [5] de l'UCI Référentiel d'apprentissage automatique est utilisée dans cette étude. Le cancer-du-sein-Wisconsin compte 699 cas répartis en 2 classes :

- Bénin : 458 cas soit 65,5%
- Malin : 241 cas soit 34,5%

Chaque cas est caractérisé par 11 attributs (de valeur entière comprise entre 1 et 10), qui seront détaillés ultérieurement. Cette base de données est disponible via le serveur ftp UW CS, à l'adresse suivante : http://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/WDBC/.

On trouvera également un référentiel UCI d'apprentissage automatique à l'adresse ci-dessous :

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29 [6].

Le statut ganglionnaire correspond au nombre de ganglions lymphatiques auxiliaires positifs observés au moment de la chirurgie. Les ganglions lymphatiques dans l'aisselle (les ganglions lymphatiques axillaires) sont le premier endroit où le cancer est susceptible de se propager, comme le montre la figure 1.2 [2].

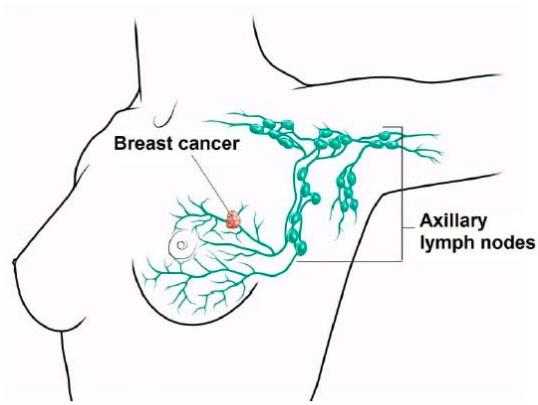


Figure 1.2 : Ganglions lymphatiques axillaires près d'une poitrine avec cancer

Le statut ganglionnaire est fortement lié au pronostic. Nymphe-négatif signifie que les ganglions lymphatiques ne contiennent pas de cancer, alors que le terme ganglionnaire signifie que les ganglions lymphatiques contiennent un cancer.

2.3.2. Description de la base de données [6]

L'ensemble de données sur le cancer du sein du Wisconsin provenant de la base de données d'apprentissage automatique de l'UCI est utilisé pour distinguer les échantillons malins (cancéreux) des échantillons bénins (non cancéreux). Une brève description de cet ensemble de données est présentée dans le tableau 1.1 [6]. L'ensemble de données consiste en quelques modèles de classification ou se compose de certains modèles de classification ou d'instances avec un ensemble de caractéristiques numériques ou d'attributs.

Tableau 1.1 : Description de l'ensemble de données sur le cancer du sein

Ensemble de Données	Nb. d'Attributs	Nb. de Cas	Nb. de Classes
Cancer de Sein de Wisconsin (Original)	11	699	2

Sur les 11 attributs, un seul correspond à la sortie (0 pour une tumeur bénigne et 1 pour une tumeur maligne) et 10 pour les entrées. Sur ces 10 attributs en entrée, seulement 9 seront utilisés par notre système de classification. Ils sont détaillés au paragraphe suivant et résumés dans le tableau 1.3.

Des échantillons arrivent périodiquement lorsque Dr Wolberg signale ses cas cliniques [6]. La base de données reflète donc ce regroupement chronologique des échantillons. Ces informations de regroupement chronologiques sont répertoriées dans le tableau 1.2.

Tableau 1.2 : Distribution chronologique des 699 cas

Groupe	Cas	Date
1	367	Janvier 1989
2	70	Octobre 1989
3	31	Février 1990
4	17	Avril 1990
5	48	Août 1990
6	49	Mis à jour Janvier 1991
7	31	Juin 1991
8	86	Novembre 1991
Total	699	À partir de la base de données donnée le 15 Juillet 1992

2.3.3. Description des attributs

Dans l'épaisseur de l'amas, les tumeurs malignes ont tendance à se regrouper sous formes d'amas multicouches alors que les cellules bénignes ont tendance à être groupées en monocouches. Concernant le couple (uniformité de la taille et forme des cellules), les cellules cancéreuses ont tendance à varier en taille et en forme. C'est la raison pour laquelle ces paramètres sont utiles pour déterminer si les cellules sont cancéreuses ou non. Dans le cas de l'adhérence marginale, les cellules normales tendent à rester collées entre elles, alors que les cellules cancéreuses ont tendance à perdre cette capacité. Ainsi, la perte d'adhérence est également un signe de malignité. Quant à la taille des cellules épithéliales simples, elle est liée à l'uniformité mentionnée ci-dessus. Les cellules épithéliales, qui sont significativement élargies, peuvent correspondre à une cellule maligne. Le noyau nu est un terme utilisé pour les noyaux qui ne sont pas entourés de cytoplasme (reste de la cellule). Ceux-ci sont généralement observés dans les tumeurs bénignes. La chromatine fade décrit une "texture" uniforme du noyau observé dans les cellules bénignes. Dans le cas des cellules cancéreuses, la chromatine a tendance à être plus grossière. Les nucléoles normaux sont de petites structures observées dans le noyau. Pour les cellules normales, ce nucléole est généralement très petit s'il est visible. Inversement pour les cellules cancéreuses, les nucléoles deviennent plus proéminents, et sont parfois plus nombreux. Enfin, les mitoses correspondent à une division nucléaire des cytokines et produisent deux cellules filles identiques pendant la prophase. C'est le processus dans lequel la cellule se divise et se réplique. Les pathologistes peuvent déterminer le degré d'avancement du cancer en comptant le nombre de mitoses. L'ensemble de ces 9 attributs est résumé dans le tableau 1.3. Les neuf attributs sont classés sur une échelle d'intervalle de 1 à 10, 1 étant la plus proche de bénigne et 10 la plus proche de maligne.

Tableau 1.3 : Attributs de la base de données de Wisconsin sur le cancer du sein

No. Identité	Attribut	Valeurs extrêmes de 1=Bénin à 10=Malin
1	Epaisseur de l'amas	1=Groupés en monocouche, 10=Multicouches
2	Uniformité de la taille de la cellule	1=Très uniforme, 10=Taille variable
3	Uniformité de la forme cellulaire	1=Très uniforme, 10=Peu uniforme
4	Adhésion marginale	1=Forte adhésion, 10=Perte de cette capacité
5	Taille de cellule épithéliale unique	1=Taille uniforme, 10=Taille élargie
6	Noyaux nus	1=Présence, 10=Absence de noyaux nus
7	Chromatine fade	1=Texture uniforme, 10=Texture grossière
8	Nucleole normal	1=Petit, 10=Plus nombreux et plus gros
9	Mitoses	1=Absence, 10=Présence (Cancer incontrôlé)

La valeur de chaque attribut est calculée à partir de 10 caractéristiques déterminées pour chaque cellule (chaque image de tumeur de la base de données contenant 699 cas) [2] :

a) Rayon (moyenne des distances du centre aux points du périmètre)
b) Texture (écart-type des valeurs d'échelle de gris)
c) Périmètre
d) Surface
e) Régularité (variation locale des longueurs de rayon)
f) Compacité (périmètre ^ 2 / surface - 1.0)
g) Concavité (gravité des parties concaves du contour)
h) Points concaves (nombre de parties concaves du contour)
i) Symétrie
j) Dimension fractale («approximation de la côte» - 1)

Toutes les valeurs de ces caractéristiques sont enregistrées avec 4 chiffres significatifs. La base de données est complète et fournit en sortie une valeur binaire (2 classes) :

- 0 : tumeur bénigne
- 1 : tumeur maligne

2.4. Diagnostic

2.4.1. Introduction

Le cancer du sein représente l'une des maladies qui provoque un nombre élevé de décès chaque année. C'est le type le plus commun de tous les cancers et la principale cause de décès des femmes dans le monde entier. La deuxième cause majeure de décès des femmes est le cancer du sein (après le cancer du poumon). De nombreuses techniques ont été proposées pour le diagnostic des cas de cancer du sein ainsi que pour le pronostic de la maladie [7], [1], [8], [9], [10], [11], [12] et [13]. La méthode qui peut confirmer la malignité avec une sensibilité de haut niveau est la biopsie chirurgicale. Toutefois, elle est considérée comme une opération coûteuse et a un impact négatif sur

la psychologie du patient. Ces considérations ont privilégié les techniques d'apprentissage automatique visant à fournir les mêmes niveaux de précision, sans les aspects négatifs de la biopsie chirurgicale.

Le rôle du diagnostic est de faire la distinction entre les tumeurs mammaires malignes et bénignes. Dans le cas où un cancer du sein est diagnostiqué chez une patiente, la masse maligne doit être excisée. Un suivi de la patiente est ensuite obligatoire afin de réaliser une prédiction de l'évolution de la maladie. Cependant, la prédiction pronostique n'appartient pas non plus aux paradigmes d'apprentissage classiques de l'approximation ou de la classification des fonctions. Ceci est dû au fait qu'un patient peut être classé comme un cas «récurrent» si la maladie est observée, alors qu'il n'y a pas de seuil auquel le patient peut être considéré comme un cas «non récurrent». Les données sont donc censurées puisqu'une période de récurrence n'est connue que pour un sous-ensemble de patients. Par conséquent, l'ensemble de données d'apprentissage pour la phase d'apprentissage n'est pas bien défini. Plusieurs groupes ont approché le pronostic comme un problème de séparation en utilisant différentes architectures d'apprentissage telles que les réseaux neuronaux artificiels de rétro-propagation [14], les réseaux de maximisation d'entropie [15], [16], arbres de décision [17] et des mesures basées sur la logique floue [18].

2.4.2. Modèle proposé de diagnostic du cancer du sein

La figure 1.3 représente le schéma fonctionnel du projet de modèle de diagnostic du cancer du sein [6]. Il se compose de deux phases, à savoir : les phases d'apprentissage et de test. La phase d'apprentissage comprend quatre étapes : acquisition, prétraitement, extraction de caractéristiques et sélection de caractéristiques, tandis que la phase de test comprend les mêmes quatre étapes de la phase de formation en plus de l'étape de classification. Dans une étape d'acquisition, les données du capteur sont soumises à un processus d'extraction et de sélection de caractéristiques pour déterminer le vecteur d'entrée du système de classification. Cela rend une décision concernant la classe associée à ce vecteur de modèle. En fonction de la sélection des caractéristiques ou de l'extraction des caractéristiques, la réduction de la dimensionnalité est accomplie. Dans l'étape de prétraitement, l'image est préparée et filtrée pour éliminer le bruit et améliorer sa qualité. D'un autre côté, l'extraction de caractéristiques considère l'ensemble du contenu de l'information et « mappe » le contenu de l'information utile dans un espace de caractéristiques dimensionnel inférieur. La sélection des entités est basée sur l'omission de ces caractéristiques à partir des mesures disponibles qui ne contribuent pas à la séparabilité des classes. Autrement dit, les fonctionnalités redondantes et non pertinentes sont ignorées. Dans l'étape dite de classification, différents traitements sont appliqués pour obtenir le meilleur résultat de diagnostic et de pronostic de la tumeur.

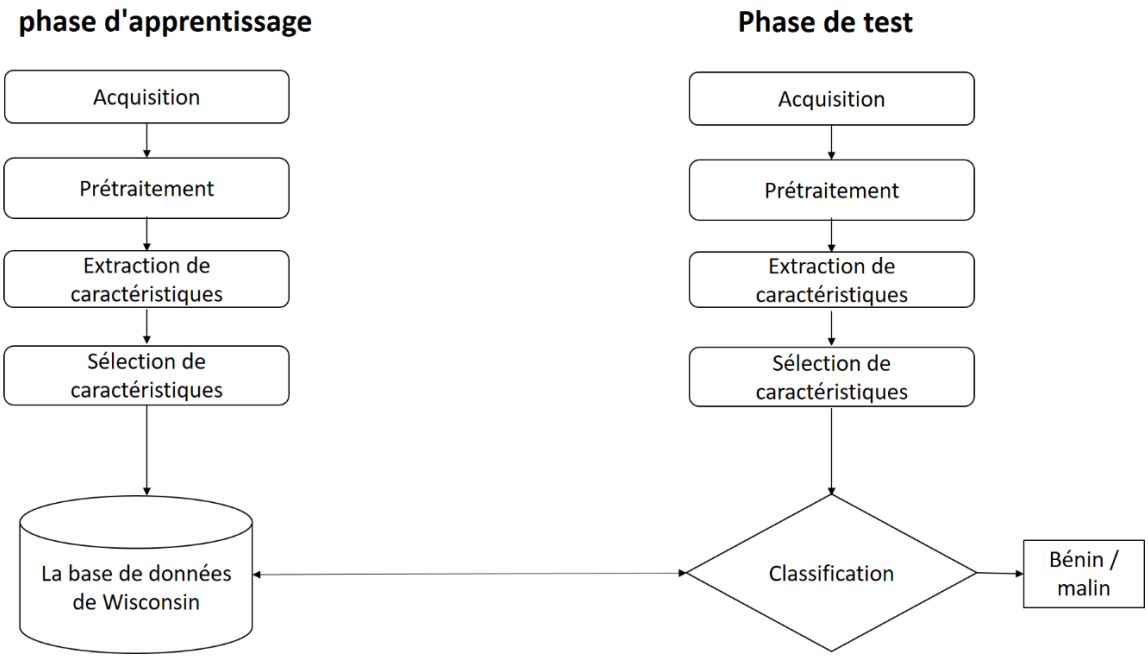


Figure 1.3 : Méthodologie proposée pour la classification du cancer sein

2.4.3. Extraction et sélection de caractéristiques

Une étape importante dans le modèle de diagnostic du cancer du sein est l'extraction de caractéristiques. Le jeu de fonctionnalités optimum devrait avoir des fonctionnalités efficaces et discriminantes, tout en réduisant la redondance de l'espace des fonctionnalités pour éviter les problèmes de "malédiction de dimensionnalité". La "malédiction de la dimensionnalité" suggère que la densité d'échantillonnage des données d'apprentissage est trop faible pour promettre une estimation significative d'une fonction de classification de grande dimension avec le nombre fini de données d'entraînement disponibles.

La classification est l'une des tâches les plus importantes et essentielles dans l'apprentissage automatique et l'exploration de données. Beaucoup de recherches ont été menées pour appliquer l'exploration de données et l'apprentissage automatique sur différents ensembles de données médicales pour classer le cancer de sein. Beaucoup d'entre elles montrent une bonne précision de classification. Ce point sera particulièrement approfondi lors de l'étude et du choix de notre algorithme de classification.

2.5. Conclusion

La base de données Wisconsin sur le cancer du sein (BCDB) permet à un chercheur de se familiariser sur cette maladie. Un glossaire, lié à cette base de données, décrit tous les termes liés au cancer du sein. Il s'agit d'une base de données complète contenant des données sur tous les aspects de la maladie. Elle servira de référence pour concevoir et valider nos algorithmes et notre

électronique de classification. Ces outils informatiques ou électroniques de sauvegarde, d'analyses d'images pour diagnostiquer des pathologies sont aujourd'hui incontournables dans le monde de la santé. Le système de santé intelligent peut aider les médecins à diagnostiquer les patients avec une plus grande précision ou à fournir des repères plus significatifs. De plus, il pourra aider les gens à planifier leur condition physique dans le futur.

Dans le paragraphe suivant, une explication détaillée de la méthode de classification appliquée au cancer du sein sera présentée. Nous nous concentrerons sur le réseau neuronal artificiel (RNA), de type perceptron multicouche avec rétro-propagation, qui est la méthode la plus utilisée dans le diagnostic du cancer du sein et le pronostic.

3. Etat de l'art sur l'implémentation matérielle des réseaux de neurones

3.1. Introduction

Une revue sur l'état de l'art des implémentations matérielles microélectroniques des réseaux neuronaux est présentée dans ce chapitre. La vue générale des réseaux de neurones microélectroniques est très variée. En fait, il existe de nombreuses implémentations rapportées dans la littérature pour les environnements ambiants, industriels ou académiques. Entre autres, M. Lawson dresse un panorama des implémentations « neurone-ordinateur » au Japon [19]. Dans [20] est présenté l'état des recherches sur les réseaux neuronaux matériels. Y. Hiray fait un état de l'art de l'implémentation matérielle des réseaux de neurones dans le monde [21]. Il analyse ainsi le type d'implémentation (analogique, numérique, optique, etc.).

Les implémentations microélectroniques (entièrement personnalisées) sont multiples et il est difficile de les comparer en raison des différents modèles et approches utilisés. Une première distinction entre les implémentations numériques et analogiques peut être caractérisée comme suit: pour le premier type, l'information est codée de manière numérique [0,1] et le calcul est réalisé par des circuits numériques; pour le second type, l'information demeure analogique (tension ou courant) et le calcul est réalisé à travers des circuits analogiques ou mixtes (les circuits numériques servant par exemple pour mémoriser les poids).

Les applications de l'implémentation numérique sont généralement utilisées pour un accélérateur neuronal matériel dans des systèmes pour «applications générales» et sont basées sur l'architecture SIMD (Single Instruction Multiple Data). Parmi toutes, il convient de signaler trois de

celles qui sont les plus significatives. Le CNAPS [22] est un processeur SIMD modulaire qui permet de construire des réseaux de neurones hiérarchiques dans lesquels ils peuvent être formés en utilisant différents algorithmes d'apprentissage. L'IBM ZISC 036 [23] permet d'implémenter le réseau de fonction de base radiale de toutes les dimensions. Enfin, le Philips L-Neuro 2.3 [24] est un processeur neuronal qui permet de mettre en œuvre de nombreux réseaux neuronaux différents. Une caractéristique importante de L-Neuro 2.3 est la possibilité de réaliser la procédure d'apprentissage à travers la puce elle-même (apprentissage sur puce).

Les principales caractéristiques de la puce de réseau neuronal numérique sont :

1. L'apprentissage est hors puce.
2. Le calcul anticipé est sur puce.
3. La précision du poids est limitée par le nombre de bits, en général 8 à 16 bits.
4. L'absence de traitement parallèle (notez que les réseaux de neurones biologiques ont un traitement parallèle).
5. La vitesse du réseau est obtenue par un compromis entre la dimension de l'architecture du réseau neuronal et la précision.
6. Aucune optimisation de la consommation d'énergie.

Parce que l'argument de la présente thèse est l'implémentation analogique microélectronique du réseau de neurones, je ne m'attarderai pas davantage sur les implémentations numériques. De toute façon, il est utile de souligner que les implémentations analogiques ne sont pas mises en compétition avec les implémentations numériques mais leur sont complémentaires. En fait, les implémentations analogiques ne sont généralement pas destinées aux implémentations de d'accélérateur matériel générique, mais plutôt aux implémentations des systèmes électroniques neuronaux pour des applications spécifiques.

Les principales caractéristiques de la puce de réseau neuronal analogique sont:

1. Réduction des dimensions et de la consommation d'énergie et haute miniaturisation du système final.
2. Haut parallélisme du traitement des données ; en fait puisque le neurone et la synapse peuvent être réalisés avec une faible surface de silicium, toute l'architecture neuronale peut être réalisée sur une seule puce.
3. Grande vitesse de calcul compatible avec la précision requise.
4. Tolérance d'erreur élevée, due au fait puisqu'ils peuvent être réalisés avec des dimensions élevées pour un traitement de signal parallèle où l'information est distribuée sur toutes les synapses et les neurones du réseau.

De plus, les puces de réseau de neurones analogiques n'ont pas besoin de convertisseurs A/N et N/A complexes et coûteux… Toutefois, les implémentations analogiques présentent quelques inconvénients:

1. Précision réduite de chaque primitive qui doit être utilisée pour des applications où la précision n'est pas un paramètre critique.
2. Immunité au bruit peu élevée; ce facteur peut être corrigé si l'apprentissage sur puce le prend en compte.
3. Temps de conception et réalisation du projet et du test de la puce élevé.
4. Difficulté et prix élevé pour réaliser une mémoire analogique (certains utilisent la technologie EEPROM).

Quoi qu'il en soit, les implémentations analogiques ont déjà trouvé des applications significatives. La puce EPSILON [25] a été appliquée à la reconnaissance vocale. La puce Kakadu [26] a été appliquée avec succès pour la reconnaissance morphologique. Enfin, le "jeu de puces" développé par Mead [27] a été appliqué pour la classification des chèques manuscrits.

Dans les systèmes de réseau de neurones analogiques microélectroniques, les informations sont codées soit en utilisant un réseau de neurones à temps continu de courant ou de tension, ou en utilisant des impulsions utilisant une modulation d'amplitude ou de fréquence.

3.2. Réseaux neuronaux de flux d'impulsions

Les réseaux de neurones à impulsions utilisent des signaux numériques pour transmettre des informations qui sont codées de manière analogique par modulation d'amplitude d'impulsions (PAM), modulation de fréquence d'impulsion (PFM), modulation de largeur d'impulsion (PWM) ou modulation de phase (ou position) d'impulsion (PPM). Par exemple, PWM est relativement simple, la valeur instantanée de l'état de la grandeur analogique est représentée par la largeur des impulsions numériques individuelles. Le signal peut être décodé en une valeur analogique par intégration.

Une revue des réseaux neuronaux de flux d'impulsions est présentée dans [28]. Une description complète de cette approche et une présentation détaillée de la puce EPSILON sont présentées dans [25]. La puce EPSILON (Edinburg Mise en œuvre d'un réseau de flux d'impulsions orienté vers l'apprentissage) est réalisée en technologie CMOS 1.5 µm, avec une surface de silicium de 9,5 x 10,1 mm^2 et une consommation d'énergie est de 350 mW. La puce est composée de 3600 synapses et la vitesse est de 360MCPS (Mega Connections Per Second), les poids synaptiques sont mémorisés dynamiquement dans des condensateurs. La puce EPSILON a été utilisée pour implémenter un MLP pour la classification des signaux vocaux.

L'avantage de cette approche est l'immunité élevée au bruit, les communications entre les modules sont simples et certaines opérations (par exemple multiplication, addition) peuvent être exécutées par de simples circuits numériques. Les inconvénients sont d'une part la grande complexité du système, et d'autre part la faible vitesse, due à des calculs parallèles réduits.

3.3. Réseau de neurones d'inspiration biologique

Cette terminologie provient du fait que l'architecture de ce type de réseaux de neurones s'inspire directement des systèmes biologiques. L'information est traduite par un signal optique, acoustique ou électrique (courant ou tension) en dimension matricielle, d'où il est possible d'obtenir des informations sur le flux optique ou la position spatiale des sources acoustiques. Mead a défini ce type de réseaux comme des systèmes électroniques neuromorphiques [29]. Les caractéristiques principales de ces réseaux sont :

1. Absence d'apprentissage de la puce, les poids sont fixes.
2.. Interfaçage avec des capteurs optiques (de type CCD, capteurs matriciels CMOS, etc.).
3. Utilisation de transistors MOSFET travaillant en faible inversion.
4. Réponse de temps réduite (par exemple, dixième de seconde).

Leurs principales applications sont dans la mise en œuvre de fonctions sensorielles artificielles, comme par exemple la rétine silicium, la cochlée électronique, le flux optique, … [30], [31], [32], [33], [34], [35], [36], [37].

3.4. Exemple de l'Intel 80170NX (ETANN)

L'ETANN est une puce commerciale produite par Intel Corporation [38]. La puce contient 64 neurones et deux matrices synaptiques dont les dimensions respectives sont de 80 x 64. Les entrées et les sorties sont des signaux analogiques.

L'architecture qui peut être réalisée par une puce est un perceptron monocouche ou un réseau Hopfield. Les poids sont mémorisés à l'intérieur de la puce grâce à l'utilisation de la technologie de mémoire à grille flottante (EEPROM). Un ordinateur neuronal basé sur la puce ETANN est présenté dans [39]. En dépit de sa renommée, l'ETANN présente quelques inconvénients, qui ont empêché la diffusion de ses applications en environnement industriel :

1. L'architecture de la puce est rigide et il est difficile d'obtenir une configuration MLP.
2. La précision du poids est limitée à 8 bits.
3. L'abordabilité du stockage du poids est faible, pour cela il n'est pas possible de l'utiliser pour des configurations d'apprentissage de puce dans la boucle.
4. La vitesse n'est pas très élevée (typiquement 3 ms pour une couche).

3.5. Réseau de neurones cellulaires analogiques

Les réseaux de neurones cellulaires analogiques CNN ont de nombreuses applications dans le domaine du traitement d'images, du son, de la filtration de bruit, ... Du point de vue de la mise en œuvre analogique, les CNN sont particulièrement intéressants :

1. Le traitement est local pour que le routage des signaux soit moins complexe.
2. Poids fixes (il n'existe pas de processus d'apprentissage).

Les applications liées à ces réseaux sont limitées en raison de l'absence d'apprentissage et de la rigidité de l'architecture.

M. Antiga et al. présentent une puce avec 2 CNN, dans laquelle les synapses sont basées sur des multiplieurs CMOS (signal en courant) [40]. Dans [8] est présentée une implémentation VLSI d'un CNN employant pour chaque multiplieur (n-1) OTA et 4 miroirs de courant où n est le nombre de bits, signe inclus. On peut également citer [41] où est présentée une approche pour le projet d'une puce qui implémente un CNN dans la technologie CMOS et où les signaux sont là encore représentés par des courants.

3.6. Réseau à temps continu sans apprentissage sur puce

3.6.1. Introduction

Dans ce type de réseau, l'information est directement codée par un signal électrique (courant ou tension). Les signaux varient en temps continu. En outre, l'apprentissage n'est pas réalisé sur puce. Il existe de nombreuses implémentations dans la littérature. Dans cette brève revue, seuls les réseaux innovants en termes d'architecture et d'implémentation circuits sont rappelés.

3.6.2. Réseaux reconfigurables

Le but de ces réseaux est de rendre le réseau programmable, de manière à configurer le réseau suivant l'application cible.

Une première approche pour réaliser un réseau programmable est basée sur la mise en œuvre de différentes puces pour les neurones, les synapses et les commutateurs. Une telle approche est présentée dans [42], où est décrite la mise en œuvre de l'ordinateur neuronal programmable analogique composé de plus de 100 modules VLSI qui fonctionnent en temps réel. Ces modules VLSI, réalisés en technologie CMOS 2µm, peuvent être regroupés en quatre types différents :

1. Module Neurone : il contient 8 neurones. En entrée il reçoit le courant de la synapse et produit en sortie une tension (gamme 0-4V), la bande passante est de 150 kHz.

2. Module Synapse : la puce contient de 8 à 16 synapses et convertit en courant la tension de sortie du neurone.

3. Module de constante de temps : cette constante de temps est programmable de 5ms à 1s avec un codage sur 4 bits.

4. Module de commutation : il est utilisé pour contrôler le routage des signaux entre les neurones et les synapses, pour cela il spécifie la configuration du réseau.

Par exemple, J.A Lanser et T. Lehmann présentent un "jeu de puces" permettant une topologie arbitraire [43]. Le jeu de puces est composé d'une puce qui a des neurones et une autre qui a des synapses. Une puce ayant des neurones, des synapses et des commutateurs est présentée dans [44]. La puce présente une topologie programmable et il en résulte qu'elle est modulaire et permet, en utilisant plus de puces, de réaliser un réseau neuronal à alimentation anticipée (feed forward) de grandes dimensions. Dans [45], une puce de 4096 synapses avec une précision de 6 bits est présentée. Le nombre de synapses pour un neurone est variable et programmable entre 64 et 256 synapses.

3.6.3. Réseau avec mémoire locale

L'un des principaux problèmes dans les réseaux de neurones VLSI analogiques est la mise en œuvre de la mémoire à long terme (LTM : Long Term Memory) sur les puces. La réalisation de circuits analogiques nécessite des capacités de stockage analogiques locales sur puce. Ce problème n'a pas encore été résolu de manière satisfaisante.

Pendant l'apprentissage, le poids est mis à jour à chaque itération. Pour effectuer cette opération, on peut utiliser un condensateur simple à l'intérieur de chaque synapse pour mémoriser le poids local. Toutefois le problème demeure au cours du processus de classification. En effet, le poids doit être fixé à la valeur obtenue par la phase d'apprentissage. Pour éviter la décharge du condensateur et donc conserver la valeur du poids, 3 méthodes dites LTM sont proposées :

1. Stockage analogique sur puce en utilisant une technologie à grille flottante (par exemple EEPROM). Ainsi, dans ETANN [46], une telle approche appliquée sur la mise en œuvre du réseau de neurones analogiques est discutée.
2. Mémoire binaire et locale des poids (stockage numérique sur puce) [47], [48]. Cette technique permet de diminuer le « surdébit » du système connecté à la gestion de rafraîchissement des poids. En fait, le stockage est binaire et statique; mais la précision dépend du nombre de bits et pour chaque synapse il faut un convertisseur N/A.
3. Technique de stockage multi-valeur et "self-refresh" [49], [50]. A l'intérieur de chaque synapse, la valeur du poids est stockée dynamiquement sous forme de tension dans un condensateur. Toutefois, le poids peut prendre une valeur dans l'ensemble des valeurs de tension discrètes permises pendant l'autorafraîchissement.

H.P. Graf et al. présentent une puce implémentant une mémoire associative utilisant 256 neurones avec des connexions fixes (synapses) [51]. Dans [52], les entrées et les sorties sont numériques (1 bit), la puce (appelée NET32K et implémente une mémoire associative) contient 256 neurones avec un nombre variable de connexions (synapses) entre 128 et 1024. Le maximum de précision des synapses est de 4 bits et la vitesse est de 3 GCPS (Giga Connections Per Second).

3.7. Réseau à temps continu et apprentissage sur puce

3.7.1. Introduction

Les implémentations analogiques VLSI de réseaux de neurones avec apprentissage sur puce sont divisées en deux catégories (dans cette brève revue, le réseau Kohonen et le réseau de machines Boltzmann ne seront pas considérés):

1. Les puces qui implémentent l'algorithme de propagation inverse (BP), minimisant la fonction de coût en utilisant la règle de descente de gradient. Un tel algorithme est très robuste et fiable ; il sera expliqué en détail dans cette thèse. Toutefois, il nécessite plus de précisions. Néanmoins, certaines versions de l'algorithme BP ont des caractéristiques qui le rendent adapté aux implémentations matérielles analogiques. En outre, le BP est très rapide pour obtenir la convergence puisque tous les poids sont mis à jour continuellement.

2. Les puces qui implémentent l'algorithme de perturbation de poids (ou algorithme de perturbation de nœud), minimisent la fonction de coût en utilisant une approximation de la règle de descente de gradient [26]. Un tel algorithme, bien que très coûteux en termes de calcul, est correctement utilisé pour les implémentations VLSI. Les inconvénients de cet algorithme sont :

- Il n'est pas vérifié et appliqué à de vrais problèmes.
- les poids ne sont pas mis à jour continuellement et la vitesse est très faible.

3.7.2. Processus d'apprentissage utilisant la règle de descente de gradient ou rétro-propagation

Une analyse critique de l'implémentation analogique de l'algorithme de rétro-propagation (BP) est détaillée dans [52]. B. Furman et al. décrivent un circuit qui réalise l'algorithme BP [53]. La puce contient 10 neurones, 480 synapses et 48 entrées. Tous les signaux internes sont analogiques. Les poids sont mémorisés dynamiquement à l'intérieur de la puce par des condensateurs.

Dans [54], l'architecture proposée réalise l'algorithme de rétro-propagation. Tous les calculs sont réalisés à travers des circuits analogiques. Mais il faut un convertisseur numérique-analogique car les poids sont mémorisés sous forme numérique sur puce. T. Shima et al. intègrent leur

application sur deux puces distinctes [55]. L'une contient une matrice synaptique de (24 x24) avec une mémoire de poids locale, tandis que l'autre implémente 24 neurones. Le calcul est analogique tandis que les poids sont mémorisés dans une mémoire numérique. L'architecture implémente l'algorithme de rétro-propagation, ainsi que celui de Hebbian. L'architecture du processus d'apprentissage peut être étendue à 480 neurones et 230000 synapses. La vitesse du réseau est de 36GCUPS (Giga Connections Updated Per Second). Dans [56], une puce implémentant la rétro-propagation intègre les neurones et les synapses qui exécutent toutes les opérations. Le paramètre de taux d'apprentissage est géré hors puce.

3.7.3. Processus d'apprentissage utilisant la règle de descente de gradient approximatif ou algorithme de perturbation de poids

Une revue des algorithmes de perturbation pondérale est rapportée dans [26], ainsi que l'architecture et l'analyse de deux puces réalisant un perceptron multicouche : les deux puces sont appelées Kakadu et Tarana. L'algorithme de perturbation de poids est utilisé pour réaliser la phase d'apprentissage des deux puces.

M. Jabri et B. Flower utilisent un algorithme de perturbation de poids, qui n'a pas besoin du circuit « dérivée » pendant l'apprentissage [57]. Dans [58] un réseau de neurones analogiques avec un algorithme de perturbation de poids sur puce est présenté. La principale caractéristique de la puce est l'utilisation de la mémoire analogique à long terme en utilisant la méthode de la grille flottante.

Le circuit, présenté dans [59] réalise la mise en œuvre VLSI analogique d'un réseau neuronal récurrent. La puce contient 6 neurones analogiques, 36 synapses et 6 synapses de seuil. Elle implémente l'algorithme de descente de gradient stochastique, par des perturbations stochastiques pseudo-aléatoires, de manière à éviter les calculs de gradient.

3.8. Conclusion

Les réseaux neuronaux artificiels ont été largement utilisés dans de nombreux domaines d'application. Une grande variété de problèmes peut être résolue avec les RNA comme la classification, la reconnaissance des formes, le traitement du signal, le contrôle des systèmes etc. La plupart des travaux effectués dans ce domaine concerne la simulation logicielle, l'investigation des capacités des RNA ou la définition de nouveaux algorithmes. Mais les implémentations matérielles sont également essentielles pour leur mise en œuvre en bénéficiant du parallélisme inhérent du réseau neuronal. Nous présentons dans la suite de ce chapitre la méthodologie utilisée pour concevoir notre RNA et l'implémenter sous forme de circuits VLSI analogiques.

4. Méthodologie de conception

4.1. Introduction

En se basant sur la problématique de la classification des tumeurs cancéreuses dans le sein par des réseaux de neurones artificiels présentée précédemment et une fois l'algorithme BP choisi, la méthodologie et les techniques de ce projet ont été basés initialement sur les étapes suivantes :

1. Choisir la topologie de circuit la plus adaptée à notre algorithme.
2. Dimensionner chaque bloc à partir de simulations au niveau système (Matlab/Simulink) afin d'optimiser les spécifications en termes de dynamique et précision par exemple. Notons qu'un aller-retour sera nécessaire pour prendre en compte les non-idéalités de chaque bloc ou brique de base tout en assurant la qualité du système de classification.
3. Étudier l'état de l'art des systèmes de réseaux de neurones artificiels, en particulier les fonctions non linéaires nécessaires et leurs dérivées pour le calcul des erreurs.
4. Optimiser les circuits de base (multiplieurs et fonctions non linéaires principalement), ainsi que leurs dérivées et le calcul d'erreurs.
5. Choisir une technologie CMOS VLSI mature et bas coût pour réaliser l'optimisation au niveau transistor : en fait la technologie HCMOS 130 nm a été imposée. Pour des raisons de consommation nous avons opté pour une basse tension d'alimentation (±0,9V) compatible avec nos exigences de dynamique.
6. Utiliser un logiciel spécialisé et dédié pour simuler le circuit proposé, à savoir l'outil Cadence Virtuoso 6.14.
7. Vérifier et valider les principales caractéristiques des circuits proposés.
8. Mettre en œuvre le système global (tel que MLP) et tester sa performance. Choisir ensuite les meilleurs circuits ou briques de base.
9. Simuler la disposition des circuits puis la simuler avec le logiciel Cadence Virtuoso 6.14.
10. Vérifier les caractéristiques et tester les performances du système de classification par des simulations post-layout.
11. Fondre le circuit et le tester, en fonction de l'avancement du projet.

L'exécution et la finition du projet de recherche seront principalement divisées en trois phases chronologiques. La première concerne l'étude et la conception du circuit, elle sera effectuée au sein de Polytech'Lab de l'Université de Nice Sophia Antipolis et à l'Université Internationale Libanaise. Puis l'étape de fabrication de la puce se fera dans le laboratoire Polytech'Lab Nice en France. Enfin, les mesures de la puce seront effectuées dans le laboratoire de microélectronique de l'Université

Internationale Libanaise. Notons que, par manque de temps, cette phase n'a pas été réalisée. Ainsi, les principaux points de ce travail sont :

1. Conception d'une architecture analogique réalisant l'algorithme MLP formé par BP ;
2. Caractérisation des cellules primitives computationnelles (le neurone et la synapse) ;
3. Traduction de l'opération de calcul dans les circuits analogiques microélectroniques ;
4. Conception de circuits analogiques avec une faible surface de silicium, une faible consommation d'énergie et une haute vitesse ;
5. Méthodologie pour la validation comportementale de l'architecture analogique ;
6. Simulations sous Cadence de l'architecture analogique au niveau du transistor ;
7. Projet de puce VLSI analogique.

4.2. Conception d'une architecture analogique

4.2.1. Réseau neuronal MLP

Le réseau neuronal, MLP formé par l'algorithme BP [60], nécessite la mise en œuvre d'une part des calculs d'anticipation et d'autre part des calculs rétrospectifs. Fondamentalement, les calculs d'anticipation ont besoin d'une multiplication, d'une fonction d'activation non linéaire alors que les calculs vers l'arrière ont également besoin d'une multiplication, d'une dérivation, d'un calcul de taux d'apprentissage, d'une mise à jour de poids.

Le réseau neuronal peut également être vu à travers un réseau de synapses et de neurones. La synapse effectue et exécute les opérations suivantes : multiplication en avant, multiplication en arrière, calcul du taux d'apprentissage, mise à jour du poids et mémoire de poids. Le neurone exécute les opérations suivantes : fonction d'activation non linéaire, dérivée et bloc d'erreur.

Une architecture analogique de la MLP formée par l'algorithme de BP est présentée et les structures de la synapse et du neurone sont présentées et conçues dans cette thèse.

4.2.2. Circuits analogiques CMOS VLSI

Les réseaux neuronaux sont implémentés le plus efficacement par des circuits analogiques asynchrones. Les implémentations analogiques sont généralement plus rapides (en raison du fonctionnement asynchrone) et requièrent moins de matériel (nombre de transistors plus faible) que les implémentations VLSI numériques. Les réseaux de neurones VLSI analogiques sont de lourds systèmes analogiques parallèles, utilisés et démontrés pour résoudre un large éventail de problèmes du monde réel.

Dans ce contexte, les fonctions non linéaires et leurs multiplieurs analogiques dérivés sont des éléments de calcul clés et sont très importants pour les mettre en œuvre de manière efficace. Dans le cas d'une MLP Perceptron multi-couche (cf. Figure 1.4), chaque synapse a besoin d'un multiplieur pour multiplier l'entrée par le poids et chaque neurone a besoin d'une fonction non-linéaire et de sa dérivée.

Le système RN (Réseau Neuronal ou NN Neural Network) peut contenir des milliers de multiplieurs et des fonctions non linéaires. De ce fait, la surface et la consommation d'énergie de chaque bloc (ou brique de bas) doivent être les plus faibles possibles.

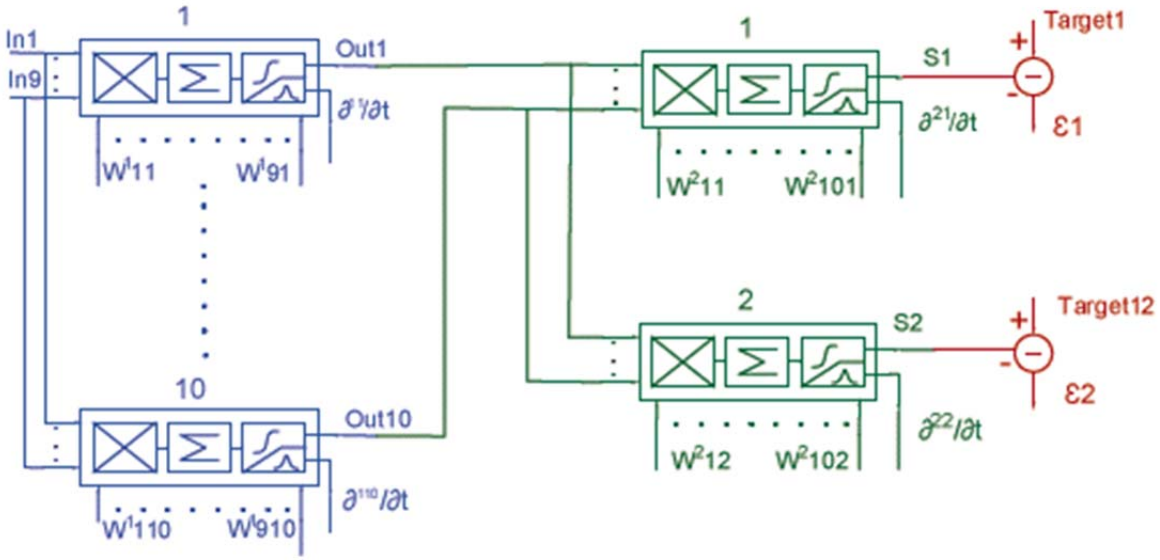


Figure 1.4 : Structure du réseau neurone à propagation vers l'avant

La figure 1.4 illustre une couche, dont les entrées sont notés "In1, In2, ..., In9" et les sorties "Out1, Out2, ..., Out10", d'un système multicouche MLP Perceptron. Si la couche est la première couche, alors le nombre d'entrées correspond à celui du modèle à apprendre, tandis que le nombre de neurones dans chaque couche est égal au nombre de neurones cachés. Dans notre cas, ce nombre d'entrée est de 9, correspondant aux 9 attributs utilisés dans la base Wisconsin. Si la couche est la dernière couche, alors le nombre d'entrées est égal au nombre de neurones cachés, tandis que le nombre de sortie représente le nombre de neurones fournissant les résultats de la classification. Dans notre cas, ce nombre de sortie est de 1 pour fournir soit 0 pour une tumeur bénigne et 1 pour une tumeur maligne.

Chaque synapse nécessite deux multiplieurs (Feed-Forward et Backward). Le multiplieur Feed-Forward va multiplier l'entrée "par exemple, In1" par le poids $W^1$11 et produit une sortie Iout11.

La fonction neuronale non linéaire prendra la somme de tous les multiplieurs de synapses connectés à ce neurone et produira une sortie Out1. La sortie de la fonction non-linéaire des neurones sera l'entrée du bloc dérivé pour calculer la dérivée de la fonction non-linéaire $\partial^{11}/\partial t$. La fonction non linéaire peut être la fonction sigmoïde (ou logsigmoid), la fonction tangente hyperbolique ou l'approximation linéaire par morceaux.

Les considérations de circuit suivantes seront prises en compte (cf. Figure 1.4) :

1. Les deux signaux d'entrée sont des tensions :
    a. La première entrée est le poids, stocké dans un condensateur, ensuite il sera représenté par une tension "$W^1$11, $W^1$21, ..., $W^1$91"
    b. La deuxième entrée sera la tension "In1, In2, ..., In9" ; la sortie du neurone prendra en entrée la somme (loi de Kirchoff ou KCL) de toutes les synapses de sortie connectées à ce neurone, et produira une tension de sortie
2. La sortie correspond aux courants "I111, I121, ..., I191", utilisés par la loi KCL afin d'obtenir la somme des sorties des synapses connectées à chaque neurone en entrée de ce neurone
3. La tension de sortie non linéaire des neurones "Out1, Out2, ..., Out10" sera représentée par une tension et l'entrée par un courant
4. Le bloc de dérivation sera représenté par une tension, ainsi que son entrée
5. Tous les circuits doivent avoir une surface de silicium la plus faible possible et une faible consommation d'énergie
6. La conception et l'optimisation de tous les circuits en utilisant la technologie HCMOS9A 130nm.

4.3. Multiplieurs VLSI analogiques d'un réseau neuronal

Le multiplieur analogique est un élément clé essentiel dans le traitement du signal analogique et en particulier dans la mise en œuvre VLSI analogique des réseaux neuronaux artificiels. Les principales conditions de ces types de multiplieurs sont les suivantes :

1. Ils doivent avoir une faible surface zone de silicium.
2. Ils doivent avoir une faible consommation d'énergie.
3. L'effet de la mobilité des électrons du transistor doit être minimisé ou annulé.
4. Trouvez une méthode pour simplifier la disposition du placement (layout).
5. Les variations d'entrées doivent être aussi grandes que possible.
6. Ils doivent avoir une bonne linéarité.

Les objectifs de cette recherche sont d'étudier, de concevoir et de réaliser des multiplieurs analogiques à quatre quadrants et les circuits de fonction non linéaire et leurs dérivées dans les modules de neurones.

Notre but sera également de trouver le meilleur et le plus approprié multiplieur à quatre quadrants, la fonction non linéaire et sa dérivée et le calcul d'erreur respectant notre cahier des charges. Après cela, nous commençons à concevoir la puce finale du réseau neuronal en utilisant les différents blocs de base.

4.4. Structure du module neuronal

4.4.1. Hypothèses et contraintes

Ce paragraphe décrit la structure du module neuronal composé d'une part du module du neurone dans la couche de sortie et d'autre part du module du neurone dans la couche cachée. Généralement, le neurone (dans la sortie de la couche cachée) possède trois blocs principaux : le bloc fonctionnel d'activation, le bloc dérivé et le bloc d'erreur.

Notre conception devra résoudre les contraintes suivantes :

1. Concevoir et optimiser les circuits CMOS VLSI qui implémentent un **multiplieur** pour le module de neurones.
2. Concevoir et optimiser les circuits CMOS VLSI qui implémentent une ***fonction non-linéaire*** pour le module de neurones.
3. Concevoir et optimiser les circuits CMOS VLSI qui implémentent une **fonction dérivée linéaire** pour le module neuronal.
4. Concevoir et optimiser les circuits CMOS VLSI qui implémentent et calculent l'erreur (sortie - cible) qui sera transmise aux synapses.

Ces circuits seront testés en mode courant ou tension, puis le meilleur compromis sera choisi. Ils seront simulés en utilisant une technologie CMOS 130 nm (HCMOS9 de la société ST Microelectronics) avec une alimentation égale à ± 0,9V. Les avantages et les inconvénients de toute option seront étudiés en détail. Ainsi, nous avons étudié et conçu les circuits synaptiques suivants :

- Le multiplieur à quatre quadrants utilisé pour la multiplication directe (poids *entrée) ; il est également utilisé pour la retro multiplication (poids *erreur);
- Le taux d'apprentissage détermine la quantité de mise à jour du poids. Il augmente si le signe de mise à jour du poids en deux itérations successives ne change pas, sinon il diminue. Toutefois, dans notre cas, nous utilisons un taux d'apprentissage constant = 0,25.
- Le circuit pour la mémorisation des poids.

Concernant les neurones, nous avons étudié et conçu deux circuits, d'une part la fonction d'activation non-linéaire réalisée par une fonction Logsigmoid et d'autre part le circuit dérivé réalisé par une paire différentielle.

4.4.2. Validation comportementale de l'architecture analogique

Une modélisation/simulation comportementale de l'architecture analogique a été développée afin d'une part de valider cette architecture et d'autre part de spécifier chacun des blocs. Ce modèle sera enrichi en prenant en compte les non-idéalités de ces différentes briques de base afin d'évaluer leurs influences sur la qualité du système de classification et éventuellement modifier les spécifications de certains blocs [61]. On a déduit les modèles fonctionnels de tous les blocs de l'architecture analogique. Ces modèles contiennent les effets non-linéaires et non-idéaux dus à la mise en œuvre du silicium analogique. Un programme écrit en Matlab est développé pour simuler les modèles fonctionnels. Il implémente le MLP formé par BP.

Cette approche a été validée en utilisant la base de données Wisconsin, contenant 699 cas. La topologie du Multi-Layer Perceptron à deux couches (MLP) est la suivante: 9 entrées, 10 neurones cachés et 2 neurones de sortie (2 classes). Les résultats ont été satisfaisants, puisque les modèles fonctionnels ont peu d'influence sur le processus d'apprentissage ; en fait, le pourcentage d'erreur de l'exemple correctement classé est similaire à celui obtenu en utilisant les modèles idéaux.

4.4.3. Simulation sous Cadence au niveau transistors

Chaque bloc a été simulé et optimisé, au niveau transistor, dans l'environnement Cadence, ainsi que l'architecture analogique du réseau de neurones complet. Pour les simulations avec mise à jour continue du temps et avec le taux d'apprentissage global, l'exactitude du processus d'apprentissage a été vérifiée. Le réseau a appris que les deux classes de détection du cancer du sein étaient malignes ou bénignes. En conséquence, l'architecture analogique et les circuits CMOS VLSI proposés sont satisfaisants. À ce niveau, l'analyse est limitée à un réseau simple en raison du temps de calcul élevé requis par ce type de simulations.

4.5. Conclusion

Le perceptron multicouche est le type de réseau de neurones le plus populaire. En utilisant l'algorithme de retro-propagation (MLP), il peut s'appliquer à une grande variété d'applications, spécialement dans notre cas "la détection et la classification du cancer du sein". Dans ce paragraphe, les blocs principaux pour la conception d'un tel réseau ont été présentés, à savoir le multiplieur, la fonction d'activation et sa dérivée. Nous nous proposons de les concevoir en utilisant le logiciel "Cadence virtuoso" à travers la technologie HCMOS9A 130 nm. De nouvelles topologies seront étudiées et optimisées afin de satisfaire notre cahier des charges, qui sera défini au chapitre III.

5. Conclusion

Le cancer du sein est l'un des cancers les plus communs et létales avec celui du poumon et des bronches, de la prostate et du côlon, … [2]. L'utilisation de l'intelligence artificielle, des réseaux de neurones et des approches d'apprentissage automatique dans le domaine médical s'avère efficiente, car de telles approches peuvent être considérées comme une grande aide dans le processus de prise de décision des médecins.

Les algorithmes d'apprentissage machine ont été « formés » pour détecter le cancer du sein à l'aide de l'ensemble de données du Wisconsin Diagnostic Breast Cancer (WDBC) [6]. La classe la plus populaire d'ANNs multicouches à alimentation directe est le perceptron multicouche, avec une ou plusieurs couches entre les couches d'entrée et de sortie. Sa forme la plus connue utilise l'algorithme de rétro-propagation. Des couches multiples de neurones avec des fonctions de transfert non linéaires permettent au réseau d'apprendre des relations non linéaires et linéaires entre les vecteurs d'entrée et de sortie.

Le réseau neuronal de perceptron multicouche (MLPNN) est un algorithme qui a été continuellement développé pendant de nombreuses années. Par conséquent, lorsque la mise en œuvre VLSI d'un algorithme d'apprentissage est nécessaire, MLPNN est un choix commun. Dans le chapitre suivant, nous rappelons rapidement les bases de la construction des réseaux de neurones artificiels, leur comparaison avec le cerveau humain. Puis nous présenterons plus en détail les réseaux avec propagation avant et rétro-propagation que nous avons retenus pour notre implémentation matérielle sous forme d'un circuit VLSI analogique.

6. Bibliographie

[1] "https://medlineplus.gov/magazine/issues/summer14/articles/summer14pg20.html".

[2] "https://www.cancer.org/cancer/breast-cancer/about/what-is-breast-cancer.html".

[3] Sakorafas et al, Sakorafas G., Krespis E. and Pavlakis G., "Risk estimation for breast cancer development; a clinical perspective," *Surgical Oncology Journal,* vol. 10, pp. 183-192, 2002.

[4] Bellaachia Erhan A and Guven E., "Predicting Breast Cancer Survivability Using Data Mining Techniques," *http://www.siam.org/meetings/sdm06/workproceed/Scientific%20Datasets/bellaachia.pdf,* 2005.

[5] A. Frank and A. Asuncion, "UCI Machine Learning Repository," *http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.,* 2010.

[6] Gouda I.Salama, M.B.Abdelhalim and Magdy Abd-elghany Zeid, "Breast Cancer Diagnosis on Three Different Datasets Using Multi-Classifiers," *International Journal of Computer and Information Technology,* vol. 01, no. 01, pp. 2277-0764, September 2012.

[7] Wang T C. and Karayiannis N B., "Detection of microcalcifications in digital mammograms using wavelets," *IEEE Transactions on Medical Imaging, Vol. 17, Issue. 4, Aug. 1998, pp. 498 – 509.,* vol. 17, no. 4, pp. 498-509, Aug.1998.

[8] Cheng Heng-DA, Lui Yui Man and Freimanis R I., "IEEE Transactions on Medical Imaging," *IEEE,* vol. 17, no. 3, pp. 442-450, June 1998.

[9] Pendharkar P C., Rodger J A., Yaverbaum G J., Herman N. and Benner M., "Association, statistical, mathematical and neural approaches for mining breast cancer patterns," *Expert Systems with Applications,* vol. 17, pp. 223-232, 1999.

[10] Setiono R., "Generating concise and accurate classification rules for breast cancer diagnosis," *Artificial Intelligence in Medicine,* vol. 18, pp. 205-219, 2000.

[11] Chen D., Chang R F. and Huang Y L., "Breast cancer diagnosis using self-organizing map for sonography, Ultrasound," *Medical Biology,* vol. 26, pp. 405-411, 2000.

[12] Giger M., Huo Z., Kupinski M. and Vyborny C., "Computer-aided diagnosis in mammography.," *Handbook of Medical Imaging, (Eds.) Sonka, M., Fitzpatrick, J., Medical Image Processing and Analysis,* vol. 2, pp. 917-986, 2000..

[13] Tourassi G D., M. M. K., Lo J Y. and Floyd Jr C E., "A neural network approach to breast cancer diagnosis as a constraint satisfaction problem," *Med. Phys.,* vol. 28, pp. 804-811, 2001.

[14] Burke H B. and Goodman P H. et al, "Artificial neural networks improve the accuracy of cancer survival prediction," *Cancer,* vol. 29, pp. 857-862, 1997.

[15] Choong P L. and deSilva C.J.S et al, "Entropy maximization networks, An application to breast cancer prognosis," *IEEE Transactions on Neural Networks,* vol. 7, no. 3, pp. 568-577, 1996.

[16] Choong P L. and deSilva C J S., "Maximum entropy estimation vs. multivariate logistic regression: which should be used for the analysis of small binary outcome data sets?," *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* vol. 3, pp. 1602-1605, 1998.

[17] Wolberg W H., Street W N., Heisey D M. and Mangasarian O L., "Computer-derived nuclear features distinguish malignant from benign breast cytology," *Human Pathology,* vol. 26, pp. 792-796, 1995.

[18] Seker H., Odetayo M., Petrovic D., Naguib R N J., Bartol C., Alasio L., Lakshmi M S. and Sherbet G V., "A fuzzy measurement-based assessment of breast cancer prognostic markers," *Proceedings of the 2000 IEEE EMBS International Conference on Information Technology Applications in Biomedicine,* pp. 174-178, 9-10 Nov. 2000.

[19] M. Lawson, "A preliminary view of Japan's high performance neurocomputing," *Neurocomputing,* vol. 4, 1992.

[20] D. Hammerstrom, "Working with neural networks," in *IEEE Spectrum, Luglio*, 1993.

[21] Y. Hirai, "Recent VLSI neural networks in Japan," *Journal of VLSI Signal Processing,* vol. 6 (No.1), pp. 7-18, 1993.

[22] “Data sheet of CNAPS (Co-processing Node Architecture for Parallel Systems). Product by Adaptive Solutions, Inc. ».

[23] “Data sheet of ZISC 036 (Zero Instruction Set Computer). Product by IBM. ».

[24] Yannick DEVILLE. , “Digital VLSI neural networks: from versatile neural processors to application-specific chips.,” in *Proceedings of ICANN’95, Industrial Conference.*, 1995..

[25] A. Murray and L. Tarassenko, “Analogue neural VLSI: a pulse stream approach. Chapman & Hall, » 1994.

[26] M. A. Jabri, R. J. Coggings and B. G. Flower, “Adaptive analogue VLSI neural systems. Chapman & Hall, » 1996.

[27] C. A. Mead and M. A. Mahowald, “A silicon model of early visual processing,” *Neural Networks,* vol. 1, pp. 91-97, 1988.

[28] A. F. Murray et al., “Pulse-stream VLSI neural networks mixing analog and digital techniques,” *IEEE Trans. on Neural Networks,* vol. 2 (No.2), pp. 193-204, March 1991.

[29] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE,* vol. 78 (No10), 1990.

[30] “IEEE SPECTRUM, toward an artificial eye,” May 1996.

[31] W. Bair et al., “An analog VLSI chip for finding edges from zero-crossing,” *Advances Neural Information Processing Systems 3, R.P. Lippmann, J.E. Moody and D.S. Touretzky Ed,* pp. 399 – 405, 1991.

[32] T. M. Bernard, B. Y. Zavidovique and F. G. Devos, “A programmable artificial retina,” *IEEE J. of Solid State Circuits,* vol. 28, pp. 789 – 787, 1993.

[33] T. Delbruck, “Silicon retina with correlation based velocity-tuned pixels,” *IEEE Trans. on Neural networks,* vol. 4, pp. 529 – 541, 1993.

[34] T. Horiuchi et al., “A delay line motion detection chip,” *Advances in Neural Information Processing Systems 3, R.P. Lippmann, J.E. Moody and D.S. Touretzky Ed.,* pp. 406 – 412, 1991.

[35] H. Kobayashi, J. L. White and A. A. Abidi, “An active resistor network for gaussian filtering of images,” *IEEE J. of Solid-State Circuits,* Vols. 26,No. 5, pp. 738-748, May 1991.

[36] C. Kock et al., “Real-time computer vision and robotic using analog VLSI circuits,” *D.S. Touretzky, Ed.Neural Information Processing Systems, Morgan-Kaufmann Publishers,* vol. 2, pp. 750-757, 1990.

[37] L. Watts et al., “Improved implementation of the silicon cochlea,” *IEEE J. of Solid State Circuits,* vol. 27 (No.5), 1992.

[38] M. Holler et al., “An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses.,” in *International Joint Conference on Neural Networks,*, Washington June 18-22,, 1989..

[39] M. L. Mumford et al., “The mod2 neurocomputer system design,” *IEEE Trans on Neural Networks,* vol. 3 (No.3), May 1992.

[40] M. Anguita et al., « Analog CMOS CNN with programmable cloning templates mode cellular neural networks implementation, » *Transactions on Circuit and System-II,* vol. 40 (No.3), 1993.

[41] A. R. Vazquez et al., “Smart-pixel cellular neural network in analog current-mode CMOS technology,” *IEEE Journal of Solide-State Circuits,* vol. 29 (No.8), August 1994.

[42] J.Van der Spiegel et al.,, “An analog neural computer with modular architecture for real-time dynamic computations,” *IEEE Journal of Solid State Circuits,* vol. 27(No. 1), pp. 82-92, J, 1992.

[43] J.A. Lanser and T. Lehmann. , “An analog CMOS chip set for neural topologies.,” in *IEEE Trans. on Neural Networks, vol. 4(No. 3):pp. 441- 444,* , 1993..

[44] S. Satyanarayana, Y. P. Tsividis and H. P. Graf., “A reconfigurable VLSI neural network,” *IEEE Journal of Solid State Circuits,* vol. 27 (No.1), pp. 67-81, J, 1992.

[45] B. E. Boser et al., “An analog neural network processor with programmable network topology,” *IEEE J. of Solid State Circuits,* vol. 26, pp. 2017 – 2025, December 1991.

[46] M. Holler et al., “An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses,” in *International Joint Conference on Neural Networks*, Washington , June 18-22, 1989.

[47] R. J. Coggins et al., “Hybrid analog and digital neural network or intracardiac morphology classification,” *IEEE Journal of Solid State Circuits,* vol. 30 (N0.5), pp. 542 – 550, 1995.

[48] H. P. Graf and D. Henderson, “A reconfigurable CMOS neural network,” *Proc. Of IEEE ISSCC’90,* pp. 144 – 145, 1990.

[49] R. Castello et al., “Self-refreshing analogue memory cell or variable synaptic weights,” *Electronics Letters,* vol. 27, pp. 1871 – 1873, 1991.

[50] E. A. Vittoz et al., “Analog storage of adjustable synaptic weights,” *VLSI Design of Neural Networks, Kluwer Academic Press,* pp. 47-63, 1991.

[51] H. P. Graf et al., “VLSI implementation of a neural network with several hundreds of neurons,” in *Neural Networks for Computing, AIP*, 1986.

[52] A. J. Annema, “ Feed Forward Neural Networks,” in *Kluwer Academic Publishers*, 1995.

[53] B. Furman, J. White and A. A. Abidi, “CMOS analog IC implementing the back-propagation algorithm,” *Neural Networks,* Vols. 1, suppl. I, 1988.

[54] J. J. Paulos and P. W. Hollis., “A VLSI architecture for feed-forward networks with integral back-propagation,” *Neural Networks,* 1, Suppl. I, 1988.

[55] T. Shima et al., “Neuro chips with on-chip backprop and/or hebbian learning,” *IEEE Journal of Solid-State Circuits,* vol. 27(No. 12), pp. 1868-1876, Dec,1992.

[56] M. Valle, D. D. Caviglia and G. M. Bisio, “An experimental analog VLSI neural network with on-chip back-propagation learning,” *Analog Integrated Circuits & Signals Processing, Kluwer Academic Publishers,Boston,* vol. 9, pp. 231-245, 1996.

[57] M. Jabri and B. Flower, “Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks,” *Neural Computations,* vol. 3, pp. 546-565, 1992.

[58] Y. Berg, R. L. Sigvartsen, T. S. Lande and A. Abusland, “An analog feed-forward neural network with on-chip learning,” *Analog Integrated Circuits and Signal Processing,* vol. 9, pp. 65-75, J, 1996.

[59] G. Cauwenberghs, “A learning analog neural network chip with continuous-time recurrent dynamics,” *Neural Information Processing System 6 (NIPS6),* pp. 858-865, 1994.

[60] M. Valle, D. D. Caviglia and G. M. Bisio, “An experimental analog VLSI neural network with on-chip back-propagation learning,” *Analog Integrated Circuits & Signals Processing, Kluwer Academic Publishers,Boston,* vol. 9, pp. 231-245, 1996.

[61] H. Jouni, M. Issa, A. Harb, G. Jacquemod & Y. Leduc, «Neural Network Architecture for Breast Cancer Detection and Classification», IMCET, Beirut, Lebanon, 2016

Chapitre II – Analyse du réseau neuronal

1. Cerveau humain : réalités et fonctions

1.1. Introduction

Le cerveau humain est le centre de commande du système nerveux. Il reçoit en entrée les signaux issus des organes sensoriels et envoie la sortie aux muscles. Le cerveau humain a la même structure de base que les autres cerveaux de mammifères, mais il est plus grand par rapport à la taille du corps que tout autre cerveau [1].

Le cerveau est l'un des organes les plus complexes et les plus magnifiques du corps humain. Notre cerveau nous donne conscience de nous-mêmes et de notre environnement, traitant un flux constant de données sensorielles. Il contrôle nos mouvements musculaires, les sécrétions de nos glandes, et même notre respiration et la température interne. Chaque pensée créative, sensation et émotion est développée par notre cerveau (cf. Figure 2.1). Les neurones du cerveau enregistrent la mémoire de chaque événement dans nos vies [1].

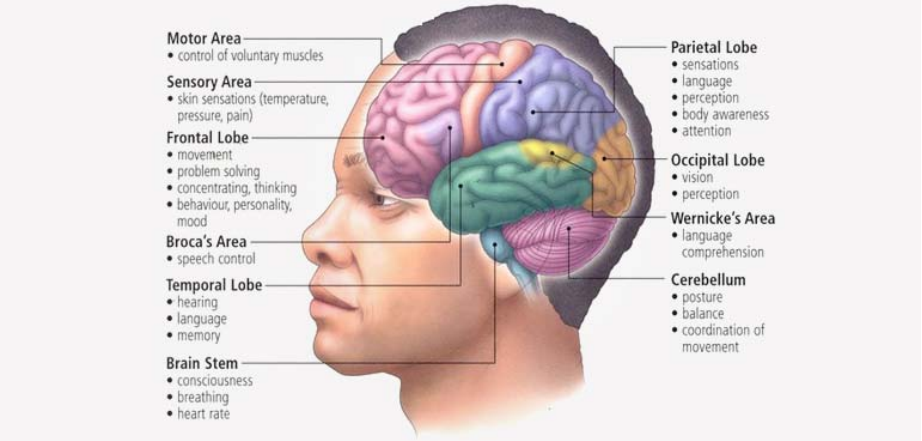


Figure 2.1 : Structure du cerveau humain et leurs fonctions dans le corps humain [2]

1.2. Réalités sur le cerveau humain [1]

Le cerveau humain est le plus grand cerveau de tous les vertébrés par rapport à la taille du corps et pèse environ 1,5 kg. Il représente environ 2% du poids corporel d'un humain. Le cerveau dit cerebrum représente 85% pour cent du poids du cerveau total, et contient environ 86 milliards de

cellules nerveuses (neurones), encore appelée "matière grise". Le cerveau contient des trillons de fibres nerveuses (axones et dendrites) ou "matière blanche". Ces neurones sont connectés par des milliards de synapses.

1.3. Fonctions [3]

Le cerveau est divisé en quatre sections, comprenant le cerveau ou cerebrum, le cervelet, le diencéphale et le tronc cérébral. Chacune de ces parties est responsable de certaines actions du travail global du cerveau. Les plus grandes parties sont, à leur tour, divisées en plus petites zones qui traitent de plus petites tâches. Différentes zones partagent souvent la responsabilité de la même tâche.

Le cerveau (cerebrum) est la plus grande partie du cerveau. Il est responsable de la mémoire, de la parole, des sens, de la réponse émotionnelle et plus encore. Il est divisé en plusieurs sections appelées lobes. Ces lobes sont appelés frontal, temporel, pariétal et occipital ; chacun gère un segment spécifique des tâches du cerveau.

Le cervelet, attaché au tronc cérébral, est en dessous et derrière le cerveau (cerebrum). Il contrôle la fonction motrice, la capacité du corps à s'équilibrer et sa capacité à interpréter les informations envoyées au cerveau par les yeux, les oreilles et d'autres organes sensoriels.

Les fonctions régies par le tronc cérébral comprennent la respiration, la pression artérielle, certains réflexes et les changements qui se produisent dans le corps au cours de ce que l'on appelle la réponse «combat ou fuite». Le tronc cérébral est également divisé en plusieurs sections distinctes: le mésencéphale, pons et medulla oblongata.

Le diencéphale est à l'intérieur du cerveau au-dessus du tronc cérébral. Ses tâches comprennent la fonction sensorielle, le contrôle de l'apport alimentaire et le cycle du sommeil du corps. Comme avec les autres parties du cerveau, il est divisé en sections. Celles-ci comprennent le thalamus, l'hypothalamus et l'épithalamus.

Le cerveau est protégé contre les dommages par plusieurs couches de défenses. Les plus externes sont les os du crâne. Sous le crâne sont les méninges, une série de membranes solides qui entourent le cerveau et la moelle épinière. À l'intérieur des méninges, le cerveau est amorti par le liquide cérébro-spinal ou céphalo-rachidien.

2. Réseau de neurones

2.1. Introduction

Un réseau de neurones peut être défini comme un modèle de raisonnement basé sur le cerveau humain. Principalement, notre cerveau se compose d'un ensemble intensif de cellules nerveuses connectées les unes aux autres. Chez l'homme, Le cerveau contient environ 10 milliards et 60 trillions de connexions (synapses) [4]. Lorsque le cerveau utilise plusieurs neurones en parallèle, il peut accomplir des tâches beaucoup plus rapidement que les ordinateurs les plus rapides de nos jours.

2.2. Structure des neurones [5]

Un neurone a un corps cellulaire, une structure d'entrée ramifiée (la dendrite) et une structure de sortie ramifiée (l'axone). Les axones se connectent aux dendrites via des synapses. Les signaux électrochimiques se propagent à partir de l'entrée dendritique, à travers le corps cellulaire, et descendent l'axone vers d'autres neurones (cf. Figure 2.2).

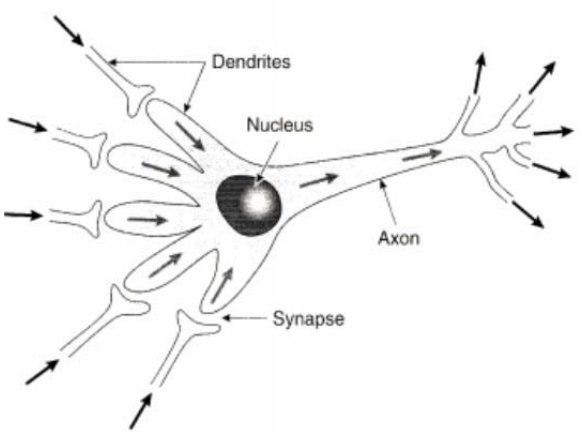


Figure 2.2 : Structure du neurone biologique

2.3. Neurones et synapses [6]

L'unité de calcul de base dans le système nerveux est la cellule nerveuse ou neurone. Un neurone possède en entrées des dendrites, en sortie un axone et un corps cellulaire. Un motoneurone, ou neurone moteur illustré par la figure 2.3, est une cellule nerveuse qui est directement connectée à un muscle et commande sa contraction. Il peut agir sur un petit ou un grand nombre de fibres musculaires, l'ensemble étant appelé une unité motrice. Les motoneurones contrôlent donc les mouvements du corps.

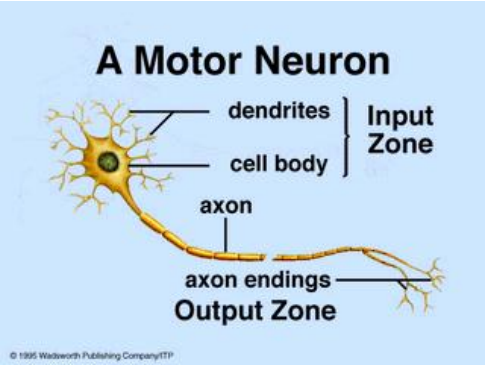


Figure 2.3 : Description d'un motoneurone

Un neurone reçoit l'entrée d'autres neurones (typiquement plusieurs milliers). Les entrées sont sommées (plus ou moins). Une fois que l'entrée dépasse un niveau critique, le neurone décharge un pic - une impulsion électrique qui se déplace du corps, vers le bas de l'axone, vers le ou les neurones suivants (ou d'autres récepteurs). Cet événement de dopage est également appelé dépolarisation, et est suivi d'une période réfractaire durant laquelle le neurone est incapable d'émettre à nouveau une décharge.

Les terminaisons axonales (zone de sortie) touchent presque les dendrites ou le corps cellulaire du neurone suivant. La transmission d'un signal électrique d'un neurone à l'autre est effectuée par des **neurotransmetteurs**, des produits chimiques qui sont libérés du premier neurone et qui se lient aux récepteurs dans le second. Ce lien s'appelle une **synapse**. La mesure dans laquelle le signal d'un neurone est transmis à un autre dépend de nombreux facteurs, par exemple, la quantité de neurotransmetteur disponible, le nombre et l'arrangement des récepteurs, la quantité de neurotransmetteur réabsorbé, etc.

Un neurone se déclenche uniquement si son signal d'entrée dépasse une certaine quantité (le seuil) sur une courte période de temps. Les synapses varient en force ; de bonnes connexions autorisent un signal important tandis que des connexions légères ne permettent qu'un signal faible. Les synapses peuvent être excitatrices ou inhibitrices.

2.4. Le cerveau en tant que système de traitement de l'information [6]

Le cerveau humain contient environ 80 milliards de cellules nerveuses, ou neurones. En moyenne, chaque neurone est connecté à d'autres neurones à travers environ 10 000 synapses. Les chiffres réels varient considérablement, selon la neuroanatomie locale. Le réseau de neurones du

cerveau forme un système massif de traitement de l'information en parallèle. Cela contraste avec les ordinateurs classiques, dans lesquels un seul processeur exécute une seule série d'instructions.

Par contre, considérons le temps nécessaire pour chaque opération élémentaire. Les neurones fonctionnent typiquement à une fréquence maximale d'environ 100 Hz, tandis qu'un processeur conventionnel effectue plusieurs centaines de millions d'opérations au niveau de la machine par seconde. En dépit d'être construit avec un matériel très lent, le cerveau a des capacités assez remarquables :

- Ses performances ont tendance à se dégrader gracieusement sous des dommages partiels. En revanche, la plupart des programmes et des systèmes d'ingénierie sont fragiles : si vous supprimez arbitrairement certaines parties, très probablement l'ensemble cessera de fonctionner.
- Il peut apprendre (se réorganiser) de l'expérience.
- Une récupération partielle des dommages est ainsi possible si des unités en bonne santé peuvent apprendre à reprendre les fonctions précédemment exercées par les zones endommagées.
- Il effectue des calculs massivement parallèles avec une efficacité extrême. Par exemple, une perception visuelle complexe survient en moins de 100 ms.
- Il soutient notre intelligence et notre conscience de soi. Personne ne sait encore comment cela se produit !

3. Réseau de neurones artificiel : un aperçu

3.1. Qu'est-ce qu'un réseau de neurones artificiel ? [7]

Quelle est l'utilité de cette nouvelle technologie? C'est une question naturelle à poser chaque fois qu'une technique émergente, telle que les réseaux de neurones, est transférée des laboratoires de recherche à l'industrie. Les définitions de base seront d'abord présentées: neurones (formels), réseaux de neurones, formation en réseaux de neurones (supervisés et non supervisés), réseaux avec propagation avant (feed-forward) et rétroaction (ou récurrents).

Un réseau neuronal artificiel (RNA ou ANN : Artificial Neural Network) est un paradigme de traitement de l'information qui s'inspire de la façon dont les systèmes nerveux biologiques, tels que le cerveau, traitent l'information. L'élément clé de ce paradigme est la structure originale du système de traitement de l'information. Il est composé d'un grand nombre d'éléments de traitement

(neurones) hautement interconnectés travaillant à l'unisson pour résoudre des problèmes spécifiques. Les RNA, comme les êtres humains, apprennent par l'exemple. Un RNA est configuré pour une application spécifique, telle que la reconnaissance de modèle ou la classification de données, à travers un processus d'apprentissage. L'apprentissage dans les systèmes biologiques implique des ajustements aux connexions synaptiques qui existent entre les neurones. Cela est également vrai pour les RNA.

Enfin, diverses applications seront décrites pour illustrer la diversité des domaines où les réseaux de neurones peuvent apporter des solutions efficaces et élégantes à des problèmes d'ingénierie tels que la reconnaissance de formes, les essais non destructifs, le filtrage d'informations, le génie biologique, la formulation matérielle, la modélisation de procédés industriels, le contrôle environnemental, la robotique, etc.

3.2. Contexte historique [7]

Les simulations de réseaux neuronaux semblent être un développement récent. Cependant, ce domaine a été créé avant l'avènement des ordinateurs, et a survécu à au moins un revers majeur et plusieurs époques.

Le premier neurone artificiel a été produit en 1943 par le neurophysiologiste Warren McCulloch et le logicien Walter Pits. Toutefois, la technologie disponible à ce moment-là ne leur permettait pas de réaliser des systèmes complexes.

Minsky et Papert, ont publié, en 1969, un livre dans lequel ils résument un sentiment général de frustration (à l'égard les réseaux neuronaux) parmi les chercheurs, et ont donc été acceptés par la plupart sans autre analyse. Avec l'avancée des technologies micro- et nano- électroniques (loi de Moore), les domaines d'application des réseaux de neurones connaissent, actuellement, un regain d'intérêt et une augmentation correspondante des financements associés.

4. Réseau de neurones artificiels

4.1. Définitions et propriétés des RNA [7]

Un neurone est une fonction non linéaire, paramétrée, liée. Par commodité, une fonction paramétrée linéaire est souvent appelée un neurone linéaire. Les variables du neurone sont souvent appelées entrées du neurone, qui fournit sa valeur en sortie. Un neurone peut être commodément représenté graphiquement comme le montre la figure 2.4. Cette représentation provient de

l'inspiration biologique qui a suscité l'intérêt initial pour les neurones formels, entre 1940 (McCulloch, 1943) et 1970 (Minsky 1969).

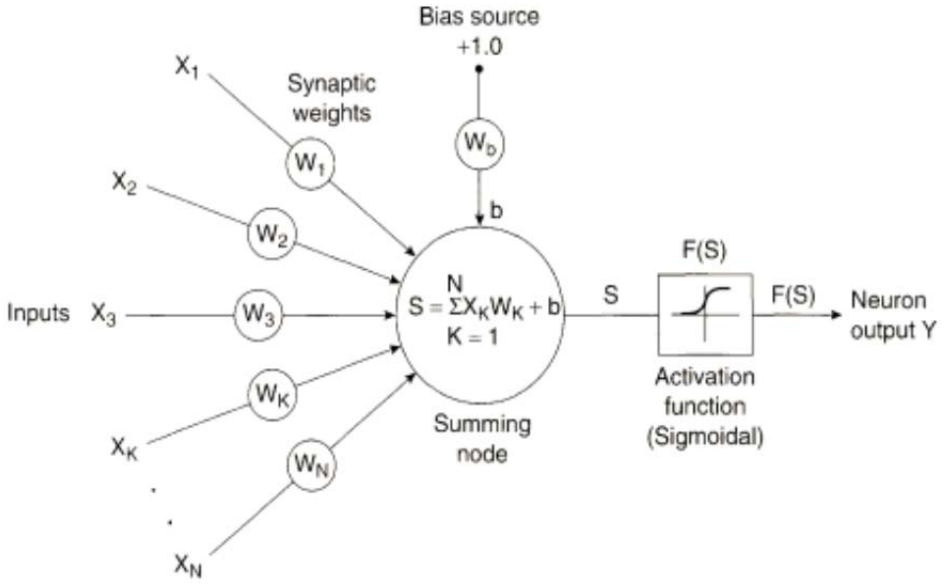


Figure 2.4 : Structure d'un neurone artificiel

Mathématiquement, l'expression de sortie du réseau est donnée par l'équation 2.1.

$$Y = F(S) = F(\sum_{K=1}^{N} X_K W_K + b) \qquad \text{(rel 2.1)}$$

Où les X_K sont les variables et les W_K sont les paramètres (ou poids) du neurone. La fonction F peut être paramétrée de manière appropriée. Deux types de paramétrage sont d'usage courant.

Les paramètres sont assignés aux entrées des neurones; la sortie du neurone est une combinaison non linéaire des entrées X_K, pondérée par les paramètres W_K, qui sont souvent appelés poids, ou, pour être réminiscents de l'inspiration biologique des réseaux de neurones, des poids synaptiques. Le potentiel le plus fréquemment utilisé v est une somme pondérée des entrées, avec un terme constant supplémentaire b appelé «biais».

La fonction F est appelée fonction d'activation. Pour des raisons qui seront expliquées ultérieurement, il est souhaitable que la fonction F soit une fonction sigmoïde (c.-à-d. une fonction en forme de s), telle que la fonction tanh ou la fonction tangente inverse.

4.2. Pourquoi utiliser des réseaux de neurones ? [7]

Les réseaux neuronaux, avec leurs capacités remarquables à dériver le sens de données compliquées ou imprécises, peuvent être utilisés pour extraire des motifs et détecter des tendances trop complexes pour être remarquées par les humains ou d'autres techniques informatiques. Un

réseau de neurones formé (i.e. ayant appris) peut être considéré comme un «expert» dans la catégorie d'informations qui lui a été donné d'analyser. Cet expert peut ensuite être utilisé pour fournir des projections compte tenu des nouvelles situations d'intérêt et répondre aux questions «et si». Parmi les autres avantages, nous pouvons citer :

- Apprentissage adaptatif : Capacité d'apprendre à effectuer des tâches en fonction des données fournies pour la formation ou l'expérience initiale.
- Auto-organisation : Un RNA peut créer sa propre organisation ou représentation de l'information qu'il reçoit pendant le temps d'apprentissage.
- Fonctionnement en temps réel : Les calculs RNA peuvent être effectués en parallèle et des dispositifs matériels spécifiques peuvent être conçus et fabriqués pour en tirer parti.
- Tolérance aux pannes via des informations redondantes de codage. La destruction partielle d'un réseau entraîne la dégradation correspondante des performances. Cependant, certaines capacités du réseau peuvent être conservées même avec des dommages majeurs sur le réseau.

4.3. Où sont utilisés les réseaux de neurones ? [5]

Le nombre d'application des réseaux de neurones ne semble limité que par l'imagination humaine. Toutes les tâches du quotidien on industrielles peuvent être réalisées par ce type d'approche, à l'instar de l'intelligence artificielle (deep ou machine learning). Sans être exhaustif, nous donnons ci-après quelques exemples typiques de cette technologie :

- Traitement du signal: suppression du bruit de ligne, suppression adaptative de l'écho, séparation de la source aveugle, …
- Contrôle et automatisation industriel : Siemens utilise avec succès les réseaux de neurones pour l'automatisation des processus dans les industries de base, par exemple, dans le contrôle des laminoirs, plus de 100 réseaux de neurones font leur travail, 24 heures sur 24
- Véhicule autonome à terme, mais déjà l'aide à la conduite
- Robotiques - navigation, reconnaissance de la vision
- Reconnaissance de motifs, c'est-à-dire la reconnaissance de caractères manuscrits ; par exemple la version actuelle de la Pomme de Newton utilise un réseau neuronal
- Médecine avec déjà de nombreuses application comme le stockage des dossiers médicaux en fonction de l'information sur le cas
- Production de discours: lecture de textes à haute voix (NETtalk)
- Reconnaissance de la parole
- Vision: reconnaissance faciale, détection des contours, moteurs de recherche visuelle

- Affaires où par exemple les règles relatives aux décisions hypothécaires sont extraites de décisions antérieures prises par des évaluateurs expérimentés, ce qui donne lieu à un réseau ayant un niveau d'accord élevé avec des experts humains
- Applications financières: analyse de séries temporelles, prédiction boursière
- Compression de données: texte, son et images
- Jeu: backgammon, échecs, go …

4.4. Réseaux de neurones versus algorithmes [8], [9]

Cette section présente quelques discussions sur le quand et le pourquoi de l'utilisation des réseaux de neurones. Il est montré que les réseaux de neurones sont simplement une nouvelle façon de résoudre les problèmes, ce qui peut être suivi avec succès pour un nombre élevé de problèmes. Pour certains problèmes, il n'est cependant pas utile d'utiliser une approche neuronale.

Comme mentionné précédemment, un réseau de neurones est essentiellement une méthode pour résoudre un problème. Les méthodes conventionnelles pour résoudre les problèmes sont basées sur la modélisation du problème, suivie généralement d'une dérivation analytique de l'algorithme qui implémente une solution du problème.

La principale différence entre l'utilisation de réseaux de neurones et l'utilisation d'une méthode conventionnelle pour résoudre un problème est que:

- Les réseaux de neurones sont *formés* pour fonctionner de façon satisfaisante: dans une phase d'apprentissage, des exemples d'entraînement sont présentés aux réseaux et les poids du réseau de neurones sont adaptés par une règle d'apprentissage.
- Les méthodes conventionnelles utilisent généralement un modèle (analytique ou empirique) de la tâche. En utilisant ce modèle, un algorithme est dérivé qui est suffisant pour résoudre le problème.

La mise en œuvre de la solution (neuronale ou conventionnelle) peut être soit logicielle, soit matérielle, soit mixte avec un partitionnement. La figure 2.5 montre les deux façons de mettre en œuvre une solution dédiée à un problème spécifique. Il s'ensuit qu'une manière neuronale de mettre en œuvre une solution à un problème est une manière alternative de la façon classique, algorithmique, afin de résoudre un problème.

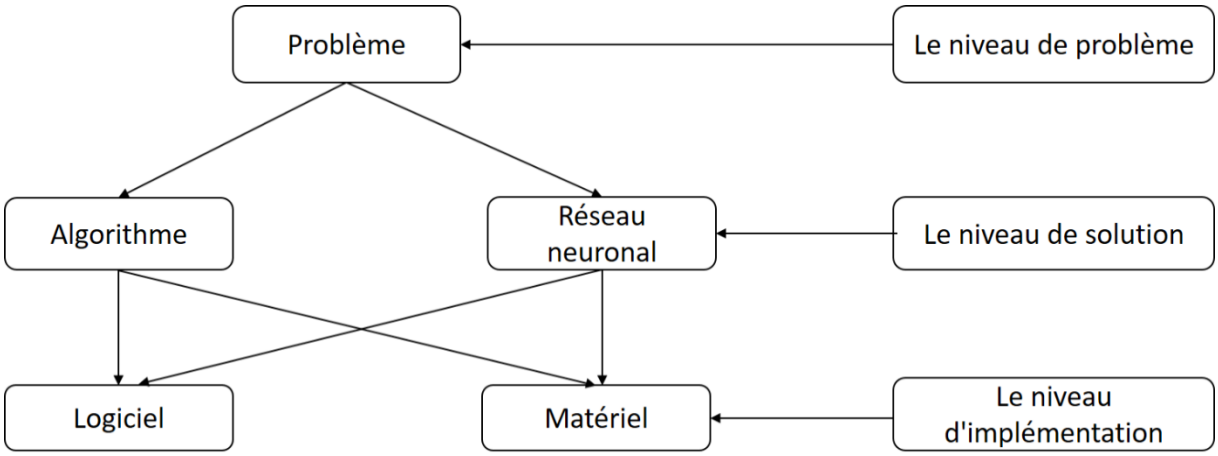


Figure 2.5 : Résolution d'un problème par une approche neuronal ou algorithmique

5. Similitudes entre neurones humains et neurones artificiels

5.1. Comment le cerveau humain apprend ? [7]

On ne sait pas encore grand-chose sur la façon dont le cerveau s'entraîne à traiter l'information, de sorte que les théories abondent. Dans le cerveau humain, un neurone typique recueille les signaux des autres à travers une foule de structures fines appelées dendrites. Le neurone envoie des pointes d'activité électrique à travers une longue et mince fibre connue sous le nom d'axone, qui se divise en milliers de branches (cf. Figure 2.6-a). À la fin de chaque branche, une structure appelée synapse convertit l'activité de l'axone en effets électriques qui inhibent ou excitent l'activité dans les neurones connectés (cf. Figure 2.6-b). Quand un neurone reçoit une entrée excitatrice suffisamment grande par rapport à son entrée inhibitrice, il envoie un pic d'activité électrique sur son axone. L'apprentissage se produit en changeant l'efficacité des synapses de sorte que l'influence d'un neurone sur un autre change.

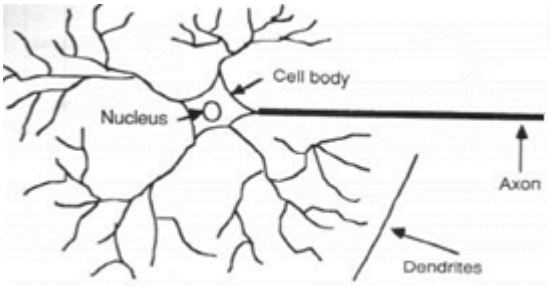


Figure 2.6-a : Composants d'un neurone

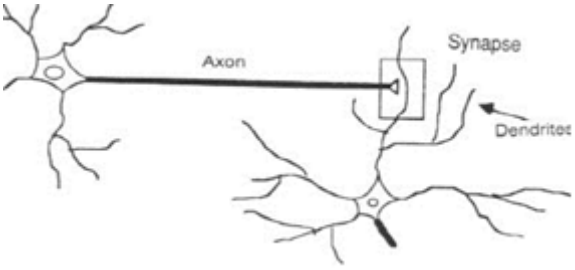


Figure 2.6-b : Connexion par synapse

5.2. Des neurones humains aux neurones artificiels [7]

Nous conduisons ces réseaux neuronaux en essayant d'abord de déduire les caractéristiques essentielles des neurones et de leurs interconnexions. Nous programmons ensuite typiquement un ordinateur pour simuler ces caractéristiques. Cependant, parce que notre connaissance des neurones est incomplète et que notre puissance de calcul est limitée, nos modèles sont nécessairement des approximations grossières de réseaux réels de neurones. La figure 2.7 illustre le synoptique d'un type de neurone artificiel.

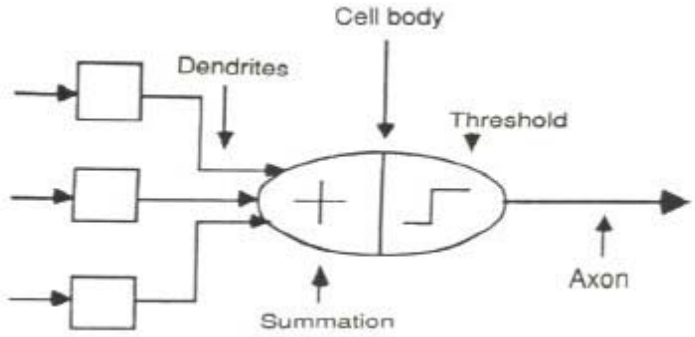


Figure 2.7 : Modèle neuronal

5.3. Evolution du nombre de neurones [10]

Les données sur les réseaux de neurones dédiés aux systèmes d'Intelligence Artificielle (AI) demeurent souvent confidentielles. Estimer ce nombre, à partir des publications, reste une gageure. Prenons toutefois la valeur de 86 millions comme chiffre de référence en 2016, pour le nombre de neurones et 150 milliards celui de synapses [10], et comparons ce chiffre à celui du cerveau humain. Bien que pour ce dernier, les valeurs puissent être également contestées, les nombres de neurones et synapses dans un cerveau humain sont respectivement de l'ordre de 86 milliards et 150 trillons.

De ces chiffres certes approximatifs, nous pouvons faire une projection sur la date de la convergence entre la complexité d'un réseau de neurones artificiel et le cerveau humain. En considérant la loi de Moore (le nombre de transistors sur une puce de silicium double tous les 2 ans, ou tous les 18 mois), clé de voute de l'industrie du semi-conducteur depuis les années 1960, le tableau 4.1 résume cette projection.

En considérant cette projection, nous pouvons constater que cette convergence devrait avoir lieu vers 2031 (ou 2036 si le doublement a lieu tous les 2 ans). Est-ce que nous pourrons dire alors que l'IA aura dépassé l'intelligence humaine ? Nous ne répondrons bien évidemment pas à cette question qui n'est pas qu'une affaire de nombre. Notons toutefois que certains y ont déjà répondu et

que nous leur en laissons la paternité! Ainsi par exemple, le directeur de l'ingénierie de Google, Ray Kurzweil, a déclaré lors du festival SXSW que la théorie de la singularité technologique, qu'il défend, se réalisera en 2029 : https://www.01net.com/actualites/ray-kurzweil-espere-la-fin-de-l-humanite-pour-dans-douze-ans-1125964.html.

Tableau 2.1 : Evolution prévisionnelle du nombre de neurones dans un réseau artificiel

Année (Tous les 2 ans)	Année (Tous les 18 mois)	Nombre de neurones
2016	2106	86 000 000
2018	2017,5 (2017 et 6 mois)	172 000 000
2020	2019	344 000 000
2022	2020,5	688 000 000
2024	2022	1 376 000 000
2026	2023,5	2,752,000,000
2028	2025	5,504,000,000
2030	2026,5	11,008,000,000
2032	2028	22,016,000,000
2034	2029,5	44,032,000,000
2036	2031	88,064,000,000

6. Réseau de neurones versus ordinateur [7]

Les réseaux de neurones adoptent une approche différente de la résolution de problèmes que celle des ordinateurs conventionnels. Ces derniers utilisent une approche algorithmique, c'est-à-dire que l'ordinateur suit un ensemble d'instructions afin de résoudre un problème. À moins que les étapes spécifiques, que l'ordinateur doit suivre, soient connues, l'ordinateur ne peut pas résoudre le problème. Cela limite la capacité de résolution de problèmes des ordinateurs classiques à des problèmes que nous comprenons déjà et que nous savons résoudre. Toutefois, les ordinateurs seraient beaucoup plus utiles s'ils pouvaient faire des choses que nous avons du mal à appréhender.

Les réseaux de neurones traitent l'information de la même manière que le cerveau humain. Le réseau est composé d'un grand nombre d'éléments de traitement hautement interconnectés (neurones) travaillant en parallèle pour résoudre un problème spécifique. Les réseaux de neurones apprennent par l'exemple. Ils ne peuvent être programmés que pour effectuer une tâche spécifique. Lors de la phase d'apprentissage, les exemples doivent être sélectionnés avec soin, sinon le temps utile est gaspillé ou pire encore, le réseau peut fonctionner incorrectement. L'inconvénient est que, puisque le réseau découvre comment résoudre le problème par lui-même, son fonctionnement peut être imprévisible.

Par contre, les ordinateurs conventionnels utilisent une approche cognitive pour résoudre les problèmes; la manière dont le problème doit être résolu doit être connue et énoncée en forme de

petites instructions non ambiguës. Ces instructions sont ensuite converties en un langage de programmation haut niveau, puis en code machine que l'ordinateur peut comprendre. Ces machines sont totalement prévisibles. Si quelque chose ne va pas, c'est à cause d'un défaut logiciel ou matériel.

Les réseaux de neurones et les ordinateurs algorithmiques conventionnels ne sont pas en compétition mais se complètent. Certaines tâches sont plus adaptées à une approche algorithmique comme les opérations arithmétiques et d'autres tâches plus adaptées aux réseaux de neurones. Plus encore, un grand nombre de tâches, nécessitent des systèmes qui utilisent une combinaison des deux approches (normalement un ordinateur conventionnel est utilisé pour superviser le réseau de neurones) afin de fonctionner avec une efficacité maximale. Le tableau 2.2 résume quelques différences entre les deux approches.

Tableau 2.2 : Comparaison entre les réseaux de neurones et les ordinateurs classiques

Cerveau	**PC classique**
Vprop = 100 m/s	Vprop = $3*10^8$ m/s
ω_N = 100 Hz	$\omega_N = 10^9$ Hz
N = 10^{10} à 10^{11} neurones	N = 10^9 octets
Degré de parallélisme # 10^{14} : soit 10^{14} processeurs à une fréquence de 100 Hz 104 connectés en même temps	

7. Une approche d'ingénierie [6]

7.1. Un neurone

Un des neurones le plus sophistiqué est le modèle McCulloch et Pitts (MCP), où les entrées sont «pondérées» (cf. Figure 2.8). L'effet de chaque entrée sur la prise de décision dépend du poids de cette entrée particulière. Le poids d'une entrée est un nombre qui, multiplié par l'entrée, donne l'entrée pondérée. Ces entrées pondérées sont ensuite additionnées et si elles dépassent une valeur seuil prédéfinie, le neurone se déclenche. Dans tous les autres cas, le neurone ne se déclenche pas ; il n'y a pas d'activation.

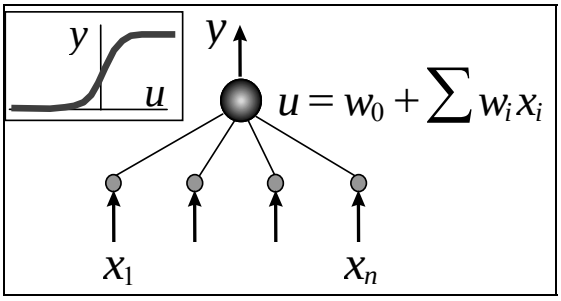


Figure 2.8 : Neurone avec entrées «pondérées»

En termes mathématiques, le neurone se déclenche, si et seulement si, la sortie dépasse le seuil de décision. L'ajout de poids d'entrée et de seuil fait de ce neurone un neurone très souple et puissant. Le neurone est régi par l'équation mathématique suivante :

$$y = f\left(\sum_{j=1}^{d} w_j x_j + w_0\right) = f\left(\sum_{j=1}^{d} w_j x_j\right) \quad \text{(rel 2.2)}$$

Le neurone MCP a la capacité de s'adapter à une situation particulière en changeant son poids et/ou son seuil. Divers algorithmes existent qui font que le neurone «s'adapte»; les plus utilisés sont la règle Delta et la propagation des erreurs de retour. Le premier est utilisé dans les réseaux avec Propagation Avant et le dernier dans les réseaux avec rétroaction.

7.2. Architectures d'un réseau de neurones

7.2.1. Réseau avec Propagation Avant ou Feed-Forward

Les RNA avec propagation avant permettent aux signaux de se déplacer dans un seul sens; de l'entrée vers la sortie. Il n'y a pas de retour (boucles), c'est-à-dire que la sortie de n'importe quelle couche n'affecte pas cette même couche. Un tel mécanisme est illustré par la figure 2.9.

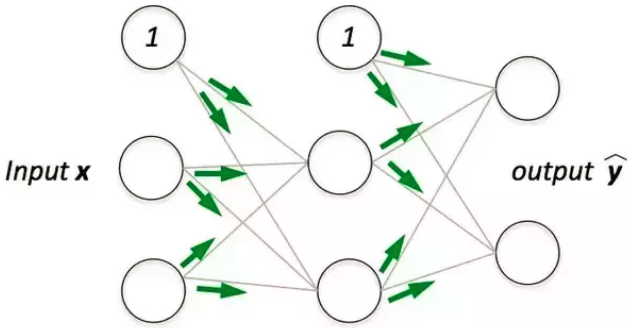


Figure 2.9 : Réseau avec Propagation Avant

Les RNA avec propagation avant tendent à être des réseaux simples qui associent les entrées aux sorties. Ils sont largement utilisés dans la reconnaissance de formes. Ce type d'organisation est également appelé bottom-up (bas-haut) ou top-down (haut-bas).

7.2.2. Réseau avec Rétroaction ou Back-Propagation

Les réseaux avec rétroaction, illustrés par la figure 2.10, peuvent avoir des signaux qui circulent dans les deux directions en introduisant des boucles dans le réseau.

Les architectures avec rétroaction sont également appelées interactives ou récurrentes, bien que ce dernier terme soit souvent utilisé pour désigner les connexions de rétroaction dans les organisations monocouches.

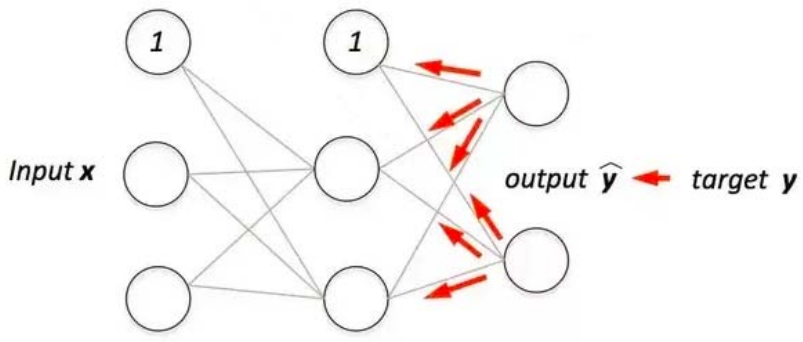


Figure 2.10 : Réseau avec Rétroaction

Les réseaux avec rétroaction sont très puissants et peuvent devenir extrêmement compliqués. Ils sont dynamiques ; leur «état» change continuellement jusqu'à ce qu'ils atteignent un point d'équilibre. Ils restent au point d'équilibre jusqu'à ce que l'apport change et qu'un nouvel équilibre soit trouvé.

7.2.3. Couches du réseau

Le type de réseau neuronal artificiel le plus courant consiste en trois groupes, ou couches d'unités : une couche d'unités "**d'entrée**" est connectée à une couche d'unités "**cachées**", connectée elle-même à une couche d'unités "**de sortie**".

L'activité des unités d'entrée représente les informations brutes qui sont introduites dans le réseau. L'activité de chaque unité cachée est déterminée par les activités des unités d'entrée et les poids sur les connexions entre les unités d'entrée et les unités cachées. Le comportement des unités de sortie dépend de l'activité des unités cachées et des poids entre les unités cachées et les unités de sortie.

Ce type de réseau simple est intéressant car les unités cachées sont libres de construire leurs propres représentations de l'entrée. Les poids entre les unités d'entrée et les unités cachées déterminent l'instant où chaque unité cachée est active. Ainsi, en modifiant ces poids, une unité cachée peut choisir ce qu'elle représente.

Nous distinguons également les architectures monocouche et multicouche. L'organisation monocouche, dans laquelle toutes les unités sont connectées les unes aux autres, constitue le cas le plus général et présente un potentiel de calcul plus important que les organisations multicouches hiérarchiquement structurées. Dans les réseaux multicouches, les unités sont souvent numérotées par couche, au lieu de suivre une numérotation globale.

7.2.4. Réseau Perceptron

Le travail, sans doute le plus influent, sur les réseaux neuronaux date des années 60 ; il est connu sous le terme de «Perceptron», un terme inventé par Frank Rosenblatt. Le perceptron s'avère être un modèle MCP (neurone avec des entrées pondérées) avec un prétraitement supplémentaire fixe. Les unités appelées A_1, A_2, A_j, A_p sont appelées unités d'association et leur tâche consiste à extraire des éléments spécifiques localisés des images d'entrée comme le montre la figure 2.11. Les perceptrons imitent le système de vision des mammifères. Ils étaient principalement utilisés pour la reconnaissance de formes même si leurs capacités sont en fait plus étendues.

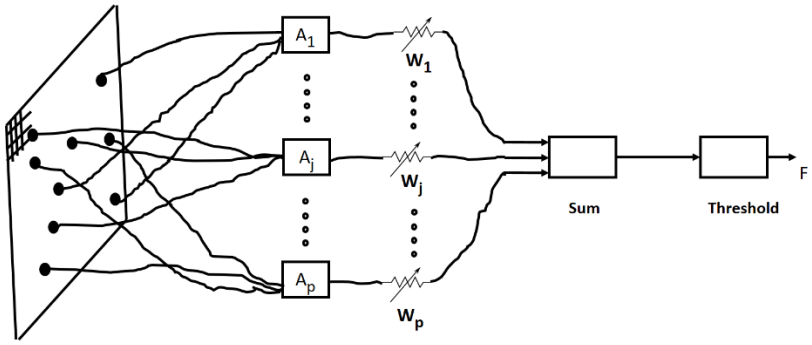


Figure 2.11 : Modèle MCP – Neurone avec entrées pondérées

En 1969, Minsky et Papert ont écrit un livre dans lequel ils ont décrit les limites des Perceptrons monocouches. L'impact de ce livre a été énorme et a fait perdre beaucoup d'intérêt à de nombreux chercheurs sur les RNA. Le livre, très bien rédigé, montrait mathématiquement que les perceptrons à une seule couche ne pouvaient pas faire certaines opérations de reconnaissance de formes de base comme déterminer si une forme est connectée ou non (par exemple pour séparer les nuages d'une constellation d'une transmission numérique). Ce qu'ils n'avaient pas réalisé, est que, vu l'entraînement approprié, les perceptrons multiniveaux peuvent faire ces opérations. Ce point fut corrigé dans les années 80.

8. Le processus d'apprentissage

8.1. Types d'apprentissage

Toutes les méthodes d'apprentissage utilisées pour les réseaux de neurones adaptatifs peuvent être classées en deux grandes catégories :

- **Apprentissage Supervisé** : il incorpore un « enseignant » externe, de sorte que chaque unité de sortie soit informée de ce que devrait être sa réponse souhaitée aux signaux d'entrée. Pendant le processus d'apprentissage, des informations globales peuvent être

requises. Les paradigmes de l'apprentissage supervisé comprennent l'apprentissage par correction d'erreurs, l'apprentissage par renforcement et l'apprentissage stochastique.

- **Apprentissage Non-Supervisé** : il n'utilise aucun « enseignant » externe et est basé uniquement sur des informations locales. Il est également appelé auto-organisation, dans le sens où il auto-organise les données présentées au réseau et détecte leurs propriétés collectives émergentes. Les paradigmes de l'apprentissage non supervisé sont l'apprentissage hebbien et l'apprentissage compétitif.

L'aspect de l'apprentissage concerne la distinction ou non d'une phase séparée, au cours de laquelle le réseau est formé, et une phase d'exploitation ultérieure. Nous disons qu'un réseau de neurones apprend hors ligne si la phase d'apprentissage et la phase d'opération sont distinctes. Un réseau de neurones apprend en ligne s'il apprend et fonctionne en même temps. Habituellement, l'apprentissage supervisé est effectué hors ligne, tandis que l'apprentissage non supervisé est effectué en ligne.

8.2. Fonction de transfert

Le comportement d'un RNA dépend à la fois des poids et de la fonction d'entrée-sortie (fonction de transfert) qui est spécifiée pour les unités. Cette fonction appartient généralement à l'une des trois catégories suivantes :

- linéaire (ou rampe)
- seuil
- sigmoïde : $f(a) = 1/\left(1+\exp(-a)\right)$

Pour les **unités linéaires**, l'activité de sortie est proportionnelle à la sortie pondérée totale.

Pour les **unités de seuil**, les sorties sont définies à l'un des deux niveaux, selon que l'entrée totale est supérieure ou inférieure à une valeur seuil.

Pour les **unités sigmoïdes**, la sortie varie continuellement mais pas linéairement lorsque l'entrée change. Les unités sigmoïdes ressemblent davantage aux vrais neurones que les unités linéaires ou à seuil, mais toutes trois doivent être considérées comme approximatives.

Nous pouvons « enseigner » un réseau à trois couches pour effectuer une tâche particulière en suivant les trois étapes suivantes. Premièrement, nous présentons le réseau avec des exemples d'entraînement, qui consistent en un modèle d'activités pour les unités d'entrée ainsi que le modèle d'activités souhaité pour les unités de sortie. Deuxièmement, nous déterminons dans quelle mesure

la sortie réelle du réseau correspond à la sortie souhaitée. Enfin, nous changeons le poids de chaque connexion pour que le réseau produise une meilleure approximation de la sortie désirée.

Afin d'apprendre à un réseau de neurones à effectuer certaines tâches, nous devons ajuster les poids de chaque unité de manière à réduire l'erreur entre la sortie désirée et la sortie réelle. Ce processus nécessite que le réseau de neurones calcule l'erreur qui dérive des poids, dénommée **EP** (Erreur Poids ou EW en anglais). En d'autres termes, il doit calculer comment l'erreur change à mesure que chaque poids augmente ou diminue légèrement. L'algorithme de propagation inverse est la méthode la plus largement utilisée pour déterminer l'**EP**.

L'algorithme de rétro-propagation est le plus facile à comprendre, si toutes les unités du réseau sont linéaires. L'algorithme calcule chaque **EP** en calculant d'abord l'**EA** (Erreur Amplitude, à savoir la différence entre la sortie réelle et la sortie désirée), la vitesse à laquelle l'erreur change lorsque le niveau d'activité d'une unité est modifié. Pour les unités de sortie, l'**EA** est simplement la différence entre la production réelle et la sortie désirée. Pour calculer l'**EA** pour une unité cachée dans la couche juste avant la couche de sortie, nous identifions d'abord tous les poids entre cette unité cachée et les unités de sortie auxquelles elle est connectée. Nous multiplions ensuite ces poids par les **EA** de ces unités de sortie et ajoutons les produits. Cette somme est égale à l'**EA** pour l'unité cachée choisie. Après avoir calculé toutes les **EA** dans la couche cachée juste avant la couche de sortie, nous pouvons calculer de la même manière les **EA** pour les autres couches, en se déplaçant d'une couche à une autre dans la direction opposée à la propagation des activités dans le réseau, d'où le terme de rétro-propagation. Une fois que l'**EA** a été calculée pour une unité, il est simple de calculer l'**EP** pour chaque connexion entrante de l'unité. L'**EP** est le produit de l'**EA** et de l'activité via la connexion entrante. Ces mécanismes, avant et arrière, sont détaillés dans le paragraphe suivant.

9. Concevoir des modèles RNA [11]

9.1. Introduction

La figure 2.12 illustre la procédure systémique pour concevoir un RNA composée de cinq étapes de base :

1. Collection de données
2. Prétraitement des données
3. Construction du réseau
4. Formation (apprentissage)
5. Test de la performance du modèle

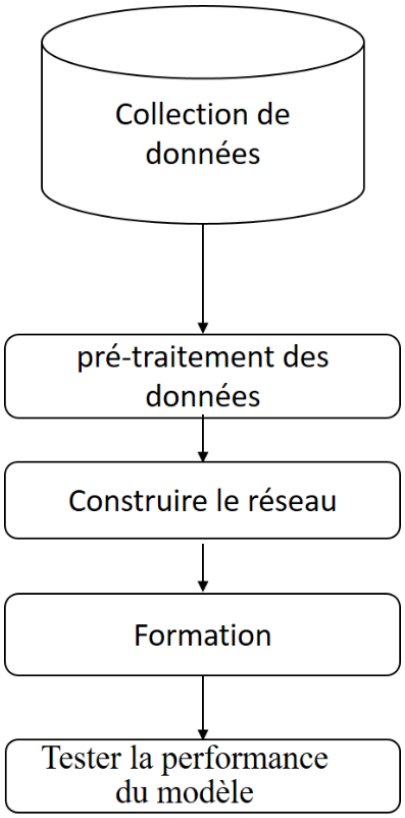


Figure 2.12 : Procédé de conception d'un modèle de réseau neuronal artificiel

9.2. Réseau de neurones avec Propagation Avant et Rétro-Propagation [12]

Un réseau de propagation inverse apprend par l'exemple. Il convient de donner à l'algorithme des exemples de ce que vous voulez que le réseau fasse et il change les poids du réseau de sorte que, lorsque l'entraînement est terminé, il vous donnera la sortie requise pour une entrée particulière. Les réseaux de propagation de retour sont idéaux pour les tâches simples de reconnaissance de formes. Comme mentionné précédemment, pour former le réseau, il est nécessaire de lui donner des exemples de ce que vous voulez, i.e. la sortie que vous voulez (appelée la cible) pour une entrée particulière.

Ainsi, si nous mettons le premier motif sur le réseau, nous aimerions que la sortie soit 0 ou 1 comme le montre la figure 2.13 (par exemple, un pixel noir est représenté par 1 et un blanc par 0). L'entrée et sa cible correspondante sont appelées une paire d'entraînement.

Une fois le réseau formé, il fournira la sortie désirée pour n'importe quel motif d'entrée. Regardons maintenant comment fonctionne cette phase d'apprentissage, également appelée la formation. Le réseau est d'abord initialisé en configurant tous ses poids pour qu'ils soient de petits nombres aléatoires, prenons entre -1 et +1.

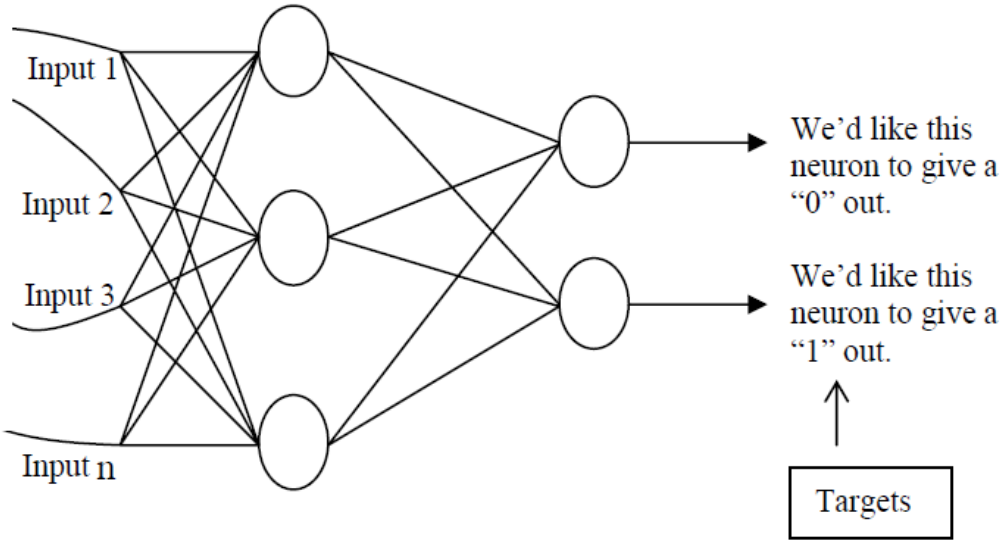


Figure 2.13 : Application d'une paire d'entrainement à un réseau

Ensuite, le motif d'entrée est appliqué et la sortie est calculée ; c'est ce qu'on appelle la passe avant. Le calcul donne une sortie aléatoire qui n'aucune raison d'être correcte. Sa valeur est complètement différente de ce que vous désirez (la cible), puisque tous les poids sont aléatoires. Nous calculons ensuite l'erreur de chaque neurone entre la cible et la sortie obtenue (i.e. la différence entre la Cible et la Sortie réelle, c'est-à-dire Ce que vous voulez - Ce que vous obtenez réellement). Cette erreur est ensuite utilisée intelligemment pour changer les poids de sorte que l'erreur devienne plus petite. En d'autres termes, la sortie de chaque neurone se rapprochera de sa cible (cette partie est appelée la passe inverse). Ce processus itératif est répété jusqu'à ce que l'erreur soit minimisée.

Prenons un exemple à partir d'un réseau réel pour voir comment le processus fonctionne. Nous observerons tout d'abord une connexion, entre un neurone dans la couche de sortie et un autre dans la couche cachée, comme le montre la figure 2.14.

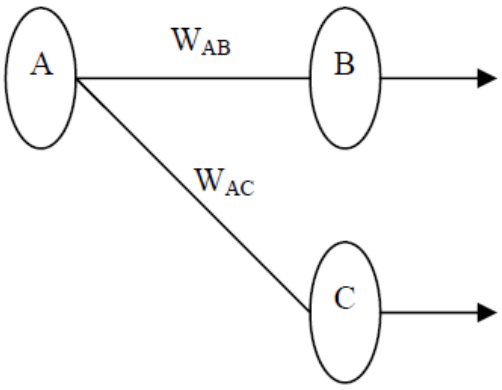


Figure 2.14 : Connexion unique apprenant dans un réseau avec une rétro-propagation

La connexion qui nous intéresse est entre le neurone A (un neurone de la couche cachée) et le neurone B (un neurone de sortie) ; elle a un poids W_{AB}. Le diagramme montre également une autre connexion entre les neurones A et C ; nous y reviendrons plus tard. L'algorithme fonctionne suivant les 5 étapes suivantes :

1. Appliquer d'abord les entrées au réseau et travaillez sur la sortie. Rappelons qu'à l'initialisation, la valeur de cette sortie pourrait être n'importe quoi, car les poids initiaux étaient des nombres aléatoires (compris entre -1 et 1).
2. Calculer l'erreur pour le neurone B, soit la différence entre la Cible et la Sortie réelle.
   Dans la pratique, nous calculons : ErreurB = SortieB (1-SortieB) (CibleB – SortieB)
   Le terme "sortie(1-sortie)" est nécessaire dans l'équation en raison du choix de la fonction d'activation (fonction sigmoïde). Si nous n'utilisions qu'un seul un neurone de seuil, ce serait juste (Cible - Sortie).
3. Calculer le nouveau poids W^{+}_{AB} formé à partir du poids initial W_{AB} (cf. rel 2.3) :

   $$W^{+}_{AB} = W_{AB} + (\text{ErreurB} \times \text{SortieA}) \qquad \text{(rel 2.3)}$$

   Notez que c'est la sortie du neurone de connexion (neurone A) que nous utilisons (pas celle de B). Nous mettons à jour tous les poids dans la couche de sortie de cette manière.
4. Calculer les erreurs pour les neurones de la couche cachée. Contrairement à la couche de sortie, nous ne pouvons pas les calculer directement (parce que nous n'avons pas de cible), donc nous les propageons à partir de la couche de sortie (d'où le nom de l'algorithme). Ceci est fait en prenant les erreurs des neurones de sortie et en les ré-exécutant à travers les nouveaux poids pour obtenir les erreurs des couches cachées.
   Par exemple si le neurone A est connecté comme montré à B et C alors nous prenons les erreurs de B et C pour générer une erreur pour A suivante la formule (rel 2.4).

   $$\text{ErreurA} = \text{Sortie A} (1 - \text{SortieA}) (\text{ErreurB}^* W_{AB} + \text{ErreurC}^* W_{AC}) \qquad \text{(rel 2.4)}$$

   Encore une fois, le facteur "Sortie(1 - Sortie)" est présent en raison de la fonction d'écrasement sigmoïde.
5. Après avoir obtenu l'erreur pour les neurones de la couche cachée, il suffit de procéder comme à l'étape 3 pour modifier les poids de cette couche cachée et remonter ainsi jusqu'aux neurones d'entrée. En répétant cette méthode, nous pouvons former un réseau de n'importe quel nombre de couches.

Pour lever toute ambigüité sur ce mécanisme, prenons l'exemple d'un réseau de taille réduite avec 2 entrées, 3 neurones de couches cachées et 2 neurones de sortie. Cet exemple est illustré par la figure 2.15.

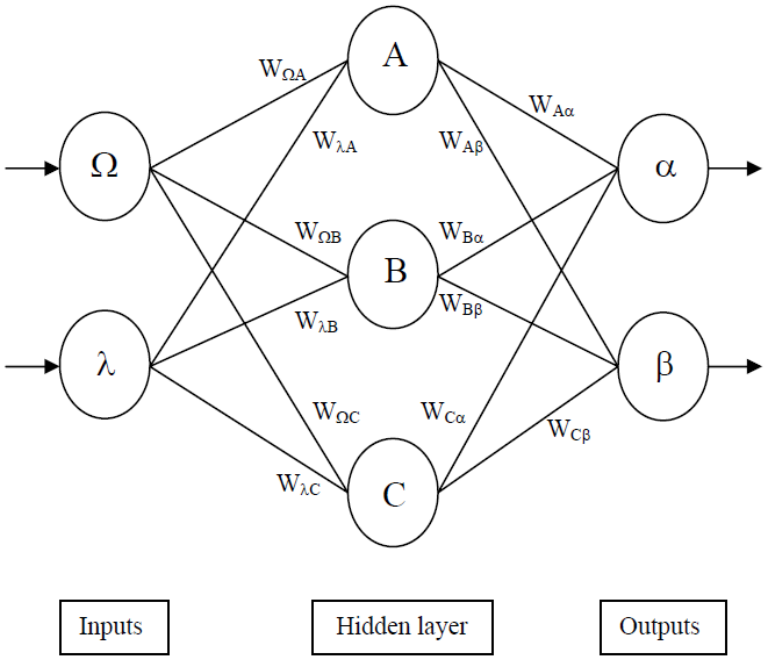


Figure 2.15 : Différents poids pour une passe inverse de rétro-propagation

Dans cet exemple, W^+ représente le nouveau poids recalculé, tandis que W (sans l'exposant +) représente l'ancien poids. L'ensemble des calculs nécessaires pour une itération complète de rétro-propagation est résumé dans les 4 étapes et les 17 équations (de rel 2.5 à rel 2.21) suivantes :

1. Calculer les erreurs des neurones de sortie

$$\delta_\alpha = sortie_\alpha\ (1 - sortie_\alpha)\ (Cible_\alpha - sortie_\alpha) \quad \text{(rel 2.5)}$$

$$\delta_\beta = sortie_\beta\ (1 - sortie_\beta)\ (Cible_\beta - sortie_\beta) \quad \text{(rel 2.6)}$$

2. Changer les poids de la couche de sortie

$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\ sortie_A \quad \text{(rel 2.7)}$$

$$W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\ sortie_A \quad \text{(rel 2.8)}$$

$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\ sortie_B \quad \text{(rel 2.9)}$$

$$W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\ sortie_B \quad \text{(rel 2.10)}$$

$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\ sortie_C \quad \text{(rel 2.11)}$$

$$W^+_{C\beta} = W_{C\beta} + \eta\delta_\beta\ sortie_C \quad \text{(rel 2.12)}$$

3. Calculer (retransmettre en inverse) les erreurs de la couche cachée

$$\delta_A = sortie_A\ (1 - sortie_A)\ (\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta}) \quad \text{(rel 2.13)}$$

$$\delta_B = sortie_B\ (1 - sortie_B)\ (\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta}) \quad \text{(rel 2.14)}$$

$$\delta_C = sortie_C\ (1 - sortie_C)\ (\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta}) \quad \text{(rel 2.15)}$$

4. Changer les poids de la couche cachée

$$W^{+}_{\lambda A} = W_{\lambda A} + \eta_{\delta A}\ \text{entrée}_{\lambda} \quad \text{(rel 2.16)}$$

$$W^{+}_{\Omega A} = W^{+}_{\Omega A} + \eta_{\delta A}\ \text{entrée}_{\Omega} \quad \text{(rel 2.17)}$$

$$W^{+}_{\lambda B} = W_{\lambda B} + \eta_{\delta B}\ \text{entrée}_{\lambda} \quad \text{(rel 2.18)}$$

$$W^{+}_{\Omega B} = W^{+}_{\Omega B} + \eta_{\delta B}\ \text{entrée}_{\Omega} \quad \text{(rel 2.19)}$$

$$W^{+}_{\lambda C} = W_{\lambda C} + \eta_{\delta C}\ \text{entrée}_{\lambda} \quad \text{(rel 2.20)}$$

$$W^{+}_{\Omega C} = W^{+}_{\Omega C} + \eta_{\delta C}\ \text{entrée}_{\Omega} \quad \text{(rel 2.21)}$$

La constante η (appelée taux d'apprentissage, et initialement égal à un) est mise en place pour accélérer ou ralentir l'apprentissage si nécessaire. En d'autres termes, le réseau continue d'entraîner tous les motifs de manière répétée jusqu'à ce que l'erreur totale tombe à une valeur cible basse prédéterminée (critère d'arrêt), puis s'arrête. Notez que lorsque vous calculez l'erreur finale utilisée pour arrêter le réseau (qui est la somme de toutes les erreurs de neurones individuelles pour chaque modèle), vous devez prendre toutes les erreurs en valeur absolue pour qu'elles s'additionnent et ne soient pas soustraites (une erreur de -0,5 est tout aussi mauvaise qu'une erreur de +0,5).

Une fois que le réseau ait été formé (i.e. qu'il ait appris, c'est-à-dire que les poids sont déterminés pour l'application cible), il devrait être capable de reconnaître non seulement les modèles parfaits, mais aussi les versions corrompues ou bruyantes. En fait, si nous ajoutons délibérément des versions bruyantes des modèles dans l'ensemble d'apprentissage lorsque nous formons le réseau (disons un sur cinq), nous pouvons améliorer les performances du réseau à cet égard. La formation (apprentissage) peut également bénéficier de l'application des modèles dans un ordre aléatoire en entrée du réseau.

Il y a une meilleure façon pour arrêter la formation du réseau, qui consiste à utiliser un ensemble de validation. Cela arrête le surentraînement du réseau (devenant trop précis, ce qui peut diminuer ses performances). Pour cela, on utilise un deuxième ensemble de modèles qui sont des versions bruyantes de l'ensemble d'entraînement (mais ne sont pas utilisés pour l'entraînement lui-même). Chaque fois que le réseau a été formé; cet ensemble (appelé l'ensemble de validation) est utilisé pour calculer une erreur. Lorsque l'erreur devient faible, le réseau s'arrête.

9.3. Problèmes liés à la rétro-propagation

Les trois principaux problèmes des réseaux de neurones sont liés aux nombreux paramètres à définir, à la notion de sur-ajustement et aux longs temps d'entraînement. Un problème important concernant l'apprentissage supervisé est le problème de la convergence d'erreurs, c'est-à-dire la

minimisation de l'erreur entre les valeurs unitaires désirées et calculées. Le but est de déterminer un ensemble de poids qui minimise l'erreur.

Ainsi, l'apprentissage supervisé par rétro-propagation a quelques problèmes associés. Le plus connu est, sans doute, le problème de la convergence des erreurs, appelé "Local Minima" ou minimum locaux. Cela se produit lorsque l'algorithme modifie toujours les poids de manière à provoquer la chute de l'erreur. Mais l'erreur pourrait brièvement augmenter dans le cadre d'une baisse plus générale, comme le montre la figure 2.16. Si tel est le cas, l'algorithme se "bloque" (car il ne peut pas monter) et l'erreur ne diminue pas davantage.

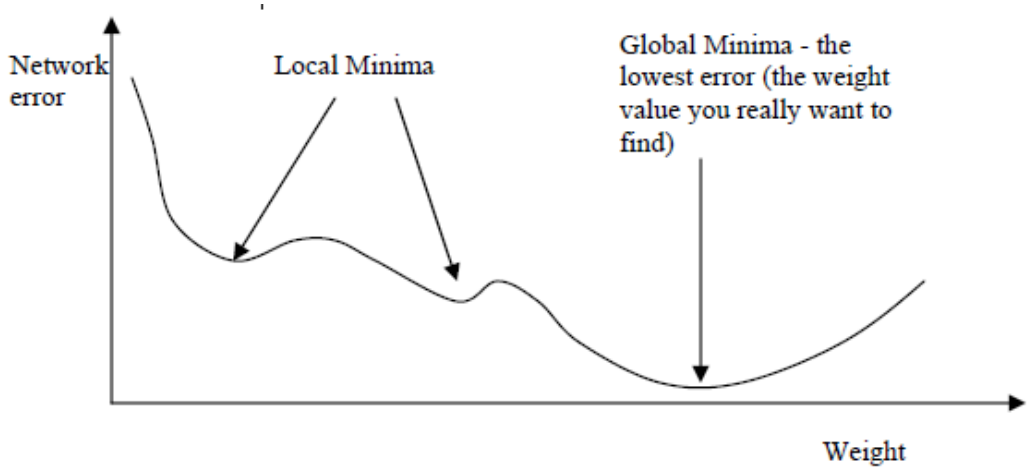


Figure 2.16 : Minimum locaux versus minimum global

Il existe plusieurs solutions à ce problème. Une est très simple et consiste à réinitialiser les poids à différents nombres aléatoires, puis à considérer le minimum minimorum (cela peut aussi conduire à plusieurs autres problèmes). Une autre solution pour minimiser l'erreur entre les valeurs unitaires désirées et calculées est de déterminer un ensemble de poids qui minimise l'erreur. Une méthode bien connue, qui est commune à de nombreux paradigmes d'apprentissage, est la convergence des moindres carrés (LMS).

10. Conclusion

Dans ce chapitre, nous avons présenté les différents types de réseaux de neurones artificiels, ainsi que leurs applications potentielles. Inspirés du fonctionnement du cerveau humain, ils permettent, entre autre, de résoudre des problèmes mal définis où les algorithmes classiques ne sont pas ou peu efficaces. Dans le cas de notre application, à savoir la classification de cellules cancéreuses en deux classes (bénigne ou maligne), le réseau avec rétro-propagation, appelé MLP,

semble le plus adapté. La phase d'apprentissage ou formation du réseau demeure un problème récurrent qui sera étudié au chapitre suivant.

Le MLP (Multi-Layer Perceptron) [14], réseau avec rétro-propagation à action directe, est la technique de réseau de neurones la plus utilisée pour la classification. Brièvement, les MLP sont des classifieurs d'apprentissage supervisé qui consistent en une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées qui extraient des informations utiles pendant l'apprentissage et assignent des coefficients de pondération modifiables aux composants des couches d'entrée. Lors du premier passage (direct), les poids attribués aux unités d'entrée et les nœuds dans les couches cachées et entre les nœuds dans la couche cachée et la sortie, déterminent la sortie. La sortie est comparée à la sortie cible. Un signal d'erreur est ensuite propagé en retour et les poids de connexion sont ajustés en conséquence. Au cours de la formation ou phase d'apprentissage, les MLP construisent un espace multidimensionnel, défini par l'activation des nœuds cachés, de sorte que les deux classes (malignes et bénignes) soient aussi séparables que possible.

Nous visons dans cette thèse à étudier les techniques d'apprentissage automatique. Nous utiliserons plusieurs algorithmes et les appliquerons à l'ensemble de données sur le cancer du sein du Wisconsin. Nous nous concentrerons sur la technique d'apprentissage automatique du réseau neuronal multicouche (rétro-propagation). Nous étudierons principalement ces différents algorithmes et analyserons leurs résultats.

La classification est l'une des tâches les plus importantes et essentielles dans l'apprentissage automatique et l'exploration de données. De nombreuses recherches ont été menées pour appliquer l'exploration de données et l'apprentissage automatique sur différents ensembles de données médicales afin de classer le cancer du sein. Beaucoup d'entre elles offrent une bonne précision de classification.

Dans le chapitre suivant, nous présenterons une comparaison entre les différents algorithmes en utilisant la MLP (Multi-Layer Perception), basée dans la base de données sur le cancer du sein (Wisconsin Breast Cancer : WBC) [15], en utilisant comme entrées les 9 attributs présentés au chapitre 1. Toutes les expériences sont menées avec le logiciel Matlab.

11. Bibliographie

[1] «https://www.livescience.com/29365-human-brain.html».

[2] https://www.humanbrainfacts.org/basic-structure-and-function-of-human-brain.php.

[3] «https://www.quora.com/What-is-the-main-function-of-the-brain».

[4] www.ai.nutn.edu.tw.

[5] Nic Schraudolph ,Fred Cummins, Introduction to Neural Networks.

[6] https://nic.schraudolph.org/teach/NNcourse/brain.html.

[7] Igor Aleksander , Helen Morton, An Introduction to Neural Computing Paperback, October 1, 1995.

[8] https://www.researchgate.net/publication/260575017_Neural_Networks_An_Overview.

[9] J. Annema, Feed-Forward Neural Networks: Vector Decomposition Analysis, Modelling and analog implementation.

[10] https://ai.stackexchange.com/questions/2330/when-will-the-number-of-neurons-in-ai-systems-equal-the-human-brain.

[11] https://www.hindawi.com/journals/ijas/2013/525383/.

[12] Sri Siddhartha,Visvesvaraya, «Training Feed forward Neural Network With Backpropogation Algorithm,» *International Journal Of Engineering And Computer Science,* т. 6 , № 1, pp. 19860-19866, Jan. 2017.

[13] Croall, Ian F., Mason, John P. , Industrial Applications of Neural Networks, Project ANNIE Handbook, 1992.

[14] Murtagh, Fionn, «Multilayer perceptrons for classification and regression,» *Neurocomputing,* т. 2, № 5-6, pp. 183-197, July 1991, Pages.

[15] http://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin/, Wisconsin.

Chapitre III – Architecture du RNA pour la détection du cancer

1. Introduction

Le Réseau de Neurones Artificiel (RNA) est un outil simulant le système nerveux biologique pour traiter l'information [1], [2]. Bénéficiant de résultats physiologiques sur le cerveau, le RNA simule un système non linéaire et dynamique pertinent pour atteindre des caractéristiques spécifiques. En outre, le réseau possède des capacités d'adaptation, d'auto-organisation et d'apprentissage [2], [3], [4], [5] et [6]. Le neurone artificiel est la base de RNA, qui peut être vu comme l'intégration de trois parties, à savoir un groupe d'entrées et de poids, une fonction de transfert telle qu'un additionneur et une fonction d'activation avec un seuil [2], [7], [8] et [9]. Le signe du poids représente l'état du composant concerné. Habituellement, le signe positif est le symbole de l'état activé et le signe négatif indique l'état non activé. La gamme normale de la sortie d'un neurone artificiel est soit l'intervalle de (-1, +1) soit le couple (0, 1) [10].

Lorsqu'il traite des informations, le réseau est en constante transformation. C'est un système dynamique complexe difficile à modéliser et simuler. Toutefois la société Mathworks propose, sous MATLAB, une boite à outils spécifique intitulée "Neural Network Toolbox" [11], ainsi qu'un outil d'interfaçage graphique associé dénommé "Neural Network Fitting". Elle fournit ainsi des sous-programmes de conception de réseau et d'apprentissage [1], [2], [5] et [8]. De nombreuses publications présentent la méthode Simulink pour créer un réseau neuronal, par exemple la fonction de base radiale (Radial Base Function ou RBF) [12] ou le réseau artificiel à la carte auto-organisatrice (self-organizing map SOM) [13].

En utilisant la base de données Wisconsin décrite au premier chapitre, l'objectif de ce chapitre est de déterminer la structure de base de chaque neurone et de proposer l'architecture du RNA. Dans un premier, nous retiendrons l'algorithme le mieux adapté à notre problème de classification de cellules cancéreuses. Une fois cet algorithme, et donc le neurone de base, déterminé, des simulations, utilisant les modèles Matlab développés précédemment, nous permettrons de définir les spécifications, cahier des charges, de chaque brique de base réalisant un neurone et une synapse du RNA. Trois réseaux neuronaux différents peuvent être appliqués pour l'approximation de la fonction de transfert , à savoir la rétro-propagation (BP) [14], le réseau de base radiale (RBF) [12] ou la régression généralisée Réseau Neuronal (GRNN) [15].

Notre choix s'est porté sur le RNA MLP avec rétro-propagation ou BP MLP, décrit en détail dans le chapitre précédent. Nous étudierons également le rôle du biais introduit dans l'équation de transfert. Nous utilisons la matrice de confusion pour comparer les résultats.

Rappelons que la sortie du neurone BP peut être exprimée sous la forme A=f(WP+b), où f est la fonction de transfert, représentant la relation entre l'entrée et la sortie, W est le poids et b le biais. La fonction de transfert est généralement désignée par une fonction log-sigmoïde (ou logsigmoid), tangente hyperbolique ou purement linéaire. La figure 3.1 illustre ces trois fonctions. Le processus d'apprentissage comporte deux parties : la première partie est l'apprentissage à travers le réglage de la structure, la deuxième partie consiste à ajuster le poids et le seuil à travers le gradient de la dernière couche [4], [6] et [9].

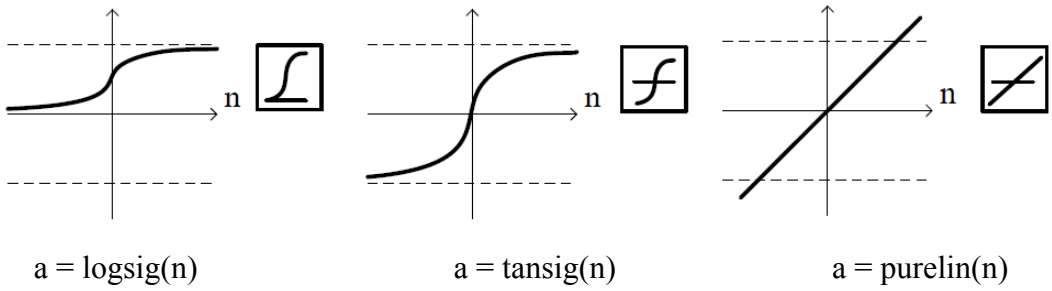


Figure 3.1 : Fonctions de transfert d'un neurone BP

2. Méthodologie proposée

2.1. RNA de référence

Fondamentalement, il y a différentes étapes nécessaires pour déterminer la fonction de transfert. La première étape est la collecte de données, qui doit être réalisée sous la forme d'une entrée matricielle. Dans notre cas, à partir de la base de données Wisconsin qui contient 699 biopsies, nous utilisons la commande LOAD cancer_dataset.MAT de la "Neural Network Toolbox" . Nous disposons ainsi de 9 entrées (9 attributs, décrits au chapitre 1, pour chaque biopsie) pour les 699 cas de la base de données (matrice totale d'entrées = 9*699) et 2*699 sorties (2 classes bénigne ou maligne). Rappelons que les 699 cas (699 biopsies) étaient répartis en 458 tumeurs bénignes (classe 1 : 65,5%) et 241 tumeurs malignes (classe 2 : 34,5%) [16].

Le réseau est ensuite créé, en utilisant là encore les outils Matlab. Ce réseau est illustré par la figure 3.2. Après cette configuration, les poids et les biais (nous reviendrons sur ce point dans la suite

de ce chapitre) sont initialisés. Sur cette figure, on constate que le nombre de neurones est de 9 pour la couche d'entrée, 10 pour celle cachée et 2 pour la couche de sortie.

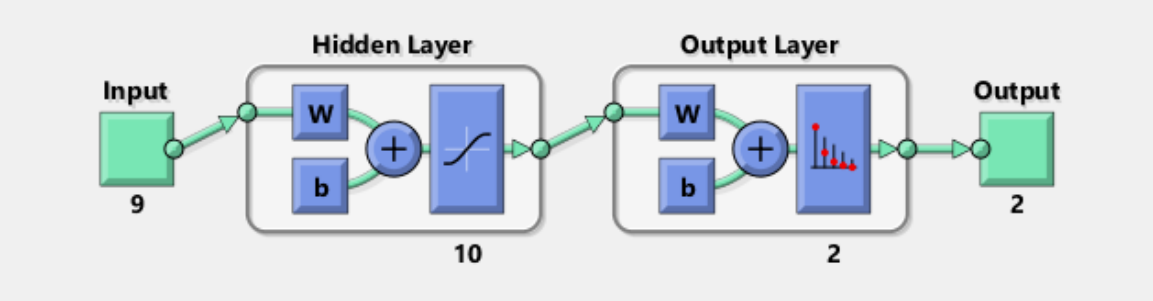


Figure 3.2 : Création d'un réseau de neurones sous Matlab

En outre, la formation et la validation sont effectuées en utilisant les données générées par la fonction de référence fournie dans la « toolbox » Matlab. Après la formation et la validation, le réseau de neurones est complet et prêt pour d'autres tests et analyses [8]. Dans un premier temps, nous avons utilisé la fonction SOFTMAX proposée par Matlab pour réaliser la fonction de transfert du neurone. Il convient de noter que cette fonction idéale utilise d'une part la fonction exponentielle et d'autre part la théorie des probabilités qu'il sera difficile, voire impossible, d'implémenter sous forme d'un circuit VLSI analogique à faible surface et faible consommation. Toutefois, les résultats obtenus nous servirons de référence pour le choix de notre algorithme final.

Les résultats de cette simulation sont résumés par la matrice de confusion, présentée sur la figure 3.3. Si on considère le résultat final, on peut constater que l'erreur totale est de 2,4%, qui sera donc notre valeur de référence.

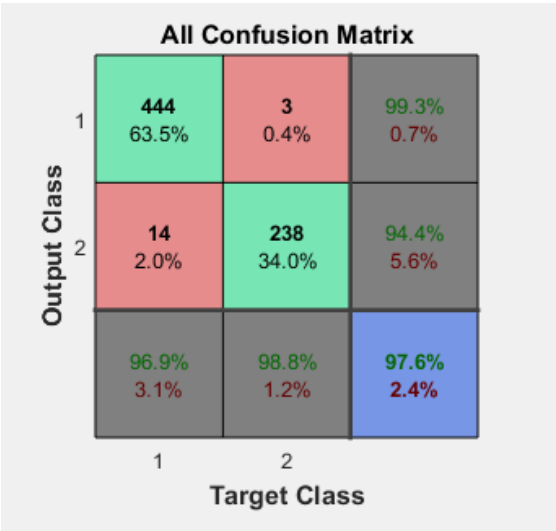


Figure 3.3 : Matrice de confusion

Une analyse plus fine montre que sur les 458 tumeurs bénignes, le réseau de neurones en a classé 444 comme bénignes (zone verte en haut à gauche de la figure 3.3) correspondant à une classification correcte. Le chiffre de 63,5% est à comparer aux 65,5% (458/699) de cellules saines, les autres cas bénins restants ont été considérés comme malins par le système de classification correspondant à une erreur (2% sur les 699). De même, concernant les biopsies avec des tumeurs malignes (241 sur la base de données), le système en a classé 238 correctement (soit 34% à comparer aux 34,5%) et a commis 3 erreurs (les 0,4% d'erreur supplémentaires).

2.2. Architecture avec ou sans biais

Notre premier objectif est de trouver l'architecture la plus simple possible pour réaliser la fonction de transfert qui puisse être très facilement intégrable par un circuit intégré analogique et offrant une réponse équivalente à celle obtenue sous Matlab. Nous conservons bien évidemment la même structure MLP BP du réseau de neurones, avec le même nombre d'entrée et de sortie, le même nombre de couches cachées, mais également pour différentes fonctions d'activation.

Dans un premier temps, nous utiliserons un algorithme classique avec biais. La sortie, y_i (avec i=1,2) de ce réseau de neurones MLP BP, illustré par la figure 3.4, est régie par l'équation (rel 3.1).

$$y_i = g\left(\sum_{j=1}^{m} w_{ji}^2\, h\left(n_j^1\right) + \theta_i^2\right) = g\left(\sum_{j=1}^{m} w_{ji}^2\; h\left(\sum_{k=1}^{K} w_{kj}^1\, x_k + \theta_j^1\right) + \theta_i^2\right) \qquad \text{(rel 3.1)}$$

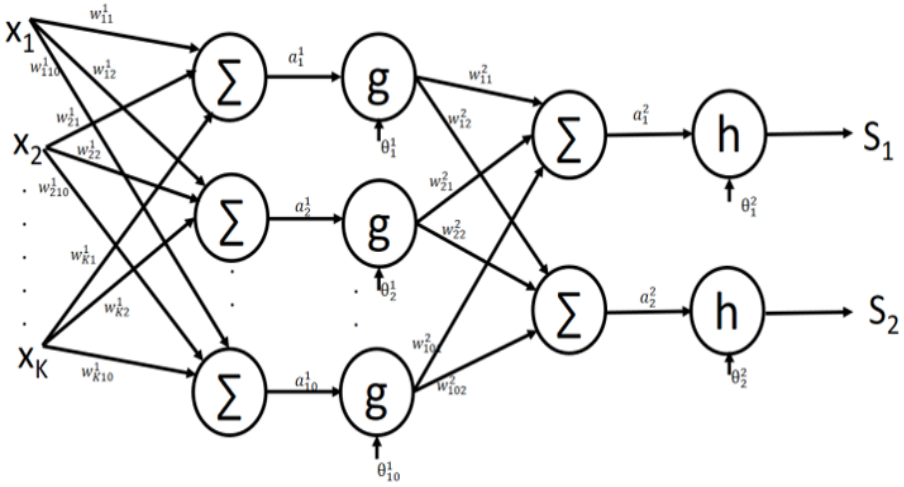


Figure 3.4 : Architecture du réseau neuronal avec biais

Afin de simplifier la structure du réseau de neurones et donc l'électronique qui l'implémentera, nous nous proposons, dans un second temps, de supprimer le biais et d'évaluer son influence sur le résultat final, à savoir la qualité de ce système de classification. La sortie de ce réseau de neurones MLP BP sans biais, illustré par la figure 3.5, est régie par l'équation (rel 3.2).

$$y_i = g\left(\sum_{j=1}^{m} w_{ji}^2\, h(n_j^1)\right) = g\left(\sum_{j=1}^{m} w_{ji}^2\; h(\sum_{k=1}^{K} w_{kj}^1\, x_k)\right) \qquad \text{(rel 3.2)}$$

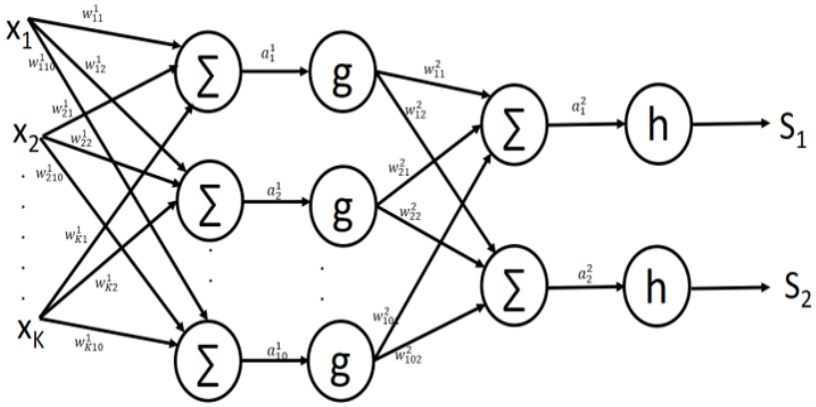


Figure 3.5 : Architecture du réseau neuronal sans biais

Pour chaque cas (avec et sans biais), deux programmes ont été développés sous Matlab, l'un en fixant la valeur minimale de l'erreur entre la cible et la sortie du réseau de neurone (variable appelée Erreur Destination) et le second en insérant un nombre spécifique d'époques. Une époque correspond à une passe complète (passe avant et arrière) de tous les exemples d'entraînement. Ces algorithmes sont illustrés respectivement par les figures 3.6 et 3.7. Ils seront testés pour différentes fonctions d'activation et avec pour entrées les valeurs résumées dans le tableau 3.1.

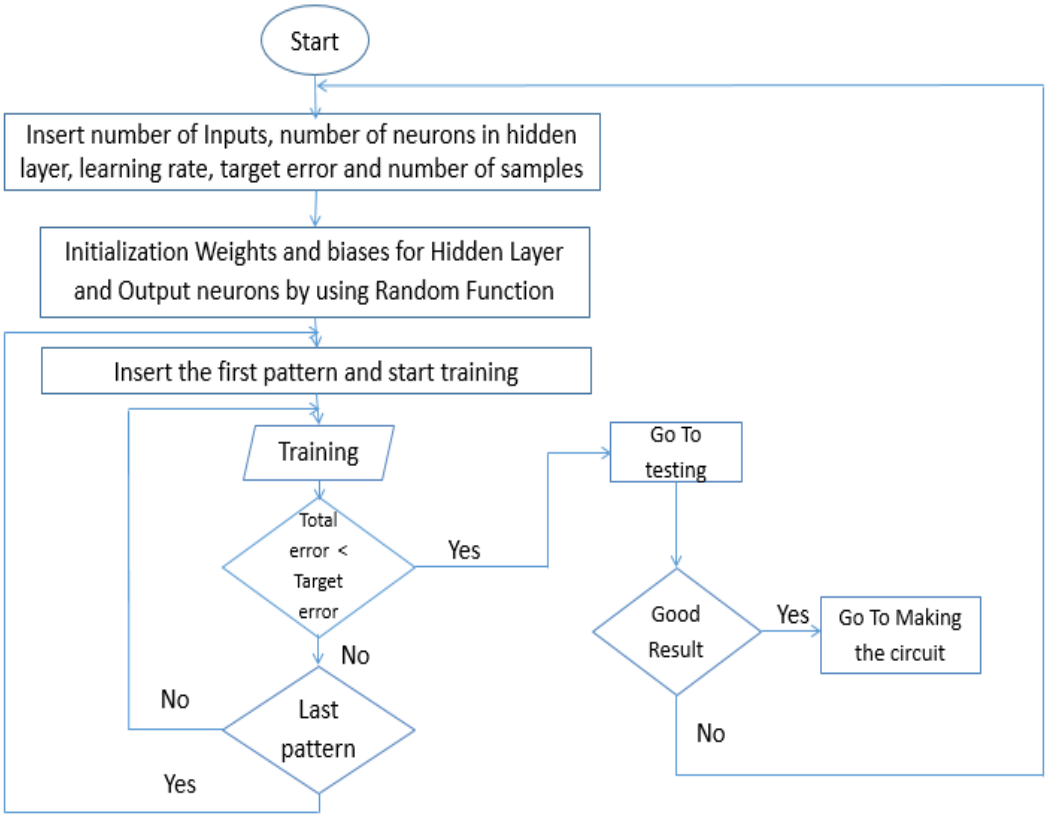


Figure 3.6 : Premier algorithme en fixant le critère d'arrêt Erreur Destination

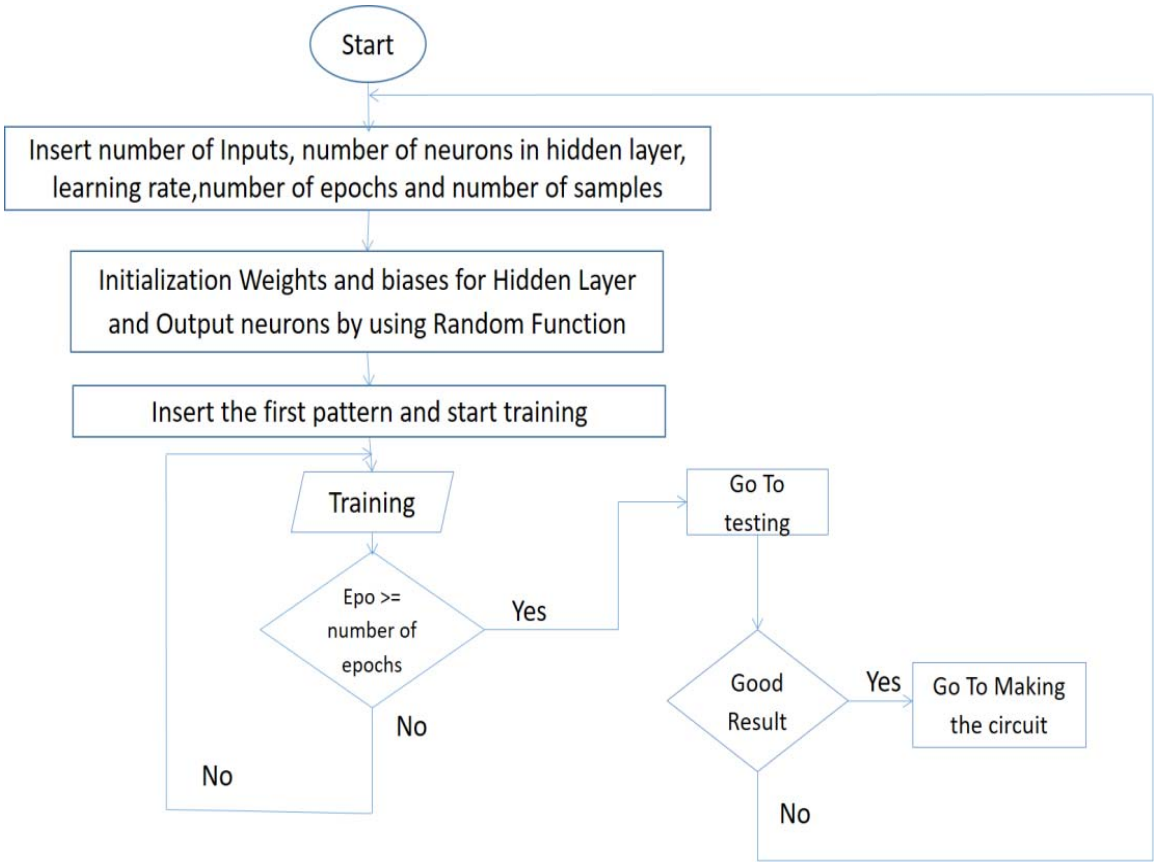


Figure 3.7 : Second algorithme en fixant le nombre d'époques

Tableau 3.1 : Entrées pour les deux algorithmes

Entrées pour le premier algorithme	**Entrées pour le second algorithme**
Nombre d'entrées: 9	Nombre d'entrées: 9
Nombre de Neurones Cachées: 10	Nombre de Neurones Cachées: 10
Nombre d'Itérations: 489	Nombre d'Itérations: 489
Erreur Destination: 0,1	Nombre d'Epoques: 14
Taux d'Apprentissage: 0,25	Taux d'Apprentissage: 0,25

Nous n'utiliserons que 489 cas (nombre d'itérations) pour la phase d'apprentissage et les 699 pour la validation finale. Toutes ces valeurs ont été optimisées sous Matlab.

L'objectif du paragraphe suivant est donc de déterminer, par des simulations Matlab, le choix de l'architecture du RNA que nous implémenterons ensuite avec des circuits intégrés VLSI analogiques. Notre exploration architecturale doit choisir entre les alternatives suivantes :

- Choix de la fonction de transfert : avec ou sans biais
- Choix de la fonction d'activation : linéaire, tanh ou sigmoid (logsigmoid en fait)
- Choix de l'algorithme d'apprentissage, en fixant l'erreur destination (E.D.) ou le nombre d'époques (Nb.E.).

3. Détermination sous Matlab du choix du RNA

3.1. Architecture avec biais

3.1.1. Fonction d'activation linéaire

La figure 3.8 décrit le premier RNA étudié. Il est basé sur l'algorithme avec biais (cf. rel 3.1) et une fonction d'activation linéaire appelée purelin. Cette fonction d'activation est utilisée pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.9.

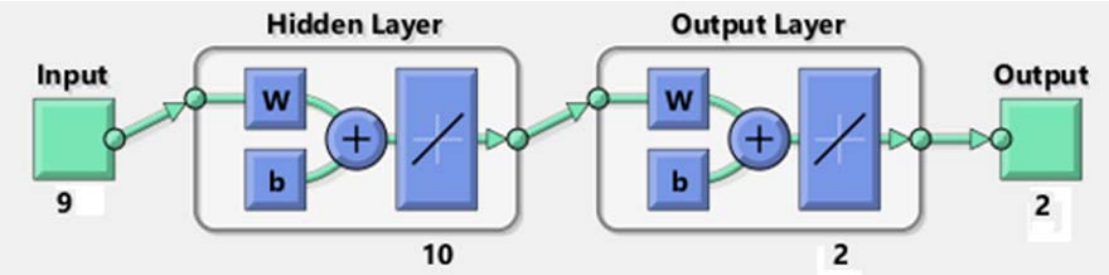
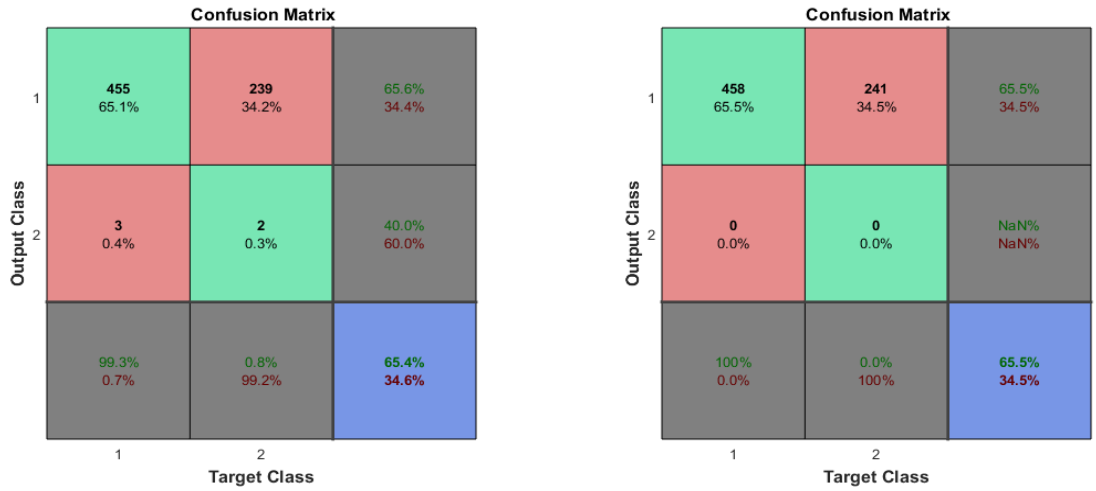


Figure 3.8 : Modèle Matlab du RNA avec biais et purelin comme fonction d'activation



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.9 : Matrice de confusion du RNA avec biais et purelin comme fonction d'activation

Si on considère le résultat final, on peut constater que l'erreur totale est de 34,6% pour le premier algorithme, valeur très supérieure à notre valeur de référence de 2,4%. Quant au second algorithme, même si le taux d'erreur est sensiblement le même (34,5%), il n'a détecté que des tumeurs bénignes, ce qui est rédhibitoire. Dans les deux cas, ce résultat n'est pas satisfaisant et ce RNA ne sera pas retenu comme architecture finale.

3.1.2. Fonction d'activation de type tangente hyperbolique

La figure 3.10 décrit le second RNA étudié. Comme dans le cas précédent, Il est toujours basé sur l'algorithme avec biais (cf. rel 3.1), mais cette fois la fonction d'activation est une tangente hyperbolique, utilisée à la fois pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.11.

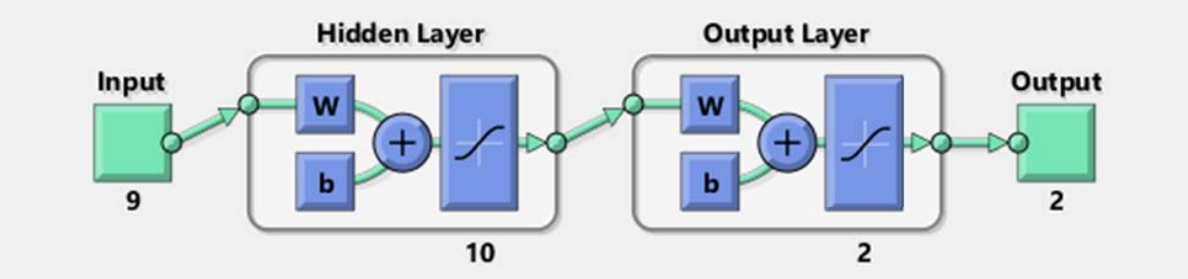
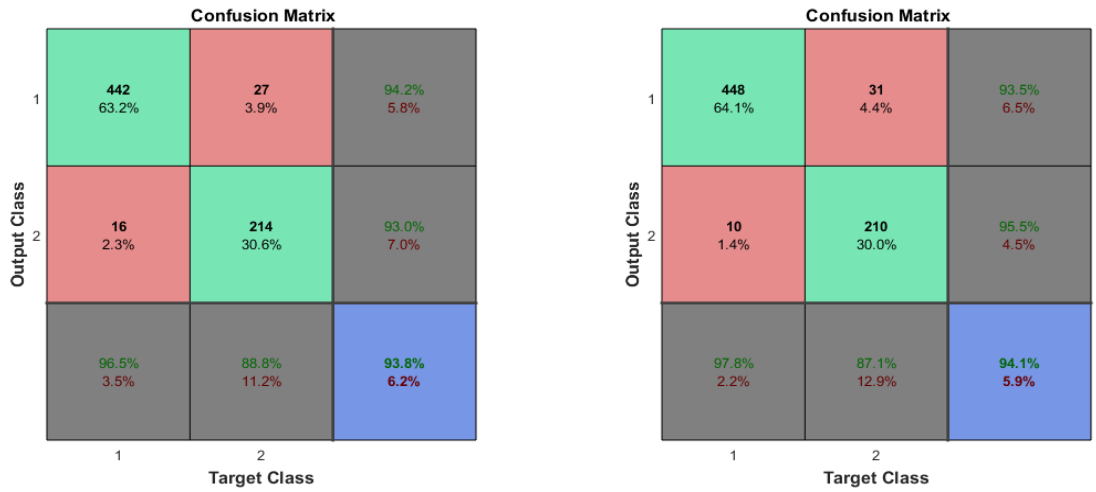


Figure 3.10 : Modèle Matlab du RNA avec biais et une tangente hyperbolique comme fonction d'activation



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.11 : Matrice de confusion du RNA avec biais et une tangente hyperbolique comme fonction d'activation

Si on considère le résultat final, on peut constater que l'erreur totale a très fortement diminuée par rapport au cas précédent. En outre, les deux algorithmes sont relativement équivalents ; ils fournissent des résultats similaires. Cette erreur totale n'est plus que de 6,2% ou 5,9%, mais demeure très supérieure à notre valeur de référence de 2,4%.

En conclusion, ce résultat n'est pas encore satisfaisant et ce RNA ne sera toujours pas retenu comme architecture finale. Il ne reste plus que la fonction logsigmoid comme fonction d'activation possible. Etudions maintenant ce dernier cas, où l'on choisit toujours la même fonction à la fois pour la couche cachée que pour les neurones de sortie.

3.1.3. Fonction d'activation de type logsigmoid

Pour ce troisième cas, le RNA étudié (cf. Figure 3.12) est là encore basé sur l'algorithme avec biais donné par l'équation (3.1). Toutefois, la fonction d'activation est maintenant une fonction de type logsigmoid, qui est utilisée pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.13.

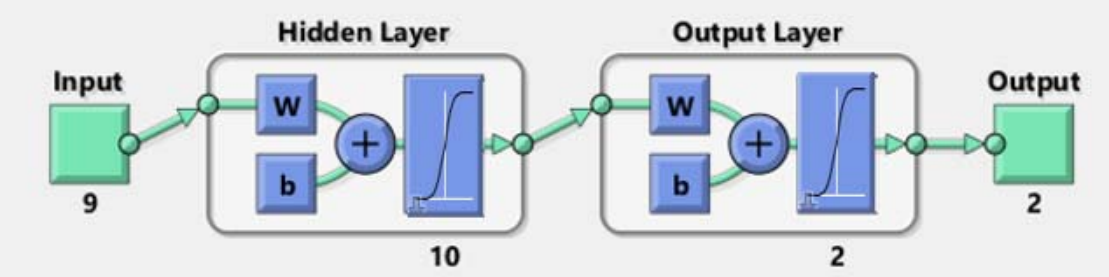
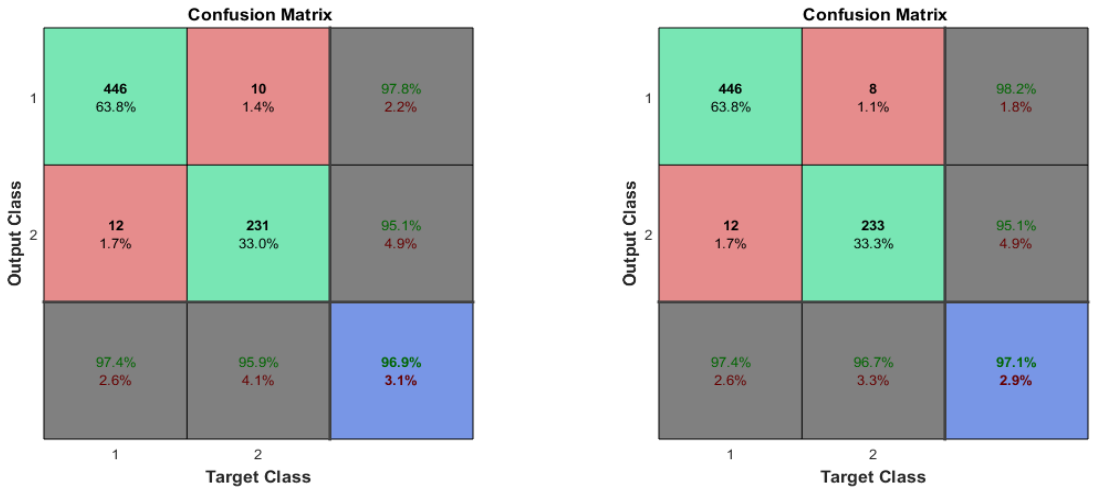


Figure 3.12 : Modèle Matlab du RNA avec biais et une fonction d'activation logsigmoid



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.13 : Matrice de confusion du RNA avec biais et une fonction d'activation logsigmoid

Le résultat final (case bleue) montre que l'erreur totale n'est plus que d'environ 3% et ceci quel que soit l'algorithme d'apprentissage (3,1% pour le premier et 2,9% pour le second). Cette erreur totale est relativement proche de notre référence de 2,4%. Ce résultat semble satisfaisant et ce RNA pourrait être retenu comme architecture finale.

On peut également constater, sur la matrice de confusion (cf. Figure 3.11), que le taux d'erreur est relativement identique pour les deux classes de tumeurs, à savoir 1,7% (dans les 2 cas) pour les tumeurs bénignes et 1,4% (pour le premier algorithme) et seulement 1,1% (pour le second) pour les tumeurs malignes.

Toutefois, avant d'arrêter notre choix, continuons nos investigations en gardant toujours l'algorithme avec biais mais en croisant le type des fonctions d'activation. Nous passerons ensuite au second algorithme sans biais.

3.1.4. Fonction logsigmoid pour la couche cachée et purelin pour celle de sortie

La figure 3.14 décrit ce quatrième RNA basé sur l'algorithme avec biais (cf. rel 3.1). Dans cette nouvelle configuration, la fonction d'activation de la couche cachée est de type logsigmoid et celle pour les neurones de sortie est linéaire ou purelin. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.15.

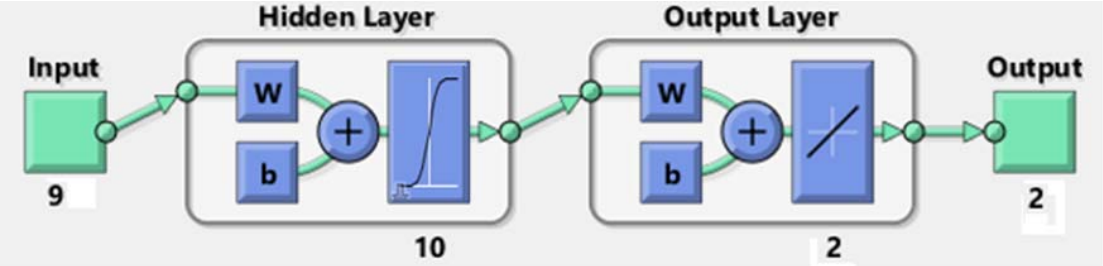
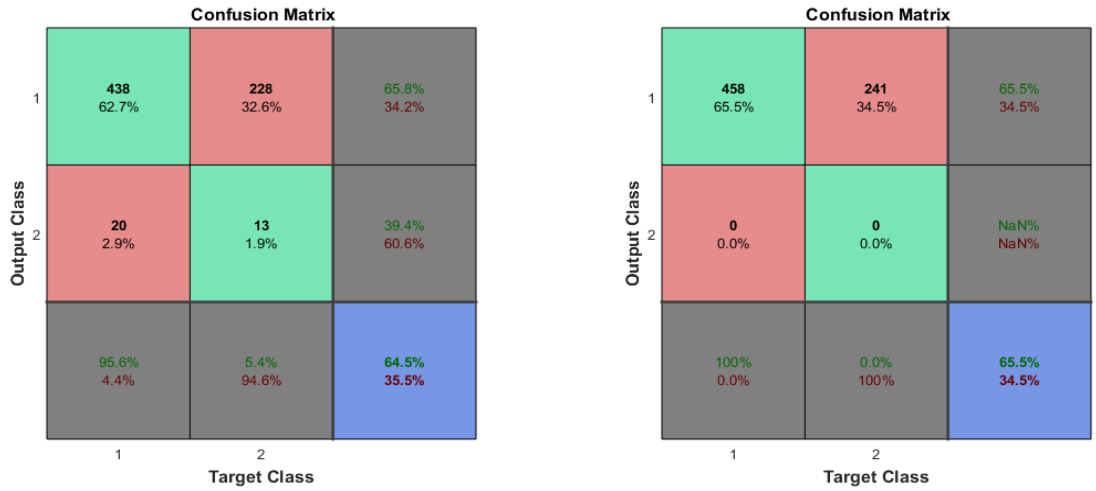


Figure 3.14 : Fonction d'activation logsigmoid pour la couche cachée et purelin pour les neurones de sortie



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.15 : Matrice de confusion du RNA avec biais et une d'activation logsigmoid pour la couche cachée et purelin pour les neurones de sortie

Avec une erreur totale de 35,5% et 34,5%, le résultat est tout aussi catastrophique que dans le premier cas où toutes les fonctions d'activation étaient linéaires (erreur totale d'environ 34,5%). Il semble que les fonctions d'activation linéaires soient à proscrire pour ce type d'application. Pour

vérifier cette hypothèse, testons le cas similaire en changeant seulement la fonction d'activation de la couche cachée par une fonction tangente hyperbolique.

3.1.5. Fonction tangente hyperbolique pour la couche cachée et purelin pour celle de sortie

La dernière configuration testée, illustrée par la figure 3.16, décrit un RNA basé sur l'algorithme avec biais (cf. rel 3.1). Dans cette nouvelle configuration, la fonction d'activation de la couche cachée est de type tangente hyperbolique et celle pour les neurones de sortie est linéaire ou purelin. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.17.

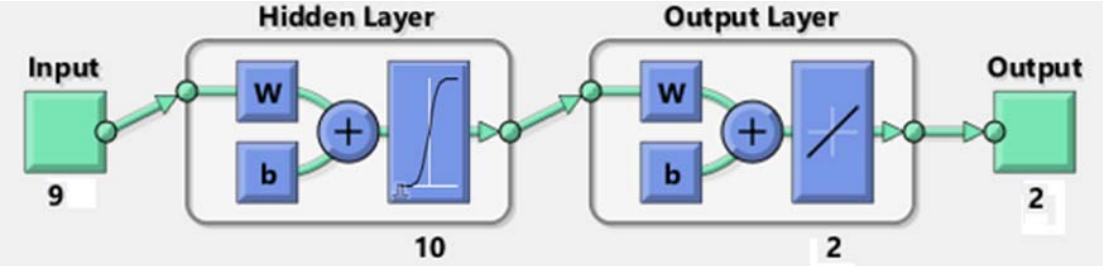
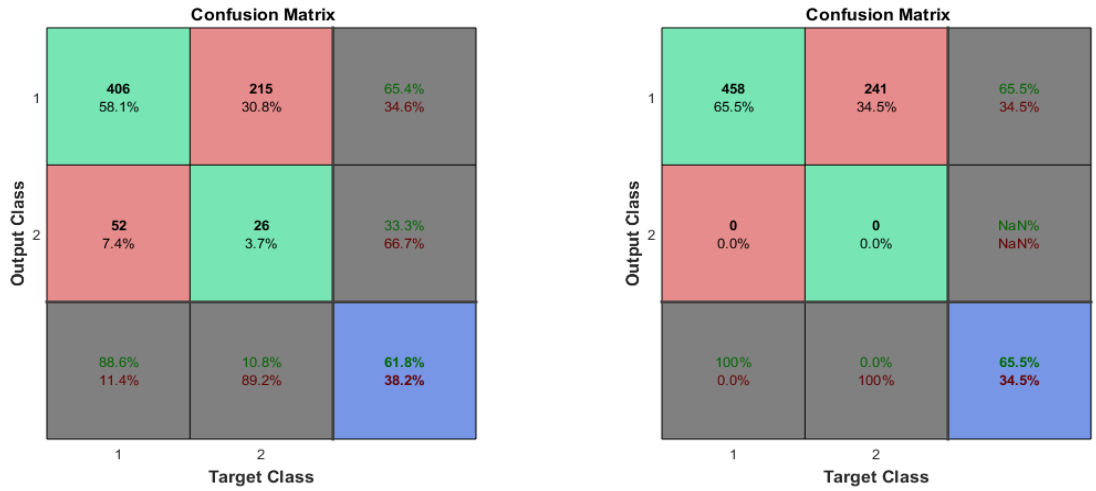


Figure 3.16 : Fonction d'activation tangente hyperbolique pour la couche cachée et purelin pour celle de sortie



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.17 : Matrice de confusion du RNA avec biais et une tangente hyperbolique comme fonction d'activation pour la couche cachée et purelin pour les neurones de sortie

Avec une erreur totale toujours supérieure à 34% (38,2% dans le premier cas et 34,5% dans le second, mais avec aucune tumeur maligne détectée pour ce dernier), le résultat est toujours aussi catastrophique. Ces résultats confirment l'hypothèse précédente, à savoir qu'une fonction d'activation linéaire (purelin) ne convient pas à notre problème.

3.2. Architecture sans biais

3.2.1. Fonction d'activation linéaire

La figure 3.18 décrit le RNA, basé sur une fonction de transfert avec biais (cf. rel 3.2) et une fonction d'activation linéaire appelée purelin. Cette fonction d'activation est utilisée pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.19.

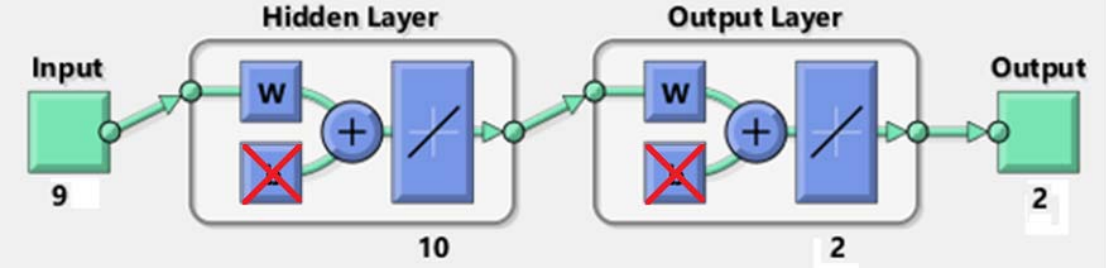
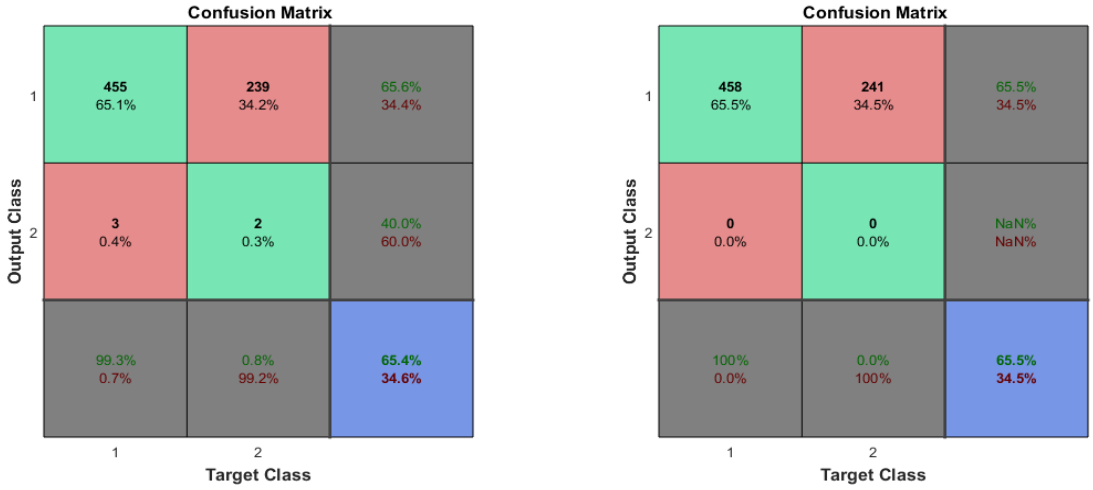


Figure 3.18 : Modèle Matlab du RNA sans biais et avec purelin comme fonction d'activation



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.1 : Matrice de confusion du RNA sans biais et avec purelin comme fonction d'activation

Si on considère le résultat final, on peut constater que l'erreur totale est de 34,6% pour le premier algorithme, valeur très supérieure à notre valeur de référence de 2,4%. Quant au second algorithme, même si le taux d'erreur est sensiblement le même, il n'a détecté aucune des tumeurs malignes, ce qui est rédhibitoire.

Dans les deux cas, ce résultat n'est pas satisfaisant et ce RNA ne sera pas retenu comme architecture finale.

3.2.2. Fonction d'activation de type tangente hyperbolique

La figure 3.20 décrit le second RNA étudié avec biais (cf. rel 3.1), mais cette fois la fonction d'activation est une tangente hyperbolique. Elle est utilisée à la fois pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.21.

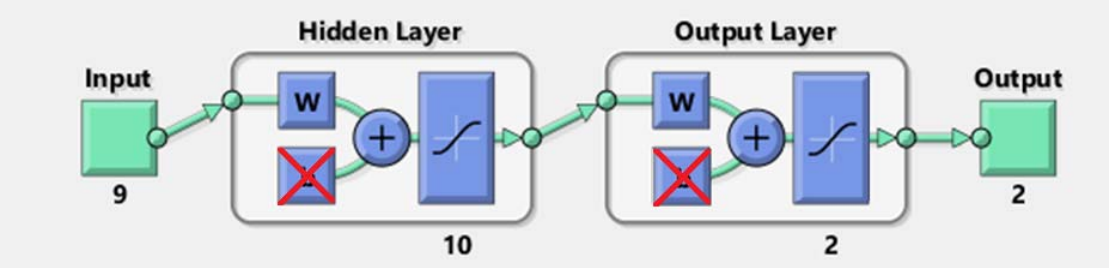
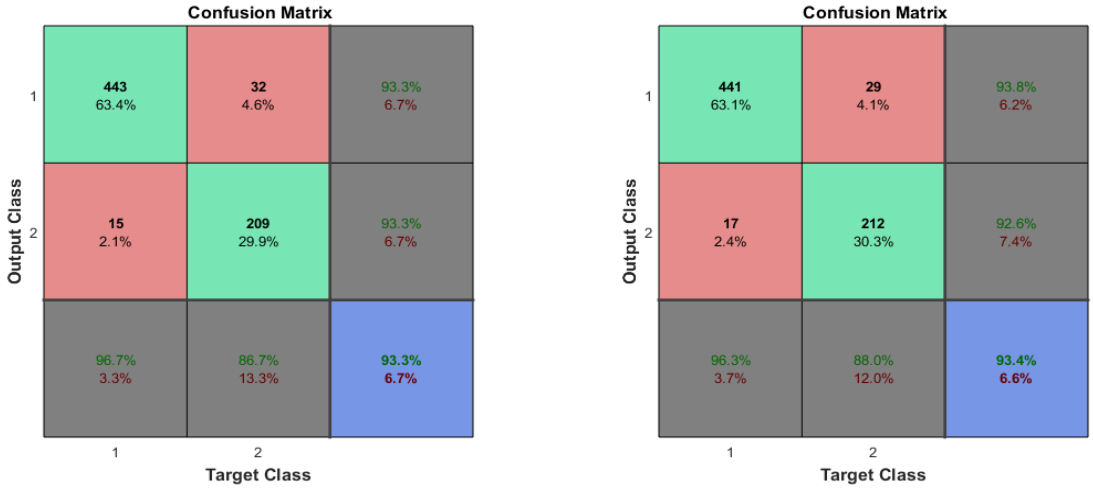


Figure 3.20 : Modèle Matlab du RNA sans biais et avec une tangente hyperbolique comme fonction d'activation



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.21 : Matrice de confusion du RNA sans biais et avec une tangente hyperbolique comme fonction d'activation

Comme dans le cas du RNA avec biais, on constate une très forte diminution de l'erreur totale par rapport au cas précédent. En outre, les deux algorithmes sont relativement équivalents ; ils fournissent des résultats similaires. Toutefois, cette erreur totale, qui n'est plus que de 6,7% ou 6,6%, reste très supérieure à notre valeur de référence de 2,4%. Cette solution ne sera pas retenue.

3.2.3. Fonction d'activation de type logsigmoid

Pour ce troisième cas, le RNA étudié (cf. Figure 3.22) est là encore basé sur l'algorithme avec biais donné par l'équation (3.1). Toutefois, la fonction d'activation est maintenant une fonction de type logsigmoid, qui est utilisée pour tous les neurones cachés et ceux de sortie. Les résultats de la classification sont résumés par la matrice de confusion présentée sur la figure 3.23.

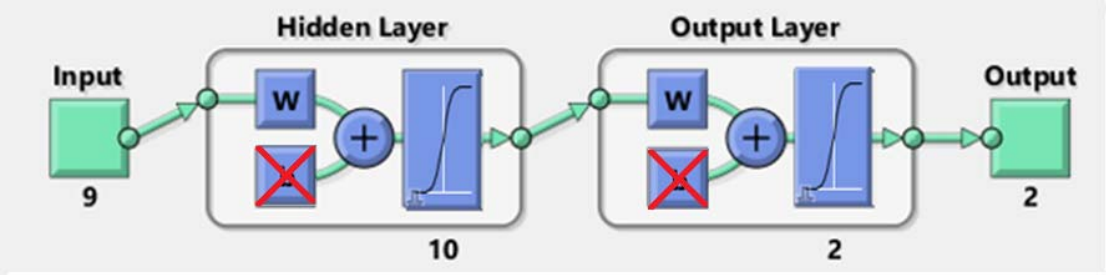
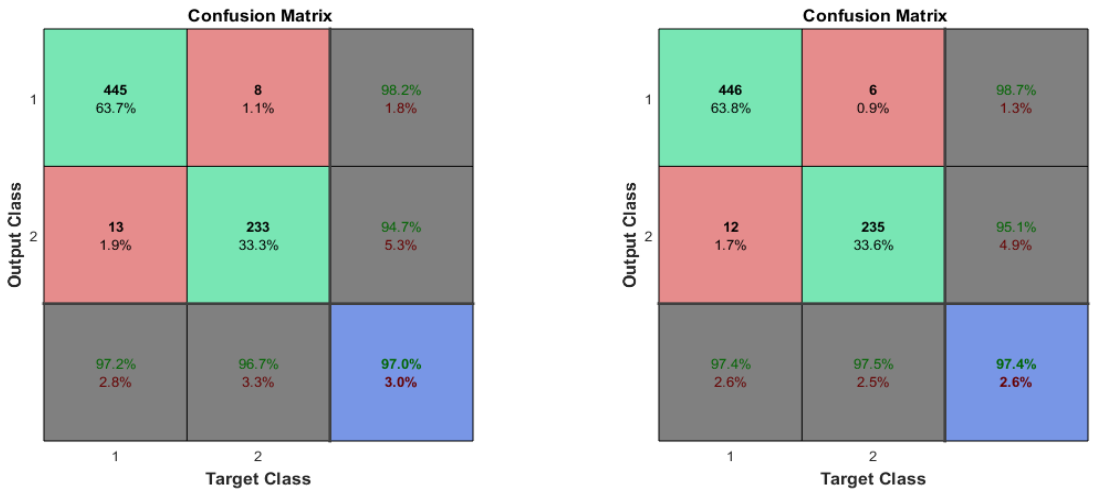


Figure 3.22 : Modèle Matlab du RNA sans biais et avec une fonction d'activation logsigmoid



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

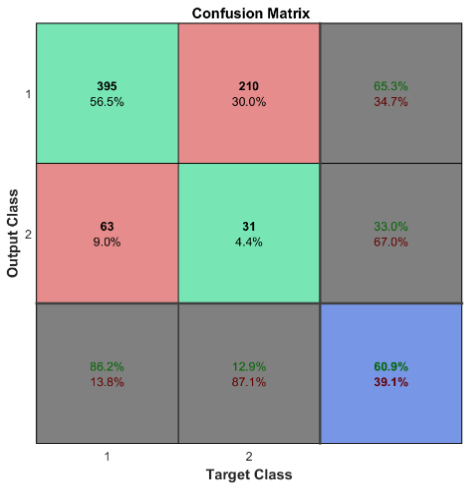
Figure 3.23 : Matrice de confusion du RNA sans biais et avec une fonction d'activation logsigmoid

Cette fois encore, le résultat final (case bleue) montre que l'erreur totale est toujours inférieure à 3% quel que soit l'algorithme d'apprentissage (3,0% pour le premier et 2,6% pour le second). Cette erreur totale est relativement proche de notre référence de 2,4%. Ce résultat est même légèrement meilleur que l'architecture avec biais, ce qui simplifiera l'électronique finale. Ce RNA sera, sans doute, retenu comme architecture finale.

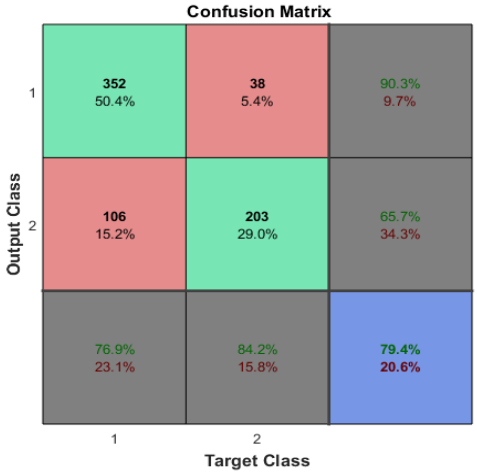
3.2.4. Fonction linéaire en sortie

Comme dans le cas du RNA avec biais, quatre dernières simulations ont été réalisées. Les deux premières utilisent une fonction d'activation logsigmoid pour les neurones de la couche cachée et une fonction linéaire pour ceux de sortie. Le résultat est présenté sur la figure 3.24. Les secondes simulations utilisent une fonction d'activation de type tangente hyperbolique pour la couche cachée et toujours une fonction linéaire pour ceux de sortie. Le résultat est présenté sur la figure 3.25.

Dans tous les cas, les résultats sont rédhibitoires, l'erreur totale est toujours supérieure à 20%. Que ce soit avec ou sans biais, la fonction linéaire n'est pas adaptée à notre problématique de classification de cellules cancéreuses.

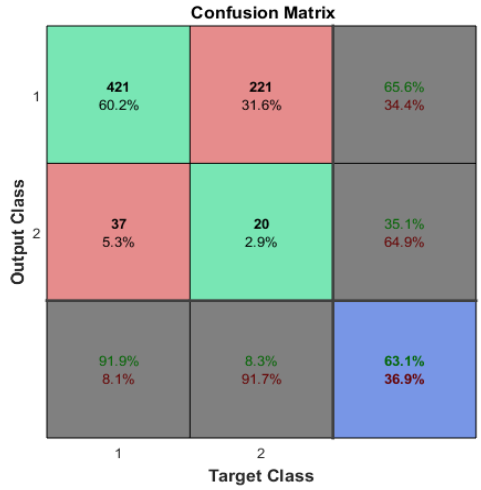


a) Algorithme 1 (E.D.=0.1)

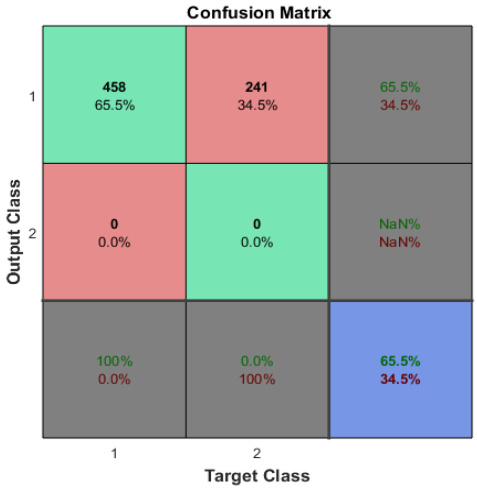


b) Algorithme 2 (Nb.E.=14)

Figure 3.24 : Matrice de confusion du RNA sans biais et avec une d'activation logsigmoid pour la couche cachée et purelin pour les neurones de sortie



a) Algorithme 1 (E.D.=0.1)



b) Algorithme 2 (Nb.E.=14)

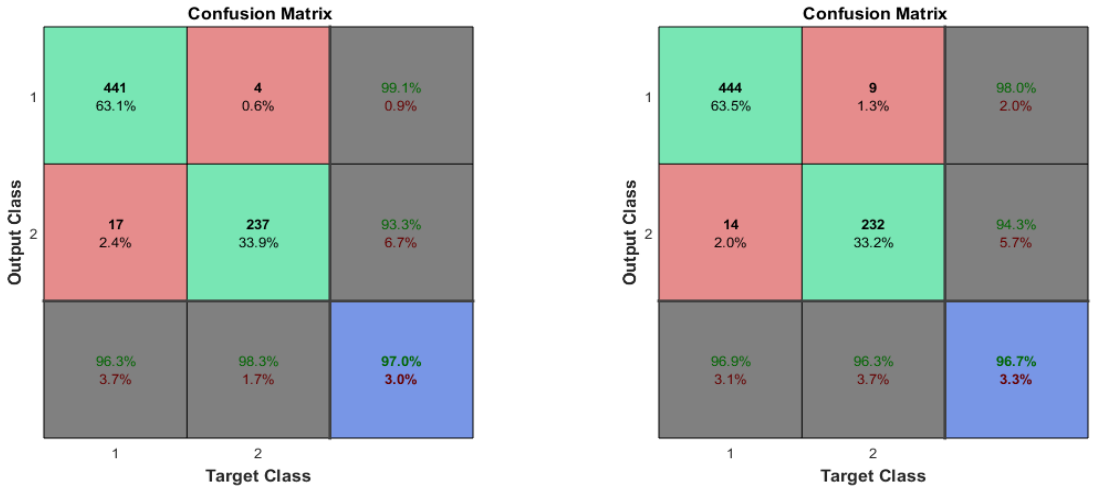
Figure 3.25 : Matrice de confusion du RNA sans biais et avec une tangente hyperbolique comme fonction d'activation pour la couche cachée et purelin pour les neurones de sortie

3.3. Validation du choix final

L'ensemble des résultats précédents montre que la meilleure architecture correspond au RNA sans biais avec une fonction d'activation de type logsigmoid pour les neurones de la couche cachée et celle de sortie. En outre, le choix de l'algorithme de la phase d'apprentissage n'a qu'une légère influence (3% pour le premier où l'erreur visée a été fixé à 0,1 et 2,6% pour le second où le nombre d'époques est fixé à 14).

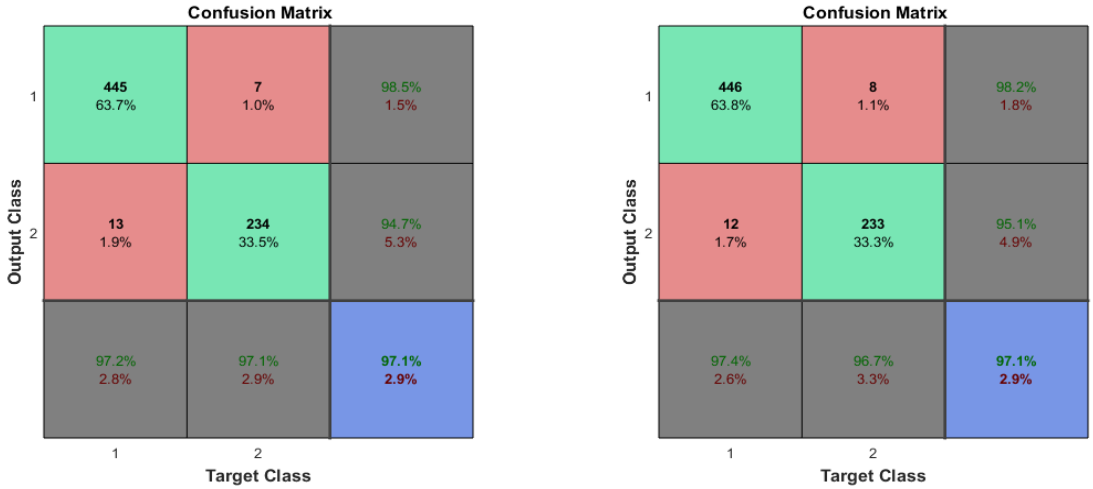
Pour entériner ce choix, une dernière série de simulation a été réalisée. Que ce soit avec ou sans biais, les fonctions d'activation « tangente hyperbolique » et « logsigmoid » ont été croisées pour la couche cachée et les neurones de sortie.

Quatre différentes configurations ont ainsi été testées, dont les différents résultats de simulation sont résumés sur les figures 3.26 à 3.29.



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.26 : Matrice de confusion du RNA avec biais, une fonction d'activation logsigmoid pour la couche cachée et une tangente hyperbolique comme fonction d'activation pour les neurones de sortie

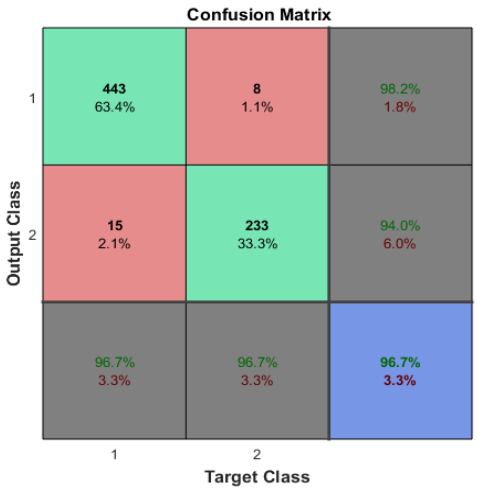


a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.27 : Matrice de confusion du RNA avec biais, une tangente hyperbolique comme fonction d'activation pour la couche cachée et une fonction d'activation logsigmoid pour les neurones de sortie

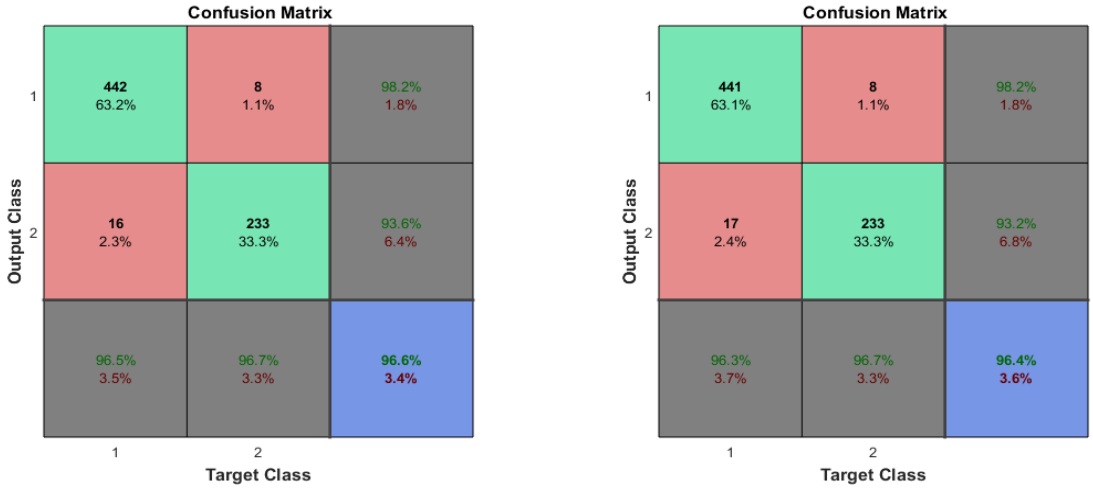
Pas de convergence

No convergence waiting for long time



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.28 : Matrice de confusion du RNA sans biais, avec une fonction d'activation logsigmoid pour la couche cachée et une tangente hyperbolique comme fonction d'activation pour les neurones de sortie



a) Algorithme 1 (E.D.=0.1) b) Algorithme 2 (Nb.E.=14)

Figure 3.29 : Matrice de confusion du RNA sans biais, avec une tangente hyperbolique comme fonction d'activation pour la couche cachée et une fonction d'activation logsigmoid pour les neurones de sortie

Bien que dans un cas, le simulateur n'ait pas convergé, il semble inutile de poursuivre plus en détail les analyses. En effet, dans tous les autres cas les résultats montrent une erreur supérieure à celle obtenue au paragraphe 3.2.3 (RNA sans biais avec une fonction d'activation logsigmoid). Concernant les résultats de la figure 3.27, ils sont quasi-identiques à ceux du paragraphe 3.2.3. Toutefois, l'architecture du RNA sera plus compliquée puisqu'il faut rajouter un biais donc un sommateur par neurone de la couche cachée et ceux de la couche de sortie, augmentant ainsi la surface et la consommation du circuit final. En outre, il est plus simple de concevoir et dupliquer simplement la même fonction d'activation pour tous les neurones.

3.4. Conclusion

L'analyse des résultats de ce paragraphe permet de retenir le réseau de neurones artificiel sans biais pour les fonctions de transfert, et une fonction d'activation logsimoid pour tous les neurones de la couche cachée et celle de sortie. Le nombre de neurones d'entrée est de 9, correspondant aux 9 attributs de la base de données Wisconsin, et celui de la couche de sortie et de 2 (2 classes bégnine ou maligne). Les simulations Matlab nous ont également permis d'optimiser le nombre de neurones de la couche cachée, à savoir 10, ainsi que certains autres paramètres. Le choix de l'algorithme d'apprentissage retenu est celui qui fixe le nombre d'époques, dont la valeur optimale est de 14, offrant ainsi une erreur totale de 2,6%, très proche de notre référence de 2,4%. Enfin le taux d'apprentissage est de 0,25.

Les simulations Matlab nous ont, par ailleurs, permis de déterminer dans chaque cas la valeur des différents poids et biais de chaque neurone, en fin de phase d'apprentissage. Considérant le choix retenu, un RNA sans biais avec une fonction de transfert logsigmoid pour tous les neurones, nous obtenons comme valeurs extrêmes de poids [-1,37 ; 1,17] pour la couche cachée et [-3,54 ; 3,48] pour celle de sortie, dans le cas du second algorithme (nombre d'époques = 14). Notons que dans le cas du premier algorithme, ces valeurs de poids sont respectivement [-7,09 ; 5,04] et [-10,83 ; 10,78], valeurs également plus défavorables car nécessitant une dynamique plus élevée. L'ensemble des valeurs des poids et biais de chaque configuration étudiée est donné en Annexe 1.

Une fois, l'architecture du RNA choisie, l'étape suivante consiste à proposer une topologie pour réaliser ce circuit, puis définir le cahier des charges de chaque bloc de base. Ce travail est présenté au paragraphe suivant.

4. Dimensionnement des blocs de base

4.1. Introduction

Les simulations Matlab précédentes nous ont permis de définir la structure d'un neurone du RNA final qui devra réaliser la classification des cellules cancéreuses. La figure 3.30 décrit les différents blocs de base de ce neurone :

- Mémorisation des poids, initialisé par un nombre aléatoire
- Entrée du neurone :
- Multiplieur pour effectuer le produit de deux valeurs précédentes

- Sommateur
- Fonction d'activation Logsigmoid

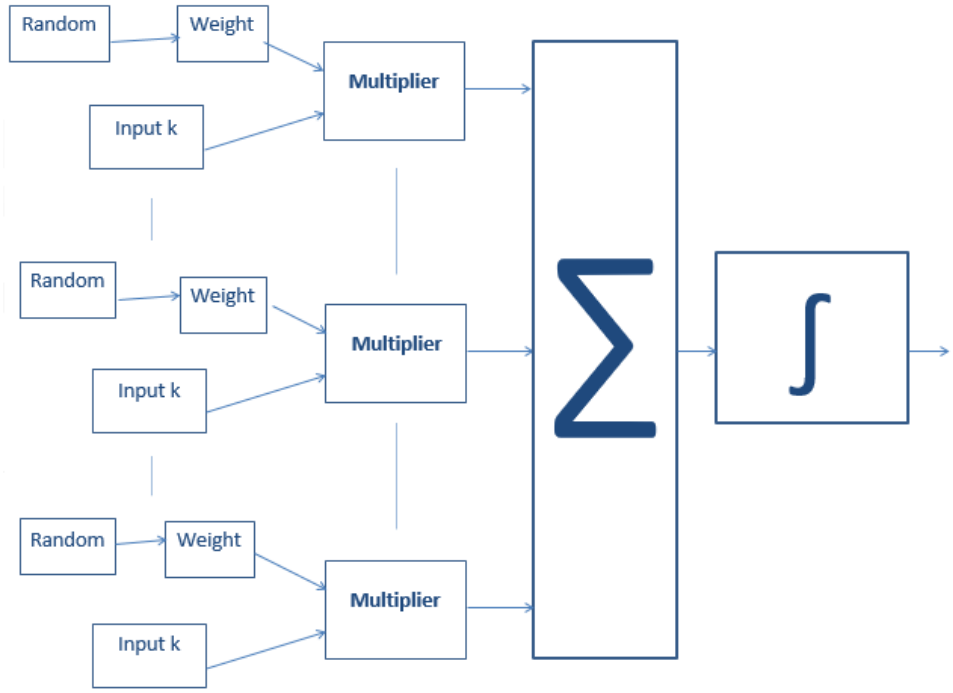


Figure 3.30 : Structure d'un neurone

L'objectif est maintenant de déterminer, à partir de simulations Matlab, les spécifications ou cahier des charges de chaque bloc. Pour réaliser cette opération, nous allons remplacer dans les modèles Matlab, les fonctions idéales par des « circuits » qui vont prendre en compte certaines de leurs limitations ou non-idéalités afin d'en calculer l'impact sur la qualité du système de classification final. Dans une seconde étape nous affinerons ce modèle, lorsque les circuits seront implémentés au niveau transistor (au chapitre suivant) et que les simulations Spice nous permettront d'extraire la valeur de ces paramètres (performances « réelles » du circuit). Une dernière simulation haut-niveau, sous Matlab, sera finalement réalisée pour valider l'ensemble du circuit final.

4.2. Implémentation Matlab

Rappelons que notre modèle de RNA, basé sur le réseau neuronal artificiel perceptron multicouche MLP à rétro-propagation [8], a été décrit sous Simulink/Matlab [17] en utilisant des fonctions mathématiques idéales. Il convient maintenant de les remplacer par des blocs non-idéaux pour le multiplieur, la fonction d'activation et ses blocs dérivés, afin de connaître l'influence de ces paramètres sur le résultat de la classification du réseau de neurones.

L'architecture du réseau de neurones sans biais et le diagramme correspondant du réseau neuronal de feed-forward, créé sous Simulink en utilisant le logiciel Matlab R2016a, sont représentés sur les figure 3.31 et 3.32.

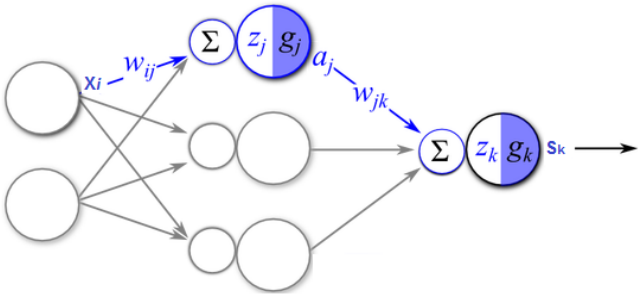


Figure 3.31 : Architecture du réseau de neurones sans biais

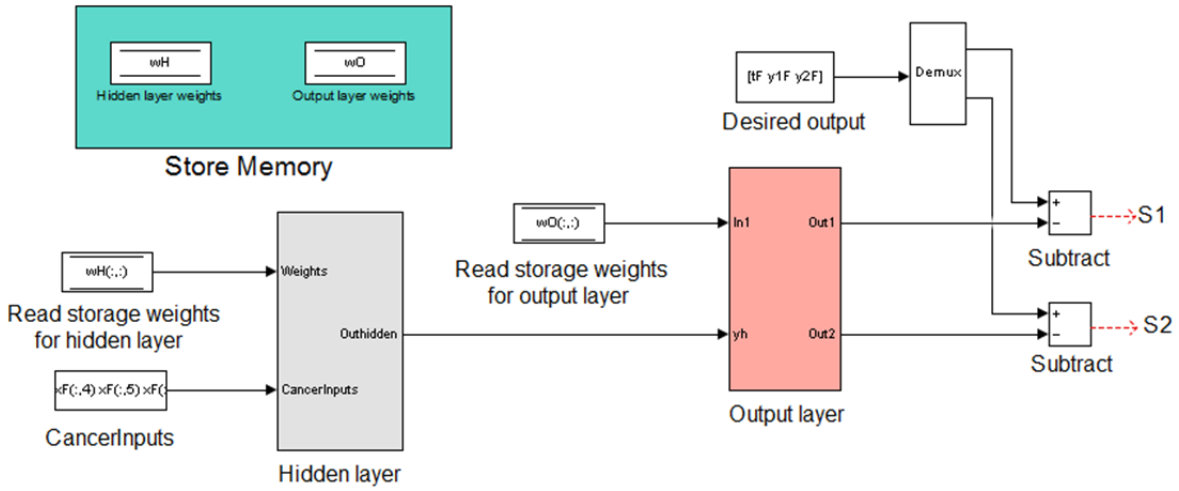


Figure 3.32 : Modèle Matlab du réseau neuronal d'anticipation

Les registres wH et wO sont utilisés pour stocker respectivement les poids pour la couche cachée et celle de sortie en utilisant le bloc de mémoire de stockage de données. Ces blocs sont initialisés en utilisant la fonction aléatoire qui crée des valeurs comprises entre -0,9 et 0,9, au lieu de -1 et 1 précédemment, pour être utilisées comme entrées par le bloc multiplieur. En effet, ce dernier sera créé par un circuit CMOS analogique en utilisant la technologie HCMOS9A 130nm avec une tension d'alimentation de ±0,9V [18].

Nos valeurs initiales, pour la détection et la classification du cancer du sein, le nombre d'entrées, le nombre de neurones dans la couche cachée, le nombre d'échantillons de biopsies, le nombre d'itérations et le nombre d'époques se trouvent dans la boîte à outils du réseau neuronal et n'ont bien évidemment pas été modifiées (cf. Tableau 3.1). Les blocs de la couche cachée et de celle de sortie sont régis par les équations (rel 3.3) et (rel 3.4).

$$a_j = g_j(\sum_{i=1}^{9} w_{ij} * x_i \ ) \quad \text{j=1, 2…10} \qquad \text{(rel 3.3)}$$

$$s_k = g_k(\sum_{j=1}^{10} w_{jk} * a_j \ ) \ \ \text{k=1, 2} \qquad \text{(rel 3.4)}$$

La sortie désirée est équivalente aux cibles de cancer de notre application [16], identiques aux valeurs précédentes (paragraphes 2 et 3 de ce chapitre). La connectivité entre couches est réalisée avec le routage Simulink [19]. Les détails du bloc "Couche cachée" de la figure 3.32 sont présentés sur la figure 3.33.

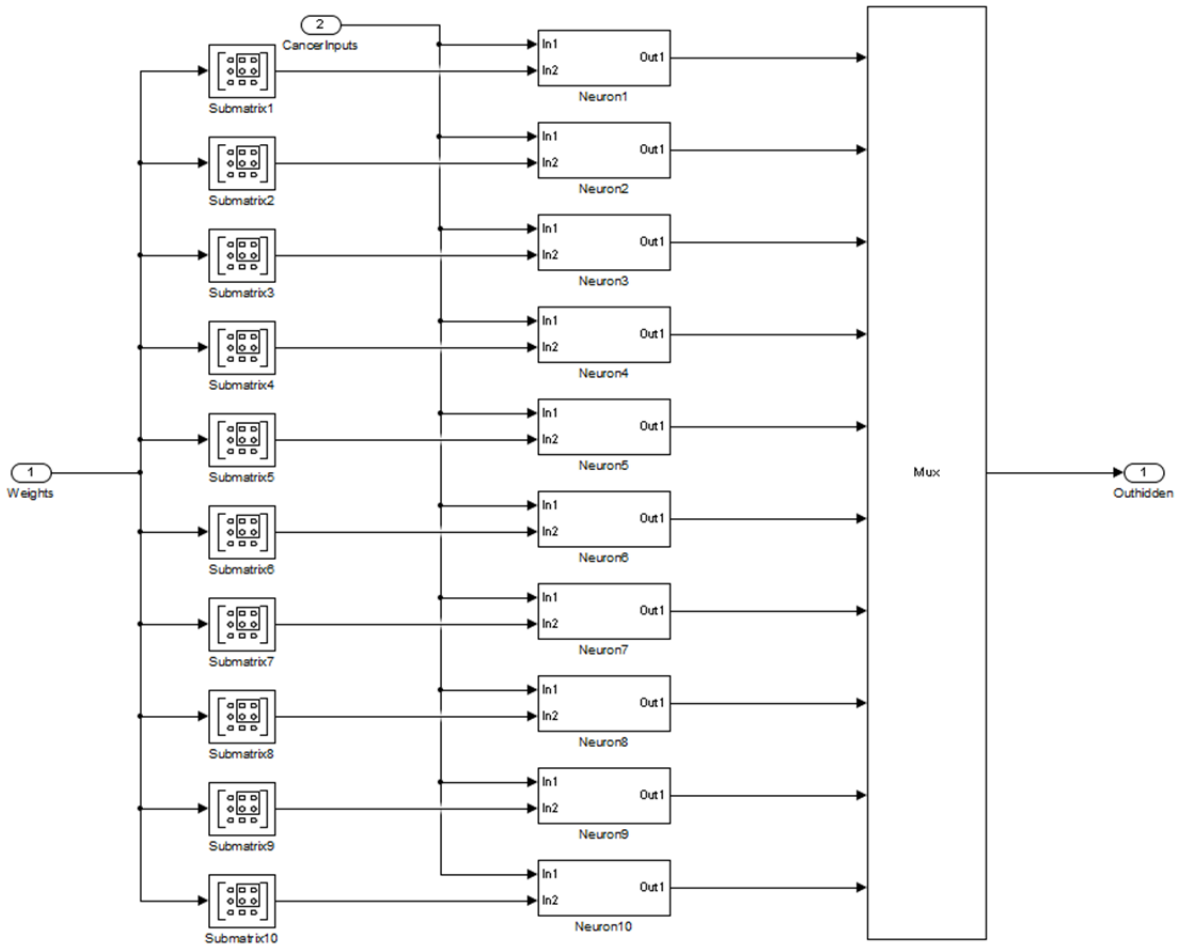


Figure 3.33 : Détail de la couche cachée

Nous choisissons à travers le bloc sous-matrice les poids correspondants de chaque neurone, appelé neurone Singleton. La figure 3.34 détaille un de ces neurones, régi par l'équation (rel 3.5) :

$$a_j = Logsig(\sum_{i=1}^{9} w_{ij} * x_i \ ) \qquad \text{(rel 3.5)}$$

Sur la figure 3.34, le bloc principal, qui réalise la somme de chacun des produits (cf. rel 3.6), est détaillé sur la figure 3.35. Elle visualise les entrées et poids de chaque multiplieur ; il y a 9 entrées et 9 poids en raison des 9 attributs dans la base de données Wisconsin.

$$Out1 = \sum_{i=1}^{9} w_{ij} * x_i \qquad \text{(rel 3.6)}$$

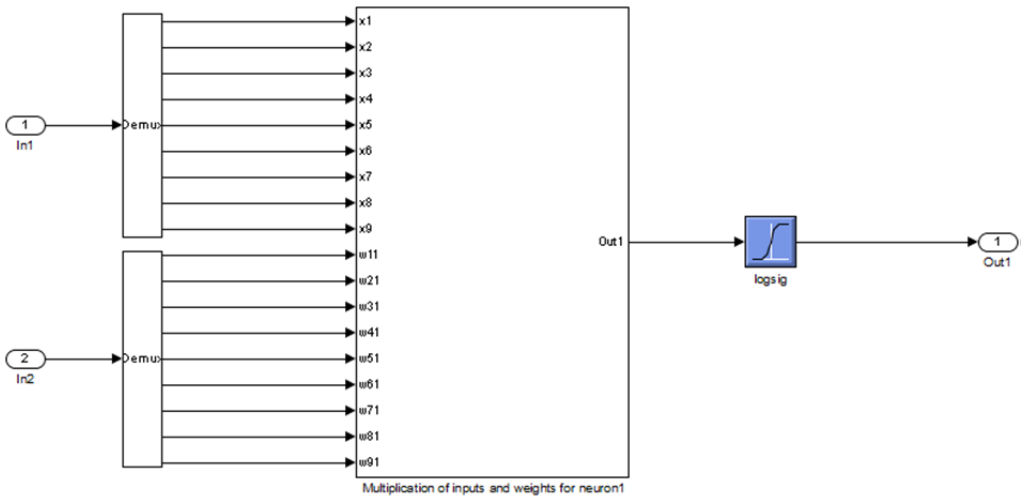


Figure 3.34 : Neurone Singleton

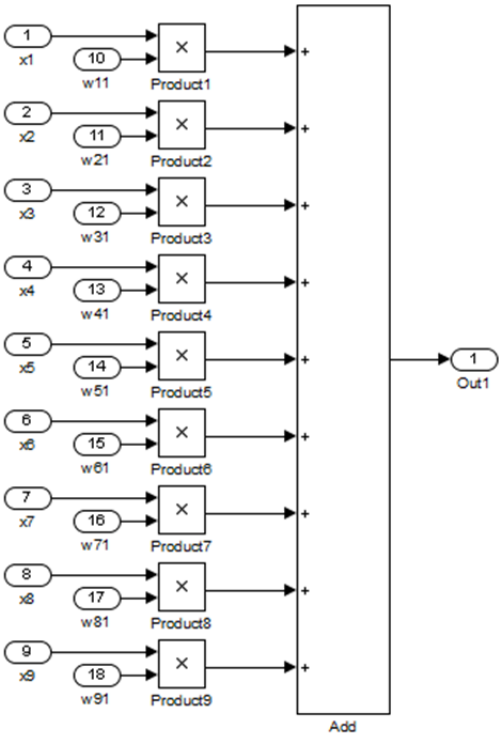


Figure 3.35 : Bloc principal du neurone Singleton réalisant la somme des produits des entrées et des poids

Sur la couche de sortie de la figure 3.32, les entrées Cancer et les blocs de sortie désirée représentent respectivement les 9 attributs des 699 biopsies et une matrice de 2*699 où chaque colonne indique une catégorie correcte (bénigne ou maligne) avec une valeur 1 dans l'un des deux éléments 1 ou 2 suivant le résultat final [20]. Ainsi, le bloc Demux est utilisé pour séparer la sortie 1 ou 2 en fonction de la valeur désirée. Enfin les blocs de soustraction permettent de calculer l'erreur entre la sortie désirée et la sortie réelle du réseau neuronal.

Conformément à l'architecture du RNA MLP BP, nous avons implémenté la propagation arrière. Cette opération est réalisée par l'ajout de blocs de rétro-propagation à ceux d'anticipation pour obtenir la conception finale illustrée par la figure 3.36.

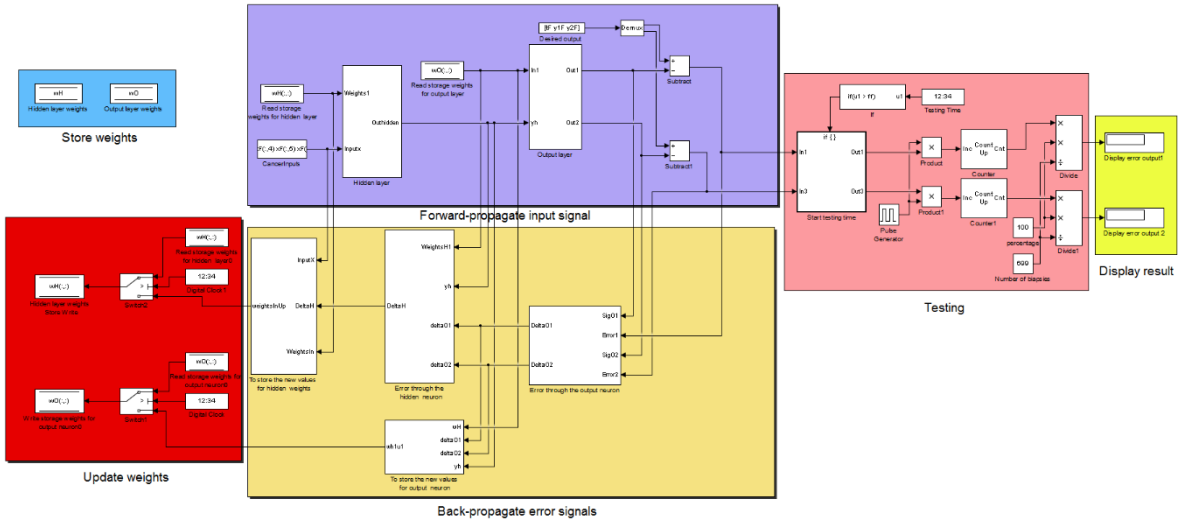


Figure 3.36 : Modèle Matlab/Simulink du RNA MLP BP

Sur cette figure, le bloc "Signaux d'erreur de propagation inverse" réalise la structure illustrée par la figure 3.37. A partir de l'erreur se propageant à travers le bloc "Signal d'erreur de propagation inverse O", les gradients locaux de la couche de sortie sont calculés en utilisant l'équation (rel 3.7).

$$\delta_k = g'_k(z_k)E(d_k, S_k) \quad k = 1, 2 \qquad \text{(rel 3.7)}$$

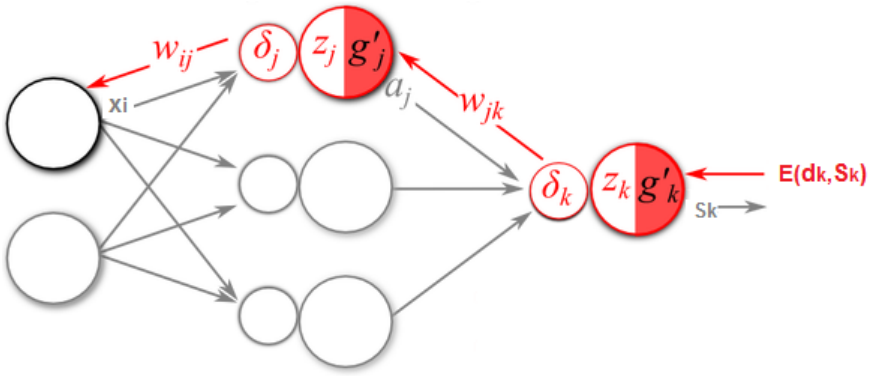


Figure 3.37 : Rétro-propagation des signaux d'erreur et mise à jour des nouvelles valeurs de poids

Ensuite, le bloc "Stocker les nouvelles valeurs pour le neurone de sortie" permet de mettre à jour les nouvelles valeurs pour les poids de couche de sortie (cf. rel 3.8).

$$w_{jk} = w_{jk} + \eta a_j \delta_k \qquad \text{(rel 3.8)}$$

De même pour la couche cachée avec les blocs "Propagation Inverse du signal d'erreur H" et "Pour stocker les nouvelles valeurs pour les poids cachés", le gradient local est déterminé à l'aide de la relation 3.9, alors que les nouvelles valeurs pour les poids dans la couche cachée sont données par l'équation 3.10 (cf. Figure 3.37).

$$\delta_j = g_j'(z_j) \sum_{k=1}^{2} \delta_k w_{jk} \quad \text{j=1,2... 10} \qquad \text{(rel 3.9)}$$

$$w_{ij} = w_{ij} + \eta a_i \delta_j \qquad \text{(rel 3.10)}$$

Le modèle complet, présenté à la figure 3.36, reste à ce niveau « idéal » à l'exception des poids, qui seront dorénavant compris entre -0,9 et +0,9. Il convient de noter, qu'à cette exception près, ce modèle a servi pour les simulations des paragraphes précédents (RNA sans biais). Avant de prendre en compte quelques non-idéalités des blocs de base, donnons quelques exemples de signaux issus des simulations Matlab.

4.3. Résultats de simulation Matlab

Les différents paramètres de simulation sont résumés dans le tableau 3.1. Pour la phase d'apprentissage, le nombre d'échantillons, le nombre d'époques et le taux d'apprentissage sont respectivement de 489, 14 et 0,25 [21]. Les simulations nous ont donné des temps d'apprentissage et de test égaux à 6845 µs et 7545 µs (soit 6845 + 699) respectivement.

Pour illustrer ces différentes simulations, différents chronogrammes sont présentés sur les figures 3.38 à 3.42. Pour chacune, l'axe des abscisses est gradué en µs (jusqu'à 7545µs, soit la phase d'apprentissage (6945µs) suivie de celle de test (699µs)).

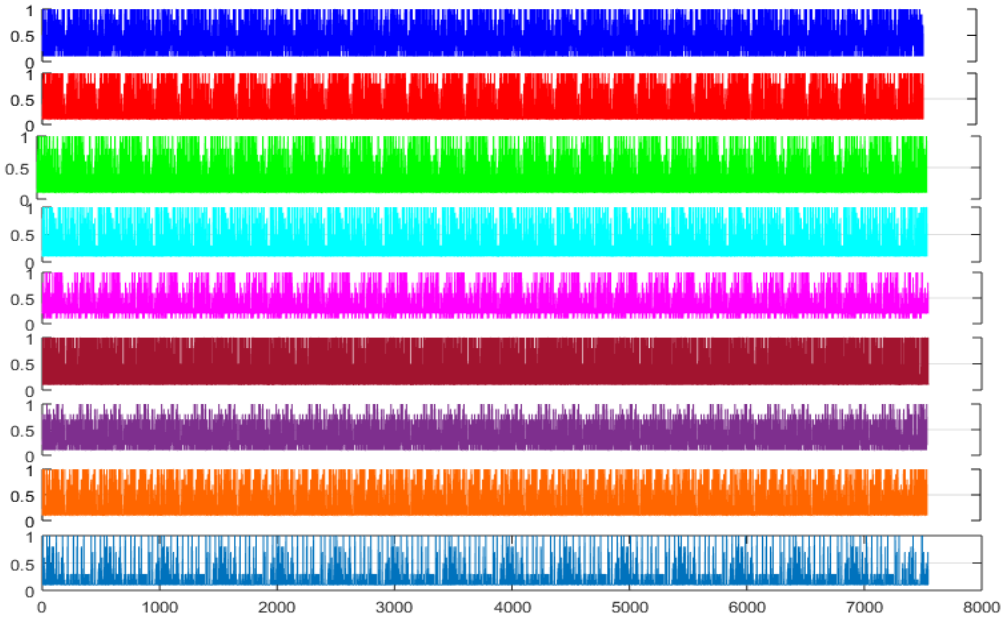


Figure 3.38 : 9 signaux d'entrée ou attributs des biopsies

Ces figures sont certes difficiles à interpréter en l'état, mais servent uniquement d'illustration. Ainsi la figure 3.38 présente les 9 attributs des 699 biopsies, valeurs d'entrée Cancer du réseau de neurone, alors que les figures 3.39 et 3.40 illustrent l'évolution des sorties pour la classe bénigne et maligne (sortie 1 et 2) respectivement en comparaison des sorties désirées ou cibles.

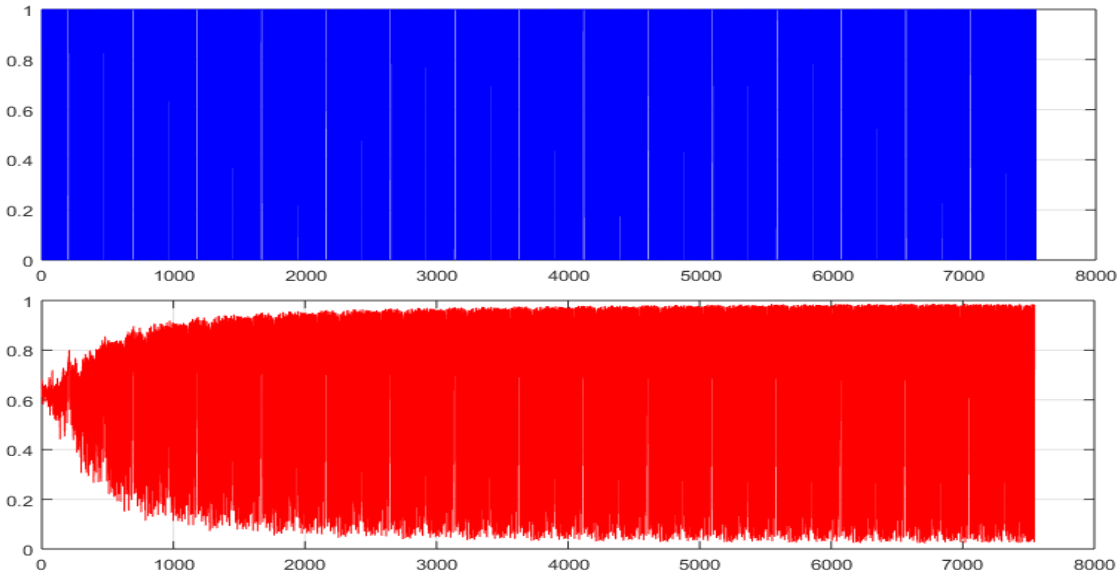


Figure 3.39 : Signal de sortie 1 (Classe bénigne) du RNA avec au-dessus la sortie désirée

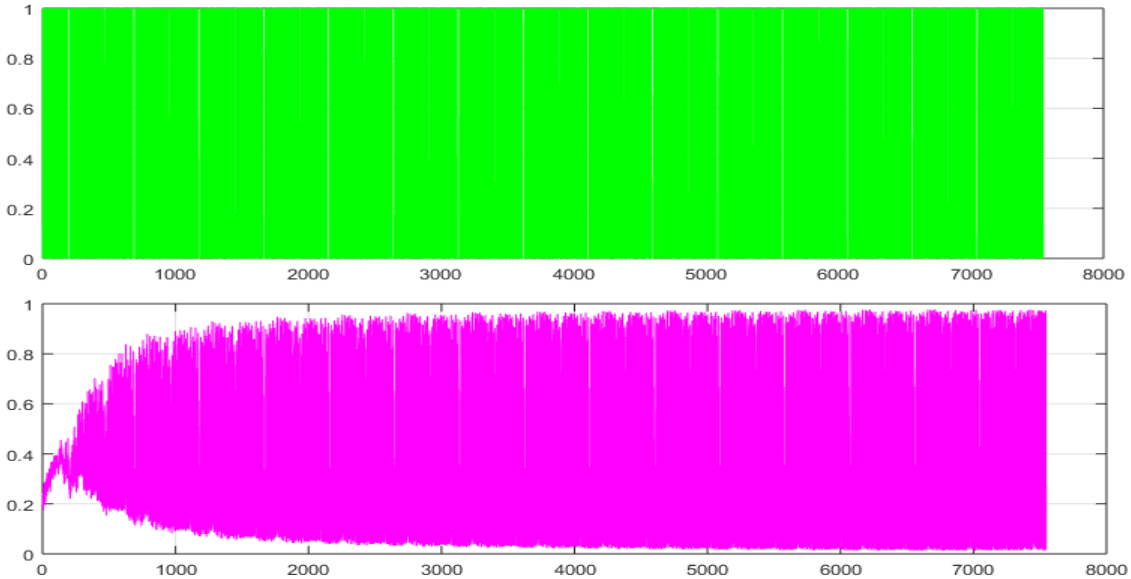


Figure 3.40 : Signal de sortie 2 (Classe maligne) du RNA avec au-dessus la sortie désirée

Ces deux figures montrent l'intérêt de rajouter le bloc de comparaison entre la sortie réelle et la sortie désirée afin de calculer le taux d'erreur du RNA. Elles sont, en effet, difficile à interpréter directement en l'état (voire quasi-illisible).

Beaucoup plus intéressante, la figure 3.41 présente, quant à elle, les variations des poids dans la couche couchée. Rappelons qu'ils sont au nombre de 90, la couche cachée est constituée de 10

neurones ayant chacun 9 entrées (9 attributs de chaque biopsie). Sur cette figure, on peut constater qu'au bout de 6845µs (phase d'apprentissage), les poids ne varient plus. Notons que dans ce cas, les valeurs extrêmes sont données par l'intervalle [-1,37 ; +1,17] (cf. paragraphe 3.4 de ce chapitre). Les valeurs initiales de ces poids ont été tirées aléatoirement entre -0,9 et +0,9, à t=0.

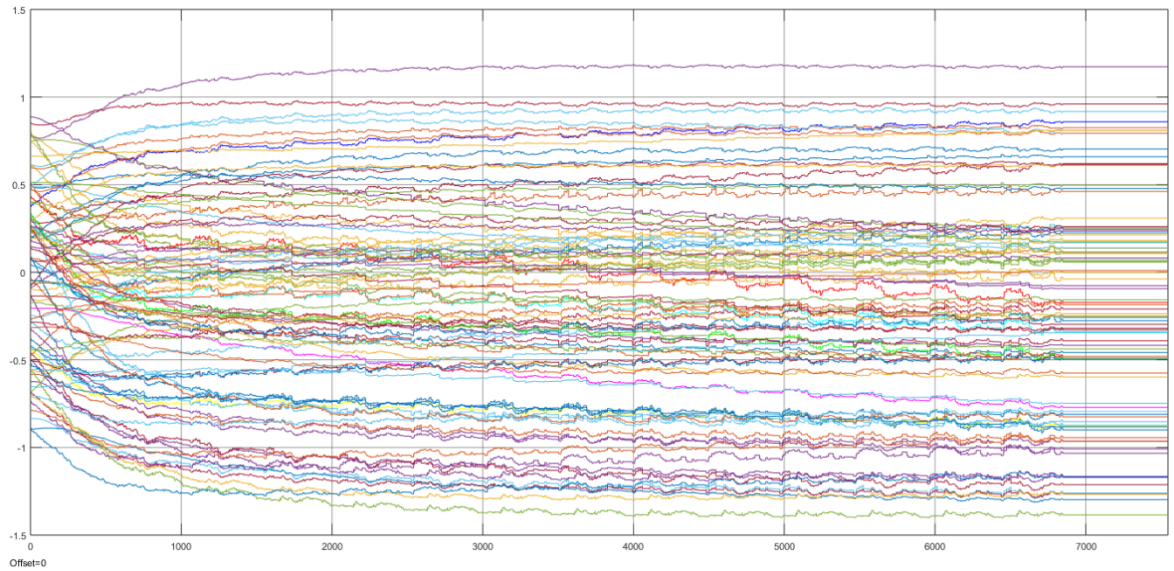


Figure 3.41 : Variations des poids de la couche cachée

Enfin, grâce à la structure de comparaison, rappelée sur la figure 3.42, le nombre d'erreur et donc son pourcentage peuvent être calculés et offrir un résultat plus facile à interpréter. Ainsi, la figure 3.43 présente la comparaison entre la sortie désirée (en haut) et celle obtenue par le réseau de neurones. Sur cette figure, un 1 signifie la présence d'une cellule correspondante à sa classe d'origine (bénigne pour la sortie 1 et maligne pour la sortie 2). Pour des raisons de lisibilité, nous n'avons imprimé qu'une partie de la phase de test (temps supérieur à 6845µs).

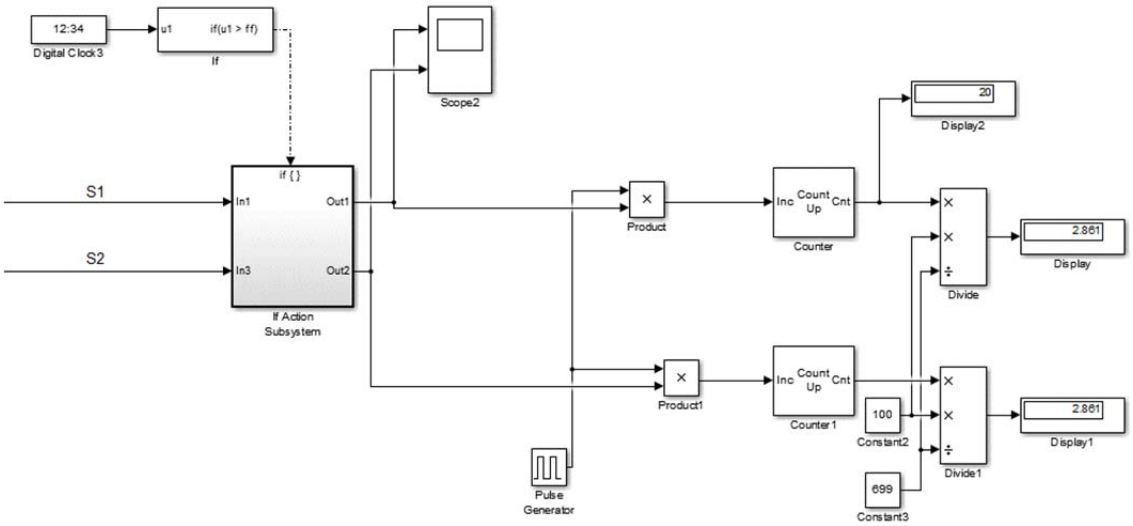


Figure 3.42 : Structure de calcul des bonnes ou mauvaises réponses du RNA

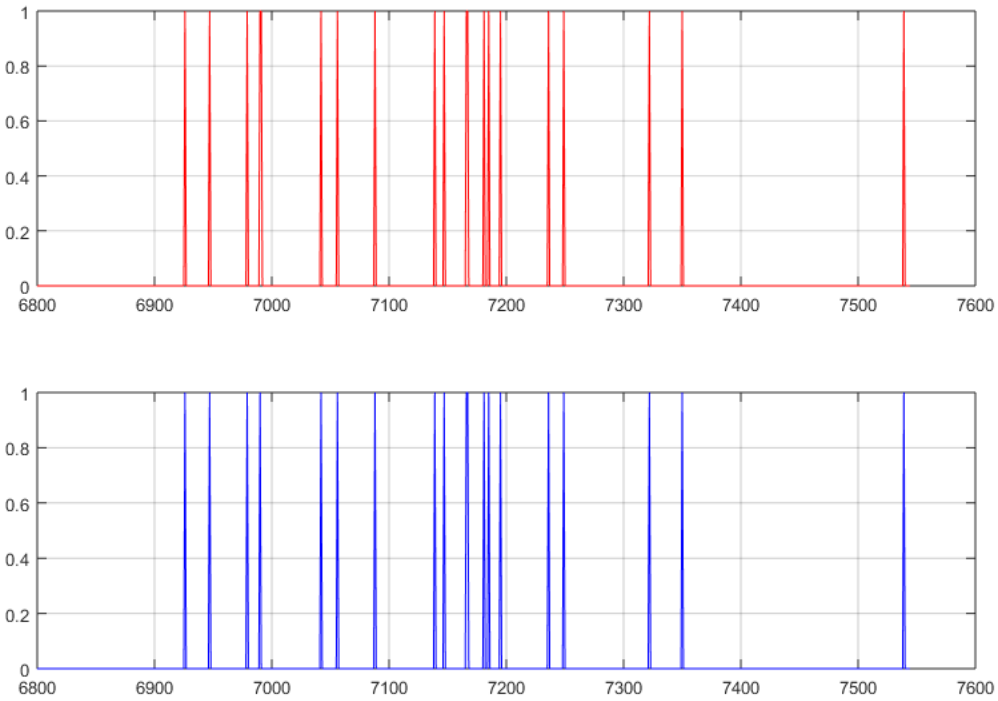


Figure 3.43 : Comparaison entre la sortie désirée et la sortie réelle pour une des deux classes

Bien que cette nouvelle figure soit plus facile à interpréter que les résultats illustrés par les figures 3.39 et 3.40, il est aisé de comprendre que les matrices de confusion, calculées à partir de ce type de résultats, est un outil beaucoup plus simple et visuel pour analyser la qualité d'un RNA.

L'ensemble des modèles et outils ayant été développés sous Matlab/Simulink, nous pouvons maintenant raffinés certains modèles, i.e. introduire des non-idéalités, pour déterminer les spécifications des blocs de base constitutifs du RNA. L'objectif est d'estimer les erreurs maximales admises sur ces blocs qui n'auront pas ou peu d'impact sur la qualité de la classification et donc offrir un taux d'erreur quasi- identique.

4.4. Création d'un bloc multiplieur non-idéal

Dans le cas du multiplieur deux paramètres sont particulièrement important, sa dynamique d'entrée et sa précision. Cette dernière doit prendre en compte à la fois le bruit ramené à l'entrée du multiplieur et le point mémoire pour le poids.

Pour réaliser cette étude, une erreur est ajoutée en entrée du multiplieur, initialement idéal sous Matlab. La figure 3.44 présente le synoptique de ce multiplieur non-idéal. Les simulations nous permettront d'une part de calculer la valeur maximale de ce « bruit » sans affecter la qualité du RNA et d'autre part de vérifier qu'une dynamique de ±0,9V est suffisante. Pour réaliser cette opération, un « bruit » a simplement été rajouté sur un second multiplieur idéal ; il est issu d'un générateur de

nombre aléatoire centré autour de 1. En fixant 0 comme valeur maximale de ce bruit, on retrouve un multiplieur idéal.

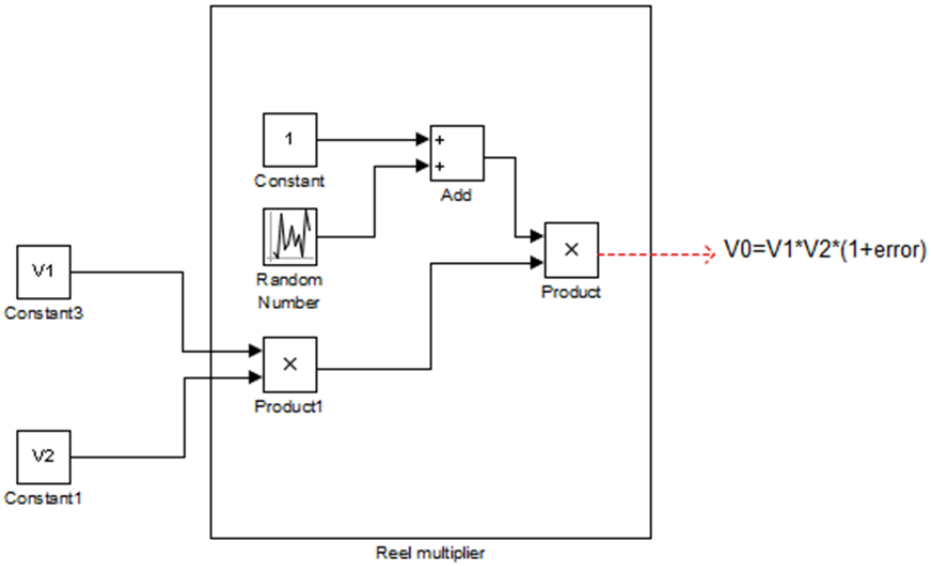


Figure 3.44 : Multiplieur non-idéal

Par exemple, si nous choisissons V1 = 3 et V2 = 2, le résultat sans bruit devrait être de 6 alors que la sortie du multiplieur est illustrée sur la figure 3.45.

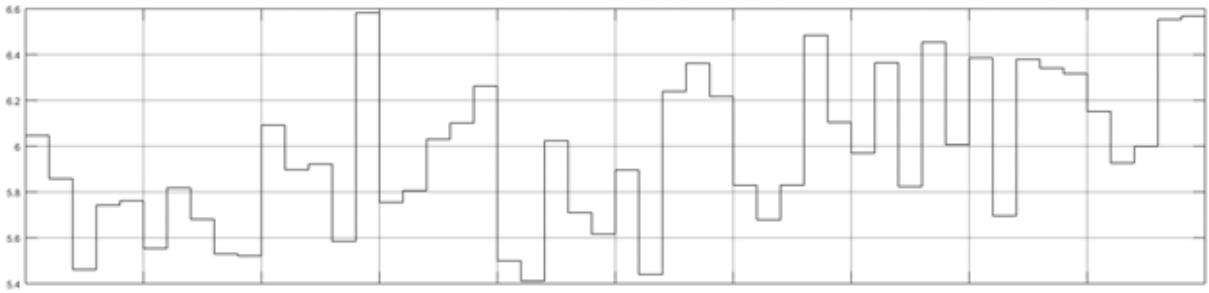


Figure 3.45 : Sortie du multiplieur « bruyant » pour une valeur idéale de 6

Le générateur de nombre aléatoire étant centré, on vérifie que la valeur moyenne de sortie est bien de 6. Par ailleurs, en remplaçant dans le modèle Matlab/Simulink du RNA tous les multiplieurs idéaux par ce nouveau circuit, nous modifions également la fonction dérivée par la même occasion. En effet, si f(x) correspond à la fonction d'activation logsigmoid, sa dérivée est donnée par la relation $f(x)*(1-f(x))$. Nous avons ainsi effectué une série de simulations où la valeur maximale du « bruit » augmente en valeur relative par rapport aux entrées du multiplieur. Les résultats sont résumés dans le tableau 3.2. La première ligne de ce tableau (er1=er2=0%) correspond au cas idéal dont les résultats ont été présentés au paragraphe 3.2.3 (RNA sans biais avec une fonction d'activation logsigmoid) de ce chapitre et sert de référence, à savoir 18 erreurs lors de la classification (soit 2,6%).

Tableau 3.2 : Résultats avec un multiplieur non-idéal

er1	er2	Nb Erreur	% Erreur	Poids extrêmes wH	Poids extrêmes wO
0%	0%	18	2,6%	-1,37 ➔1,17	-3,54 ➔3,48
-10%	10%	19	2,7%	-1,6 ➔1,4	-3,8 ➔3,5
-20%	20%	23	3,3%	-1,7 ➔1,2	-3,2 ➔3,5
-30%	30%	26	3,7%	-1,7 ➔1,2	-3 ➔3
-40%	40%	31	4,4%	-1,7 ➔1,5	-3 ➔3,2
-50%	50%	43	6,2%	-1,7 ➔1,2	-3 ➔2,8
-60%	60%	68	9,7%	-1,7 ➔1,3	-3 ➔2,8
-70%	70%	77	11%	-1,8 ➔1,6	-2,8 ➔3
-80%	80%	86	12,3%	-2,5➔1,8	-2,5 ➔3,1
-90%	90%	92	13,6%	-2,4 ➔1,8	-2,7 ➔2,5
-100%	100%	98	14%	-2,8 ➔1,8	-2,7 ➔2,5

Sur ce tableau, les variables er1 et er2 représentent les pourcentages de plage d'erreur entre deux valeurs ajoutées au bloc multiplieur. Les deux colonnes suivantes donnent le nombre total de mauvaise détection sur les 699 biopsies de la base de données Wisconsin sur laquelle nous avons travaillé. Ainsi, 18 erreurs au total (12 cas bénins et 6 malins, cf. paragraphe 3.2.3) donne un pourcentage de 18/699=2,6%. Enfin, les deux dernières colonnes représentent respectivement les valeurs extrêmes, obtenues en fin de phase d'apprentissage (Algorithme 2 où le nombre d'époques est fixé à 14), wH pour la couche cachée et wO pour la couche de sortie.

De ce résultat, nous pouvons déduire qu'une erreur inférieure à 10% sur les entrées des multiplieurs, et donc également sur les poids, n'altère pas ou très peu la qualité de classification du RNA. En outre, un « bruit » de l'ordre de 20% jusqu'à 50% peut être considérée encore comme acceptable, avec respectivement une erreur totale sur la classification des 699 cas de 3,3% et 6,2%.

Pour fixer la dynamique d'entrée des multiplieurs, il convient de considérer la dynamique (i.e. valeurs extrêmes) des poids de chaque neurone. Le pire cas est de [-3,8 ; 3,5] pour la couche de neurones de sortie, soit une valeur maximale de 3,8. Supposons maintenant que nous choisissions comme échelle relative que 10mV correspond à un poids de 0,01 ; alors un poids de 3,8 correspondra à une tension de 380mV. Il nous faudrait donc une dynamique d'entrée de ±380mV, ce qui ne posera pas de problème avec la technologie utilisée (HCMOS9A) et des circuits alimentés sous ±900mV. Nous vérifierons ce point par des simulations Spice, au chapitre suivant. Par ailleurs, il conviendra d'appliquer un facteur de 0,3 sur les attributs d'entrée afin de les ramener dans l'intervalle [0,3 ; 3] au lieu de [1 ; 10]. Enfin, le bruit ramené en entrée de ce multiplieur ne devrait donc pas dépasser 50% (ou 10% au pire cas pour le multiplieur, mais meilleur pour la classification) de l'entrée la plus faible, à savoir 0,3=30mV, soit 15mv (ou 3mV). Là encore, la contrainte ne sera pas difficile à respecter.

4.5. Création d'un bloc logsigmoid non-idéal

Pour réaliser ce bloc, nous nous sommes inspirés du travail précédent. Ainsi, la fonction d'activation logsigmoid est utilisée en rajoutant en sortie un bruit à travers un multiplieur. Ce mécanisme est décrit sur la figure 3.46.

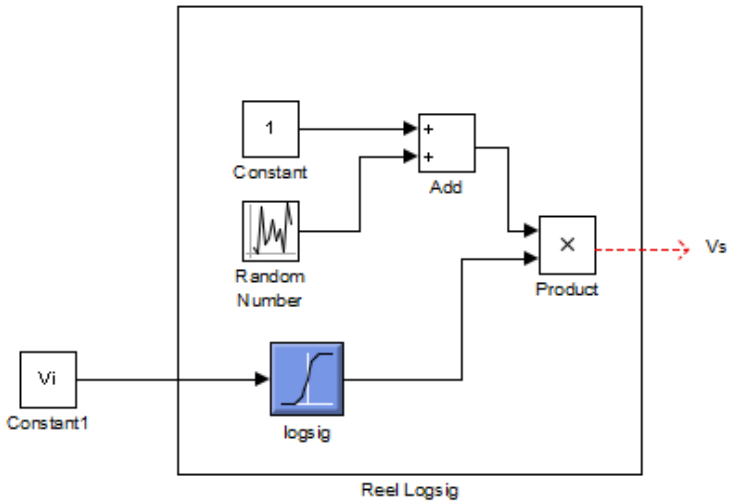


Figure 3.46 : Bloc logsigmoid non-idéal

La figure 3.47 illustre un exemple de résultats de sortie de ce bloc pour une entrée égale à Vi=3. Précisons que sans « bruit », la valeur idéale de sortie devrait être de 0,95.

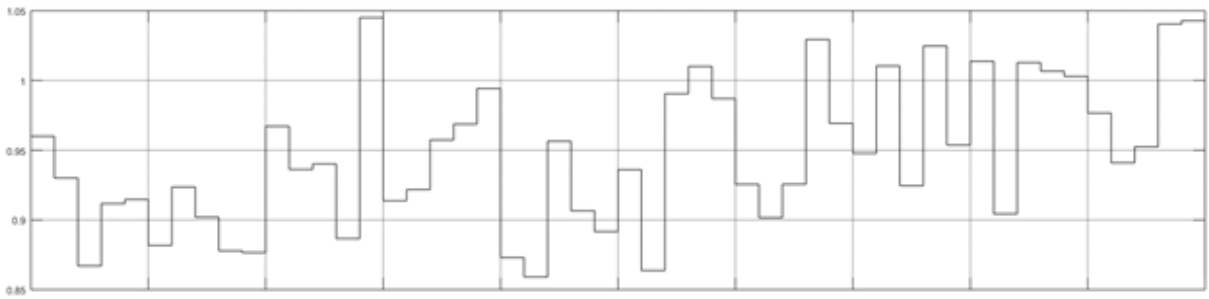


Figure 3.46 : Sortie du bloc logsigmoid non-idéal pour une entrée de Vi=3

Comme dans le paragraphe précédent sur le multiplieur, nous avons réalisé une série de simulations avec différentes valeur de « bruit ». Les résultats sont résumés dans le tableau 3.3, où le bloc logsigmoid non-idéal a remplacé la fonction idéale sous Matlab/Simulink. Lors de ces simulations, nous avons gardé le multiplieur idéal afin d'évaluer leurs influences relatives.

Tableau 3.3 : Résultats avec un bloc logsigmoid non-idéal

er3	er4	Nb Erreur	% Erreur	Poids extrêmes wH	Poids extrêmes wO
-10%	10%	24	3,4%	-1,6 ➔1,4	-3,2 ➔3,8
-20%	20%	49	7%	-1,7 ➔1,2	-4 ➔6,2
-25%	25%	241	34,5%	-1,7 ➔1,2	-4 ➔16

Dans ce tableau nous avons noté les entrées er3 et er4 pour les différencier de celles du multiplieur. Ces résultats montrent qu'une erreur de 10% demeure acceptable (erreur totale égale à 3,4%), par contre une erreur de 20% devient limite, comparée au multiplieur, avec une valeur de 7% d'erreur. De plus, cette erreur devient rédhibitoire pour atteindre presque 35% pour un « bruit » de 25%. Ce bloc est donc beaucoup plus sensible que le multiplieur et sa conception sera plus délicate. Pour cette raison, nous étudierons également la fonction tangente hyperbolique comme alternative dans le chapitre suivant sur la conception des blocs.

4.6. Simulations finales

La dernière étape consiste maintenant à réaliser une dernière série de simulations du RNA en remplaçant simultanément le multiplieur et la fonction logsigmoid par les blocs non-idéaux. Les résultats sont résumés dans les tableaux 3.4 et 3.5.

Tableau 3.4 : Résultats avec un bruit de ±10% pour le bloc logsigmoid

er1	er2	er3	er4	Nb Erreur	% Erreur	Pd extrêmes wH	Pd extrêmes wO
-10%	10%	-10%	10%	22	3,2%	-1,6 ➔1,2	-3,9 ➔4,1
-20%	20%	-10%	10%	24	3,4%	-1,6 ➔1,3	-3,6 ➔3,6
-30%	30%	-10%	10%	28	4%	-1,4 ➔1,5	-3 ➔3
-40%	40%	-10%	10%	30	4,3%	-1,6 ➔1	-4 ➔3,4
-50%	50%	-10%	10%	38	5,3%	-1,7 ➔1,5	-3,2 ➔3,4
-60%	60%	-10%	10%	241	34,5%	-5,2 ➔6,5	-6 ➔48

Ces résultats sont relativement proches de ceux présentés au tableau 3.2. Ainsi l'erreur sur le multiplieur peut atteindre 50% pour une classification acceptable. Toutefois un « bruit » inférieur à 30% offre une erreur finale en sortie du RNA inférieure à 4%. Par contre un « bruit » cumulé sur les deux blocs de ±20% est trop élevé pour assurer une classification efficiente, comme le montre le tableau 3.5.

Tableau 3.5 : Résultats avec un bruit de ±20% pour le bloc logsigmoid

er1	er2	er3	er4	Nb Erreur	% Erreur	Pd extrêmes wH	Pd extrêmes wO
-10%	10%	-20%	20%	48	6,9%	-2,2 ➔1,2	-4 ➔6,2
-20%	20%	-20%	20%	241	34,5%	-2,1➔2,3	-3,8 ➔11

5. Conclusion

Le RNA MLP BP est la technique de réseaux de neurones la plus utilisée, car la plus efficace, pour la classification. Dans ce chapitre, nous avons démontré, dans un premier temps, qu'un algorithme sans biais pour la fonction de transfert, donnait des résultats équivalents à meilleurs

qu'un algorithme avec biais dans le cadre de notre application. Cette première conclusion nous permettra d'une part de simplifier la structure finale de l'implémentation matérielle et d'autre part d'en diminuer la consommation.

Par ailleurs, l'étude des différentes structures de ce RNA nous a permis de déterminer l'architecture optimale offrant le moins d'erreur lors de la classification des biopsies de référence issues de la base de données Wisconsin. Ainsi un RNA sans biais sera retenu avec une fonction d'activation de type logsigmoid pour tous les neurones de la couche cachée et ceux de la couche de sortie. Les valeurs optimales de ce réseau sont :

- 9 neurones pour la couche d'entrée (9 attributs de la base de données)
- 10 neurones pour la couche cachée
- 2 neurones pour la couche de sortie (2 classes, bégnine ou maligne, de la base de données)
- Nombre d'époques : 14
- Taux d'apprentissage : 0,25

Les résultats de simulation nous également permis de déterminer les valeurs des différents poids affectés à chaque neurone. Ainsi les valeurs extrêmes (minimum et maximum) sont respectivement [-1,37 ; 1,17] pour la couche cachée et [-3,54 ; 3,48] pour la couche de sortie, dans le cas idéal. Enfin, nous avons ensuite affinés les modèles des blocs de base (multiplieur et fonction d'activation) afin d'évaluer l'influence de certaines non-idéalités de ces circuits sur la qualité de la classification. Cette étude nous a permis de définir certaines spécifications de ces blocs de base. En particulier le taux (ou pourcentage) d'erreur finale de la classification est très sensible aux imperfections de la fonction d'activation logsigmoid. Ainsi, si nous choisissons une échelle de 10mV pour un poids (ou une entrée) de 0,1 et en considérant la valeur maximale absolue sur les différents tableaux, la dynamique d'entrée de ces circuits devrait être d'environ ±400mV. En considérant une erreur maximale acceptable de 10% (à 20% maximum) sur les entrées du bloc logsigmoid, son bruit ne doit pas dépasser les 3mV. Dans ce cas, le bruit sur le multiplieur peut atteindre « 40% » (soit 12mV) avec une erreur totale de la classification de 4,3%.

L'étape suivante consiste donc à concevoir ces blocs de base que sont le multiplieur, la fonction logsigmoid et sa dérivée. Le dimensionnement des transistors permettra d'extraire leurs caractéristiques, puis de vérifier le taux d'erreur de la classification. Notons que si la dynamique d'entrée n'est pas vérifiée, nous pouvons toujours modifier (diminuer) notre échelle de 10 mv par poids ou entrée Cancer (attributs) de 0,1. Ce facteur d'échelle devra être, bien évidemment, appliqué au bruit ramené à l'entrée de ces circuits. Il est également possible de choisir des échelles différentes sur les 2 entrées du multiplieur.

6. Bibliographie

[1] Z. Chonglin, Y. Lijuan, and W. Weibing,, “Study on fitness data processing based on neural network information processing,,” *International Conference on Systems and Informatics (ICSAI),2012,* pp. 295-298.

[2] Z. Ying, G. Jun, and Y. Xuezhi,, “A survey of neural network ensembles,,” *in International Conference on Neural Networks and Brain, 2005. ICNN&B ‘05,2005,* pp. 438-442.

[3] H. Husain, M. Khalid, and R. Yusof,, “Nonlinear function approximation using radial basis function neural networks,,” *Student Conference on Research and Development,2002,* pp. 326-329.

[4] L. Marquez and T. Hill,, “Function approximation using backpropagation and general regression neural networks,,” *“ in Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences,1993,* vol. 4, pp. 607-615.

[5] T. Varshney and S. Sheel,, “Approximation of 2D function using simplest neural networks;,” *A comparative study and development of GUI system,” 2010 International Conference on Power, Control and Embedded Systems (ICPCES),2010,* pp. 1-4.

[6] G. Hongliang,, “The probability characteristic of function approximation based on artificial neural network,,” *in 2010 International Conference on Computer Application and System Modeling (ICCASM),2010,* pp. V1-66-V1-71.

[7] F. Heimes and B. van Heuveln,, “The normalized radial basis function neural network,,” *IEEE International Conference on Systems, Man, and Cybernetics,1998,* vol. 2, pp. 1609-1614.

[8] Jayadeva, A. K. Deb, and S. Chandra,, “Algorithm for building a neural network for function approximation,,” *IEE Proceedings –Circuits, Devices and Systems,2002,* vol. 149, pp. 301-307.

[9] R. Setiono and A. Gaweda,, “Neural network pruning for function approximation,,” *International Joint Conference on Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS,2000,* vol. 6, pp. 443-448.

[10] S. Ferrari and R. F. Stengel,, “Smooth function approximation using neural networks,,” *IEEE Transactions on Neural Networks,2005,* vol. 16, pp. 24-38.

[11] “https://www.mathworks.com/products/neural-network.html,” [Online].

[12] N. P. Thanh, Y.-S. Kung, S.-C. Chen and H.-H. Chou, “Digital hardware implementation of radial basis function neural network,2016,” in *ScienceDirect*.

[13] A.Tisan and M.Cirstea, "SOM neural network design-A new Simulink library based approach targeting FPGA implementation," in Elsevier, 2012.

[14] Jayadeva, A. Deb and S. Chandra, “Algorithm for building a neural network for function approximation,” *IEEE Proceedings –Circuits, Devices and Systems,2002,* pp. 301-307.

[15] Donald F. Specht, “A General Regression Neural Network,” *IEEE TRANSACTIONS ON NEURAL NETWORKS,NOVEMBER 1991,* vol. 2. NO. 6.

[16] “http://archive.ics.uci.edu/ml/index.php,” [Online].

[17] "MATLAB and Statistics Toolbox Release 2015a, The MathWorks, Inc., Natick, Massachusetts, United States.".

[18] H. Jouni, A. Harb, G. Jackmod and Y. Leduc, "Wide Range Analog CMOS Multiplier for Neural Network Application," in EEETEM2017, Beirut, 2017.

[19] S. T. Perez, J. L. Vasquez, C. M. Travieso and J. B. Alonso, "Artificial Neural Newtork in FPGA for Temperature Prediction," in Springer-Verlag, Berlin Heidelberg, 2011.

[20] "Matlab Neural Network Toolbox," in http://www.mathworks.com/.

[21] Jouni Hassan, Issa Mariam, Harb Adnan, Jacquemod Gilles, Leduc Yves, "Neural Network architecture for breast cancer detection and classification," in 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), Beirut, 2016.

Chapitre IV – Conception des blocs de base

1. Introduction

L'étude précédente nous a permis de choisir le réseau de neurones le plus approprié à notre objectif de classification de cellules cancéreuses (cf. Figure 4.1) [1]. En outre, quelques spécifications des blocs de base ont été extraites, leur surface et consommation devront, quant à elle, être minimisées le plus possible. Ainsi, la fonction d'activation logsigmoid impose des contraintes sur la précision (donc le bruit) des entrées plus restrictives que le multiplieur pour atteindre la qualité finale désirée de la classification. Sans anticiper le résultat final et pour limiter les risques, nous étudierons également la fonction d'activation tangente hyperbolique. Pour des raisons de coût et de consommation, la technologie pour leur réalisation est HCMOS9A (CMOS 130nm) de la société STMicroelectronics avec une alimentation de 1,8V (soit ±900mV, qui correspondra à la dynamique maximale en entrée et en sortie des circuits).

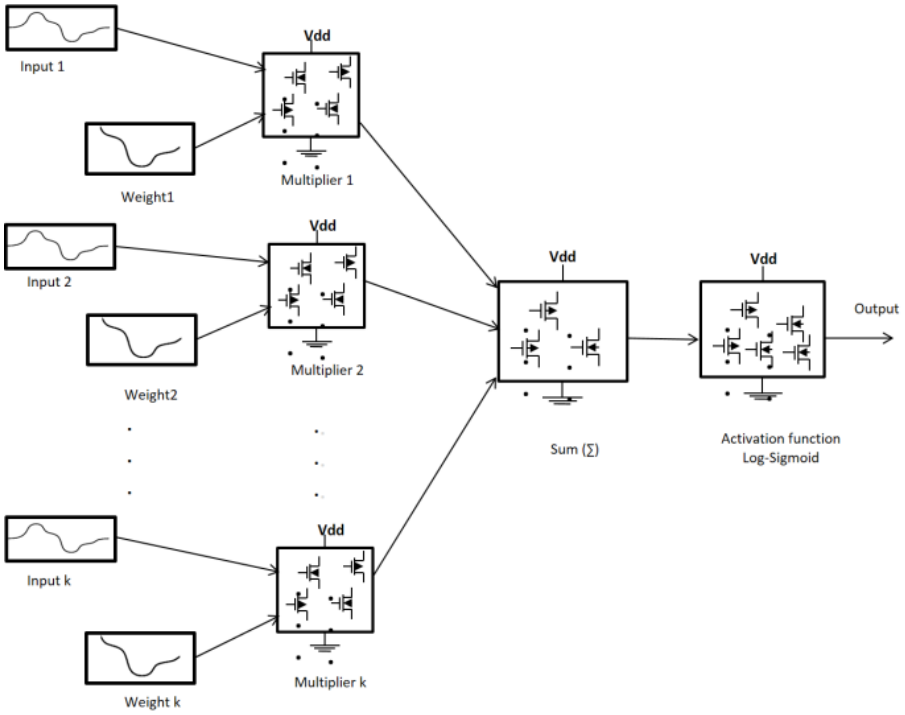


Figure 4.1: Implémentation en blocs de base du neurone Singleton

Sur la figure 4.1, les entrées du multiplieur seront des tensions et la sortie un courant. Les entrées du sommateur seront donc des courants (loi KCL) et la sortie une tension. La fonction d'activation (en tension) sera donc soit logsigmoid, soit tangente hyperbolique.

Le tableau 4.1 résume les différents paramètres des deux topologies étudiées, fonction d'activation identique, logsigmoid ou tangente hyperbolique, à la fois pour les neurones de la couche cachée et celle de sortie (cf. Annexe 1). Notre première étude portera sur le multiplieur.

Tableau 4.1 : Paramètres utilisés (cas idéal)

Fonction d'activation Couche cachée et de sortie	**Nb Erreur**	**% Erreur**	**Pd extrêmes wH Couche cachée**	**Pd extrêmes wO Couche de sortie**
Tangente hyperbolique	46	6,6%	-1,65 ➔1,55	-3,97➔1,83
Logsigmoid	18	2,6%	-1,37➔1,17	-3,54 ➔3,48

2. Conception du multiplieur

2.1. Introduction

Le multiplieur est un bloc important dans les réseaux neuronaux surtout s'il est implémenté par des circuits CMOS analogiques [2]. En fonction de la polarisation du transistor, régime linéaire ou de saturation, en faible ou forte inversion, sous le seuil, de nombreuses topologies sont disponibles pour réaliser un tel circuit [3]. Les principales caractéristiques d'un multiplieur sont sa dynamique (d'entrée et de sortie), sa linéarité et sa bande passante [3]. Ainsi, nous pouvons proposer différents modèles de multiplieurs basés sur les équations mathématiques et dépendants de différentes combinaisons de tension et de courant pour les entrées et les sorties [4].

Par exemple, selon l'équation (rel 4.1), nous pouvons concevoir un multiplicateur CMOS analogique à quatre quadrants. Dans cette structure, les entrées et sorties sont des tensions.

$$V_0 = [(V_1 + V_2)^2 - (V_1 - V_2)^2] = 4V_1V_2 \qquad \text{(rel 4.1)}$$

Cependant, ce type de multiplieur offre une tension de sortie différentielle avec une faible plage de linéarité [5]. Une autre méthode est basée sur la relation d'approximation en loi quadratique du transistor MOS saturé entre le courant de drain et la tension de grille. Dans ce type de circuit, nous ne pouvons pas bénéficier de toute la gamme d'alimentation [6]. Par exemple, Sawigun a proposé un multiplieur à quatre quadrants avec une alimentation de 1,5 V mais avec une consommation de puissance relativement élevée d'environ 290 µW [7] .

Dans ce chapitre, nous proposons de concevoir un multiplieur CMOS analogique avec une dynamique d'entrée élevée et une très faible consommation d'énergie. L'une des tensions d'entrée de ce multiplieur correspond au poids, Vw, utilisé dans l'algorithme ANN (Artificial Neural Network) pour la détection et la classification du cancer [1]. L'autre entrée correspond à l'image de la biopsie,

Entrée Cancer (attributs de la base de données Wisconsin), que nous devons traiter. Deux topologies seront étudiées. Conformément à nos études précédentes, la première utilise comme grandeurs d'entrée des tensions et un courant pour la sortie. Au regard des valeurs résumées dans le tableau 4.1, ces poids (donc les tensions d'entrée) sont respectivement compris entre -1,37 et 1,17 pour la couche cachée et entre -3,54 et 3,48 pour celle de sortie. Fournissant un courant en sortie, nous pourrons directement attaquer le sommateur. En outre, comme nous l'avons vu au chapitre précédent, les 9 attributs des 699 biopsies peuvent correspondre à des tensions comprises entre 30mV et 300mV (avec une échelle de 10mv par coefficient (poids ou entrée) de 0,1). La dynamique d'entrée du multiplieur est alors imposée par celle des poids qui est plus élevée que celle des attributs. Les valeurs minimales et maximales pour les poids dans la couche cachée et le neurone de sortie varient entre -3,54 et 3,48, pour la fonction d'activation logsigmoid. La dynamique d'entrée du multiplieur CMOS analogique est donc de -354mV à 348mV. Si on prend en compte le « bruit » sur les entrées nous avons montré, au chapitre précédent, que cette dynamique devrait augmenter jusqu'à 380mV (voir 400mV avec la fonction d'activation tangente hyperbolique).

Par ailleurs, nous pouvons appliquer un facteur multiplicatif sur les attributs de 0,1 au lieu de 0,3 comme précédemment, ainsi la tension d'entrée correspondante aux biopsies sera différente, à savoir 10mV à 100mV (au lieu de 300mV). Ainsi, nous pouvons choisir comme dynamique d'entrée du multiplieur des valeurs différentes :

- 0 à 100mV pour la « première » entrée (10mV à 100mV en fait)
- ±400mV pour la « seconde » entrée (-354mV à 348mV sans « bruit »)

La deuxième topologie consiste à créer un multiplieur CMOS analogique avec des courants d'entrée et un courant de sortie (mode courant). Similaire au précédent, la dynamique d'entrée sera comprise, dans ce cas, entre 0 et 1µA pour la première entrée, et -3,54 µA et 3,48 µA pour la seconde. Comme pour les entrées en tension, il est préférable d'étendre légèrement cette dynamique pour prendre en compte les erreurs de stockage des coefficients et le bruit du multiplieur, soit ±4µA. Nous pouvons passer à la conception du multiplieur avec des entrées en tension et une sortie en courant.

2.2. Multiplieur à paire différentielle

2.2.1. Caractéristiques d'une paire différentielle

Dans la conception illustrée par la figure 4.2, le multiplieur est basé sur l'équation (rel 4.2) où I_o et v_i sont respectivement le courant de polarisation et l'entrée de la paire différentielle [8]. Sur cette figure, on suppose que les transistors M_1 et M_2 sont identiques (parfaitement appairés) et polarisés

en régime de saturation. De plus, l'effet de modulation de la longueur du canal est ignoré. Ainsi, on suppose que le courant de drain de chaque transistor suit une loi quadratique.

$$i_{out} = \sqrt{\beta I_0}\, v_i \qquad \text{(rel 4.2)}$$

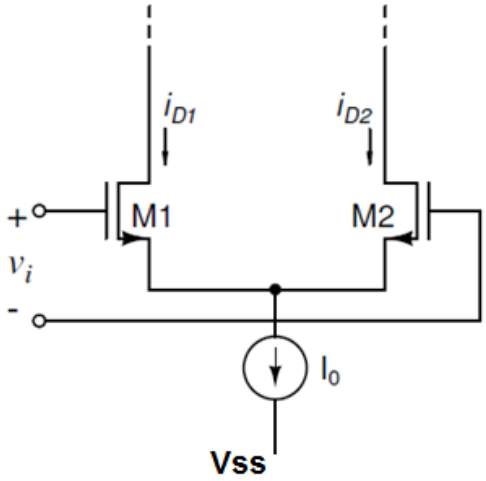


Figure 4.2: Paire différentielle

La tension grille-source des transistors M_1 et M_2 est régie par la relation (rel 4.3) suivante :

$$v_{GS1,2} = V_T + \sqrt{\frac{2i_{D1,2}}{\beta}} \qquad \text{(rel 4.4)}$$

où $\beta = \mu_N C_{ox}(W_{1,2}/ L_{1,2})$. μ_N est la mobilité des électrons, C_{ox} est la capacité de la grille par unité de surface, W et L sont respectivement la largeur et la longueur du canal de M_1 et M_2. Or $v_i = v_{GS1}-v_{GS2}$, la relation (rel 4.4) nous donne :

$$\sqrt{i_{D1}}-\sqrt{i_{D2}} = \sqrt{\frac{\beta}{2}}\, v_i \qquad \text{(rel 4.5)}$$

La somme des courants de drain au nœud commun des sources donne $i_{D1}+i_{D2}=I_0$. En utilisant la relation précédente (rel 4.6), on obtient :

$$\left(\sqrt{i_{D1}} - \sqrt{i_{D2}}\right)^2 + 2\sqrt{i_{D1}}\sqrt{i_{D2}} = I_0 \qquad \text{(rel 4.6)}$$

Après calcul, nous obtenons pour le courant de sortie différentielle :

$$i_{out} = i_{D1} - i_{D2} = \sqrt{\beta I_0} v_i \sqrt{1 - \frac{\beta v_i^2}{4I_0}} \qquad \text{(rel 4.7)}$$

Soit si $\frac{\beta v_i^2}{4I_0} \ll 1 \left(v_i \ll 2\sqrt{\frac{I_0}{\beta}}\right)$, on obtient : $i_{out} = \sqrt{\beta I_0} v_i$ (rel 4.8)

La relation (rel 4.8) montre que la sortie est linéaire avec la racine carrée du courant de polarisation et multiple de la tension différentielle v_i. Cette relation est vérifiée si cette entrée différentielle reste faible (cf. rel 4.9). La figure 4.3 illustre les variations du courant de sortie différentielle en fonction de v_i.

$$-\sqrt{\frac{2I_0}{\beta}} < v_i < \sqrt{\frac{2I_0}{\beta}} \qquad \text{(rel 4.9)}$$

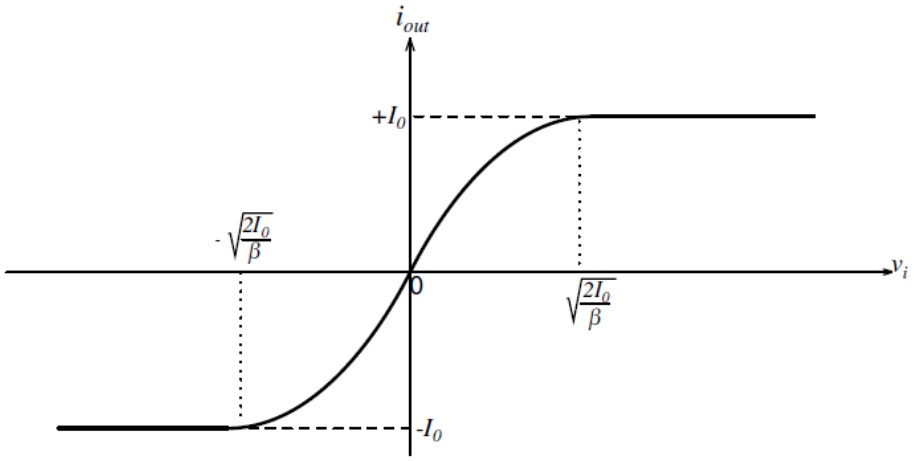


Figure 4.3: Courant de sortie d'une paire différentielle

Dans le cadre de notre application pour le multiplieur final et aux vues des tensions associées aux poids et aux attributs (400mV max pour les poids et 100mV max pour les attributs), nous choisirons de prendre l'entrée différentielle comme valeur des attributs. Il conviendra alors de vérifier la relation (rel 4.9) pour s'assurer de la linéarité du multiplieur final. Les poids seront donc associés au courant de polarisation I_0, qu'il faudra élever au carré pour avoir une relation de sortie linéaire. Une paire différentielle à charge active et un miroir de courant doivent nous permettre de réaliser ces opérations.

2.2.2. Paire différentielle à charge active

Un amplificateur différentiel à sortie asymétrique peut être mis en œuvre en plaçant un miroir de courant PMOS comme charge sur le schéma de la figure 4.2. La figure 4.4 présente la topologie classique d'une telle structure d'une paire différentielle à charge active. Ainsi, la « première » entrée du multiplieur sera la tension d'entrée différentielle de ce circuit, soit $v_i=v_2-v_1$. Dans la pratique, on prendra $v_1=0$ et V_2 correspondra donc à la valeur d'un des attributs de la cellule Cancer (image d'une biopsie), dont la valeur maximale a été fixée arbitrairement à 100mV, entre 10 et 100mV [1].

D'autre part, en raison de la racine carrée du courant de polarisation I_0, il est nécessaire de réaliser un circuit quadratique fournissant en sortie un courant proportionnel au carré de la tension

d'entrée (i.e. la valeur du poids). Cette tension, notée V_W, sera donc la « seconde » entrée du multiplieur final, qui devra donc avoir une dynamique d'au moins ±400mV. Un simple miroir de courant, illustré par la figure 4.5, permet de réaliser cette opération

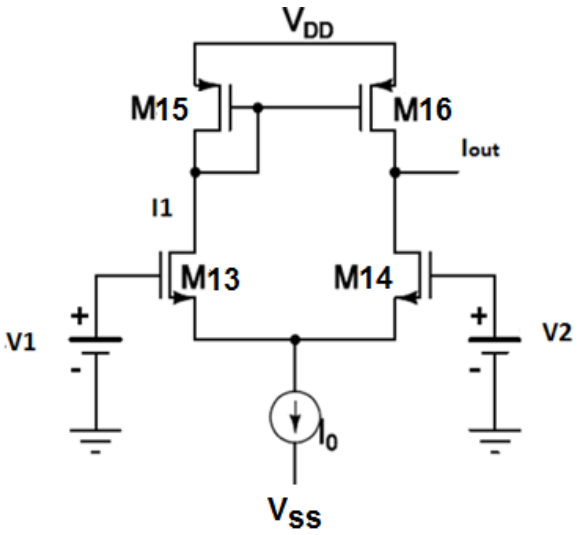


Figure 4.4: Paire différentielle à charge active

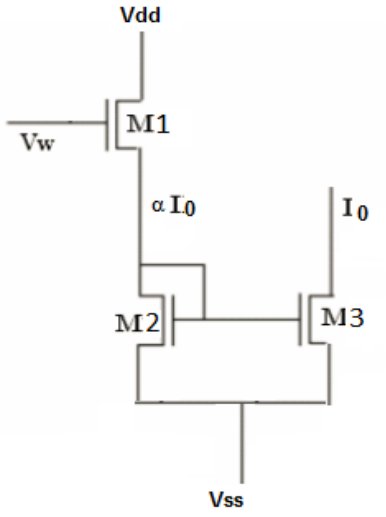


Figure 4.5: Source de courant commandée en tension

2.3. Topologie du multiplieur

2.3.1. Schéma à transistors

En prenant en compte toutes les considérations présentées aux paragraphes précédents, nous en déduisons le synoptique du multiplieur final décrit par la figure 4.6.

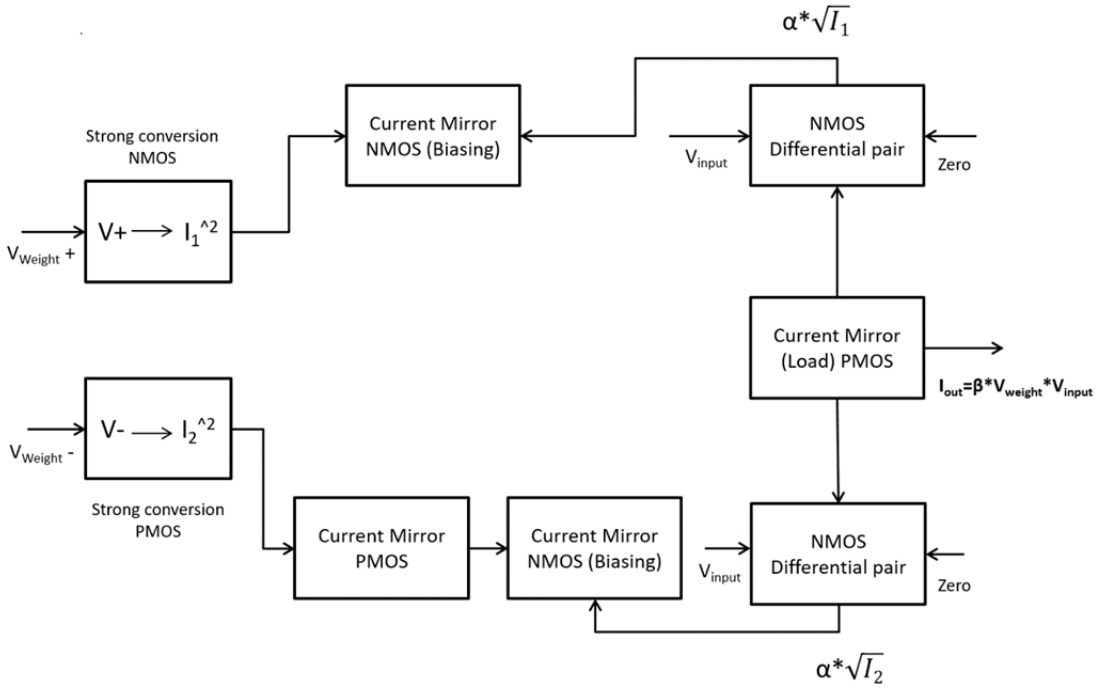


Figure 4.6: Schéma bloc du multiplieur

Sur la figure 4.6, les attributs correspondent à l'entrée v_i et les poids aux entrées V_{W+} et V_{W-}, qu'il conviendra de réaliser à partir d'une seule tension d'entrée V_W. Ainsi le contenu de chaque bloc est réalisé par un circuit à transistors, présenté sur la figure 4.7, qui décrit ainsi le multiplieur final.

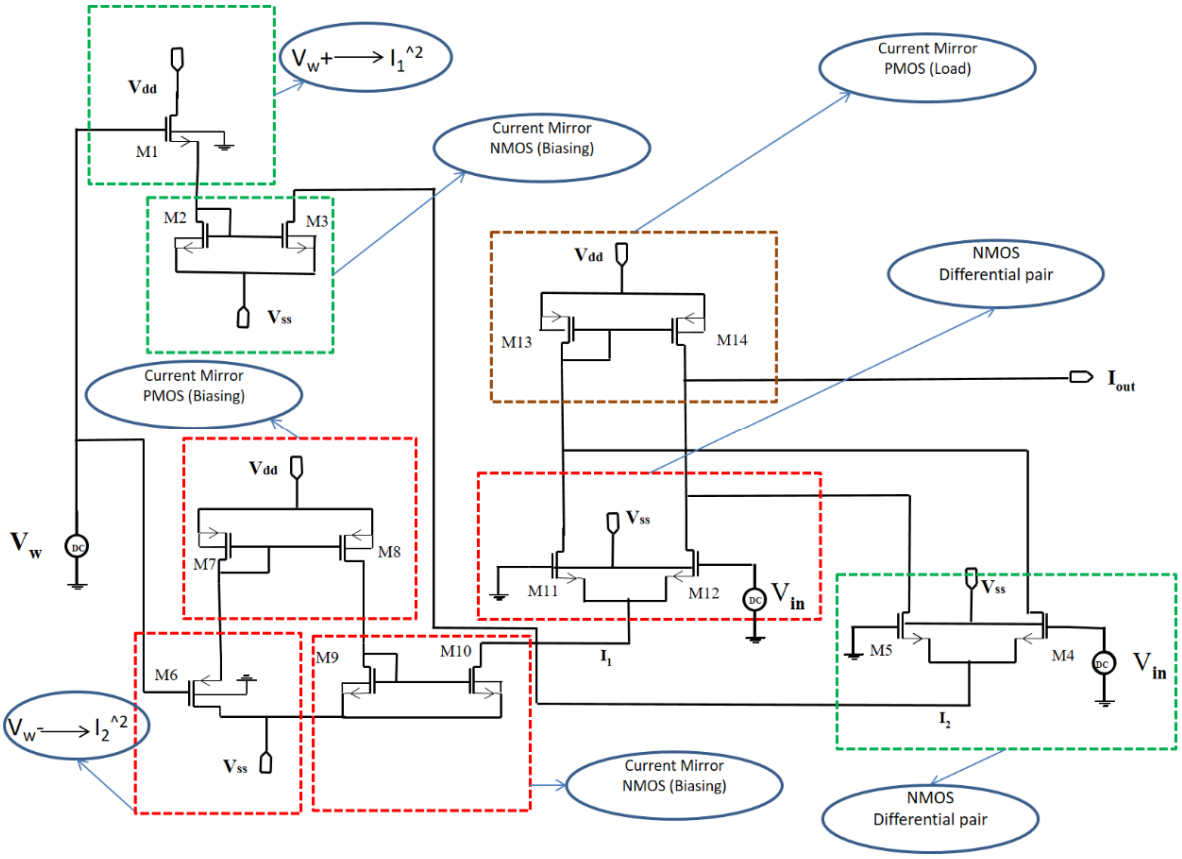


Figure 4.7: Description de chacun des blocs du multiplieur

Pour expliquer plus en détail, le comportement de ce multiplieur à 4 quadrants (cf. Figure 4.7), rappelons quelques caractéristiques du fonctionnement d'un transistor. Ainsi, un transistor ne passe pas immédiatement d'un comportement exponentiel en faible inversion à un comportement quadratique en forte inversion. Il y a une transition douce entre les deux extrêmes où le courant de « drift » et le courant de diffusion existent simultanément sans qu'aucun effet ne domine. La modélisation de cette zone est extrêmement difficile, mais le comportement est facilement compris comme un hybride entre un comportement d'inversion faible et fort.

Les limites entre l'inversion faible, modérée et forte restent floues, mais peuvent être approchées en termes de tensions ou de courants :

- $V_{GS} > V_T + 100mV$ inversion forte
- $V_T + 100mV > V_{GS} > V_T - 100mV$ inversion modérée
- $V_{GS} < V_T - 100mV$ inversion faible

Il est souvent préférable de concevoir les circuits en définissant les courants de polarisation. On peut alors utiliser les définitions suivantes afin de délimiter les limites d'inversion :

- $I_0 > 10I_S$ inversion forte (rel 4.10)
- $10I_0 > I_0 > 0{,}1I_S$ inversion modérée (rel 4.11)
- $I_0 < 0{,}1I_S$ inversion faible (rel 4.12)

Où I_S est appelé courant caractéristique d'inversion modérée. Il est défini par la relation (4.13).

$$I_s = \frac{2\mu C_{ox}}{k} U_T^2 \frac{W}{L} = \frac{2KP_{N,P}}{k} U_T^2 \frac{W_{N,P}}{L_{N,P}} \quad \text{(rel 4.13)}$$

Avec U_T=26mV (tension thermique : kT/q) et k#0,7. µ est la mobilité des charges.

2.3.2. Dimensionnement des transistors

Le tableau 4.2 résume les caractéristiques des paramètres Spice des transistors (nmos1v8 et pmos1v8) choisis de la technologie HCMOS9, CMOS 130nm de la société STMicroelectronics. On rappelle : ε_0=8,854*10^{-12} F/m et ε_r=3,9, soit ε_{0X}= ε_0* ε_r=3,45*10^{-11} F/m.

Tableau 4.2 : Paramètres Spice des Transistors

Paramètres Spice	NMOS	PMOS
T_{OX} (nm)	2,76	2,78
$C_{OX} = \varepsilon_{0X}/T_{OX}$ (F/m^2)	1,25.10^{-2}	1,24.10^{-2}
µ (m^2/Vs)	2,98.10^{-2}	6,27.10^{-3}
KP=µ C_{OX} (µA/V^2)	373	78

Afin de limiter la consommation finale du multiplieur complet, on choisit un courant de polarisation de notre structure complète (cf. Figures 4.4 et 4.7) de I_0 = 1µA. Afin d'assurer un fonctionnement des transistors correspondants (paires différentielles) en régime de forte inversion, la condition I_0>10I_S doit être vérifiée. En considérant l'équation (rel 4.13), nous obtenons pour les transistors NMOS la relation suivante : I_{SN}(µA)=10.I_0=10µA=0,721(W_N/L_N), soit W_N/L_N=0,1386. En appliquant la même relation pour les transistors PMOS, nous obtenons la relation suivante : I_{SP}(µA)=10.I_0/2=5µA=0,15(W_P/L_P), soit W_P/L_P=0,333. Il convient de noter que le courant de polarisation des PMOS est égal à I_0/2 (cf. Figure 4.7).

En reprenant la figure du multiplieur de la figure 4.7 et en choisissant W_n=300nm, on obtient L_n=2,163µm pour les transistors M_4, M_5, M_{11} et M_{12}. En fixant W_p=150nm pour les transistors PMOS, nous en déduisons L_p=450nm pour les transistors M_{13} et M_{14}. Le dimensionnement des transistors des générateurs de courant (i.e. poids V_W) est basé sur la relation suivante : I_{out}=v_{in}*√I_1 où I_1 suit une loi quadratique en fonction du poids d'entrée V_W (I_1=αV_W^2). Pour dimensionner les transistors, nous

avons considéré les valeurs maximales des entrées : $V_W \in [-0,4 ; 0,4]$ et $v_{in} \in [-0,1 ; 0,1]$. Finalement, le tableau 4.3 résume le dimensionnement des transistors du multiplieur complet (cf. Figure 4.7).

Tableau 4.3 : Dimensions des transistors du multiplieur

Transistor	Type	Largeur (nm)	Longueur (µm)
M1	NMOS	150	10,0
M2	NMOS	150	1,55
M3	NMOS	580	6,0
M4	NMOS	300	2,16
M5	NMOS	300	2,16
M6	PMOS	180	3,0
M7	PMOS	500	0,51
M8	PMOS	640	0,9
M9	NMOS	500	1,0
M10	NMOS	150	0,15
M11	NMOS	300	2,16
M12	NMOS	300	2,16
M13	PMOS	150	0,45
M14	PMOS	150	0,45

2.4. Simulations Spice

2.4.1. Analyse DC

Comme le montre la figure 4.8, le multiplieur offre une plage de linéarité sur I_{out} en fonction de v_{in} d'environ -100mV et 100mv, respectant notre cahier des charges. Cette entrée peut ainsi correspondre aux 9 attributs de la base de données Wisconsin (appelés entrée Cancer, cf. Tableau 1.3) dont les valeurs sont comprises entre 1 et 10. 1 correspondra à une tension de 10mV et 10 à une de 100mV. Cette première dynamique d'entrée est suffisante pour notre application.

La seconde entrée du multiplieur correspond aux poids des neurones, donc l'objectif est de convertir cette valeur en fonction du carré du courant de drain du miroir de courant afin là encore d'avoir ensuite une relation linéaire sur le courant de sortie du multiplieur. Les simulations DC correspondantes sont présentées sur la figure 4.9. Ces résultats montrent une plage de linéarité d'environ ±900mV, respectant là encore notre cahier des charges. En conclusion, cette seconde entrée du multiplieur pourra être utilisée pour les poids de notre algorithme. Notons que cette dynamique n'est pas nécessaire puisqu'initiallement ±400mV était suffisante ($V_W \in [-0,4 ; 0,4]$). Si nous doublons la sensibilité sur cette entrée, une dynamique de ±800mV sera suffisante.

Les simulations DC, illustrées par les figures 4.8 et 4.9, permettent de valider la dynamique d'entrée du multiplieur dans le cas de son utilisation dans le réseau de neurones artificiels dédié à la

classification des cellules cancéreuses. Avant de continuer l'étude de ce circuit avec des analyses transitoires, nous avons voulu vérifier de façon plus précise cette dynamique et la linéarité du multiplieur correspondante.

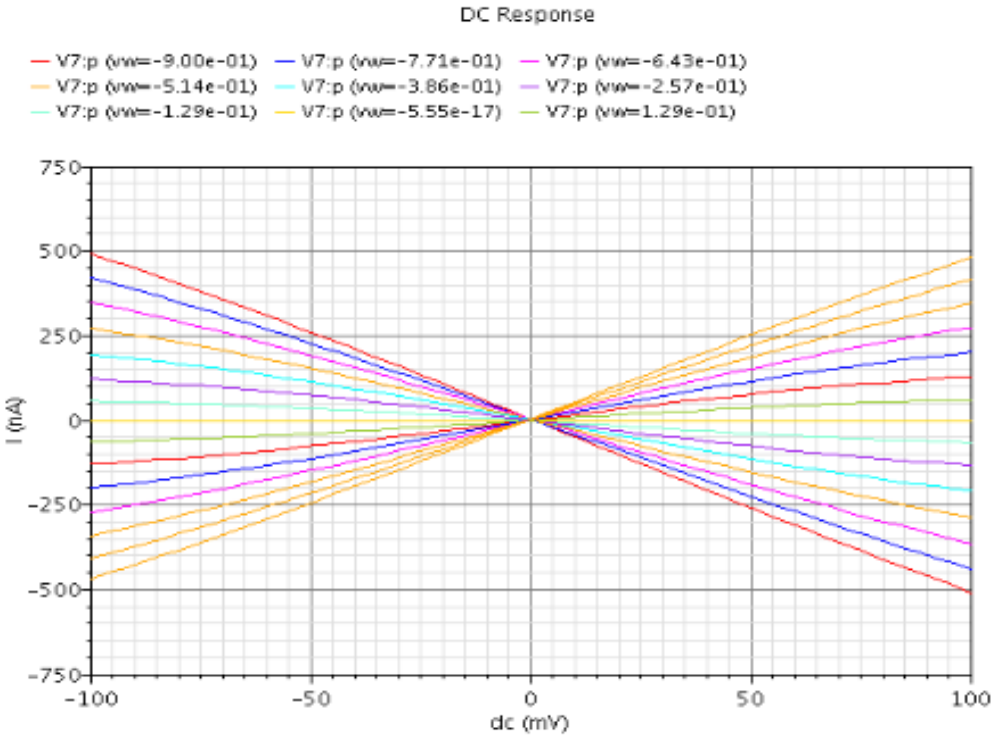


Figure 4.8: I_{out} en fonction de v_{in} pour différentes valeurs de V_W

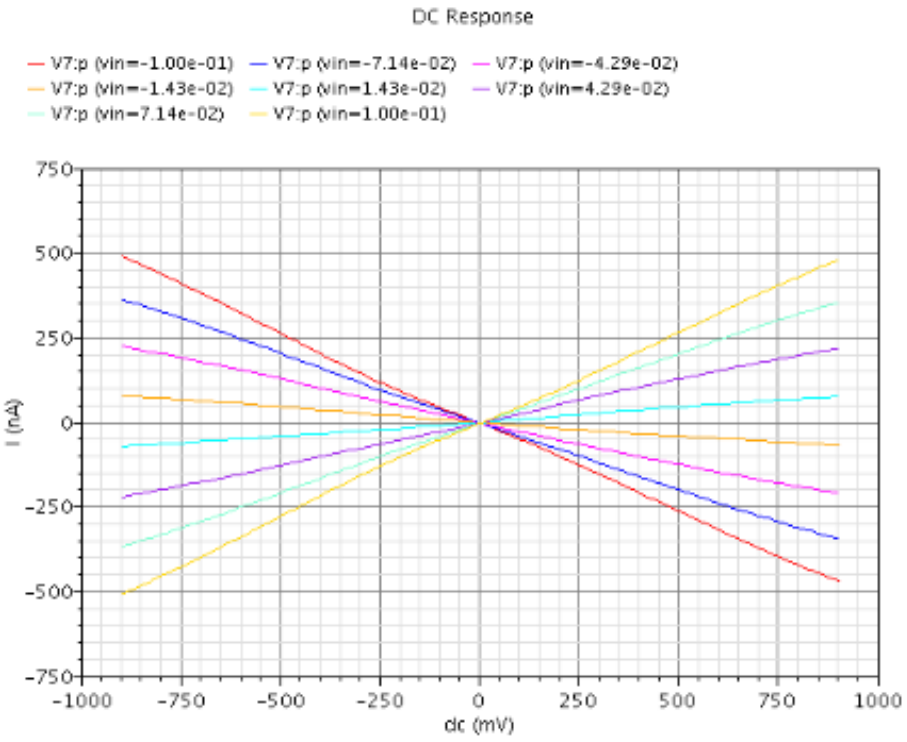
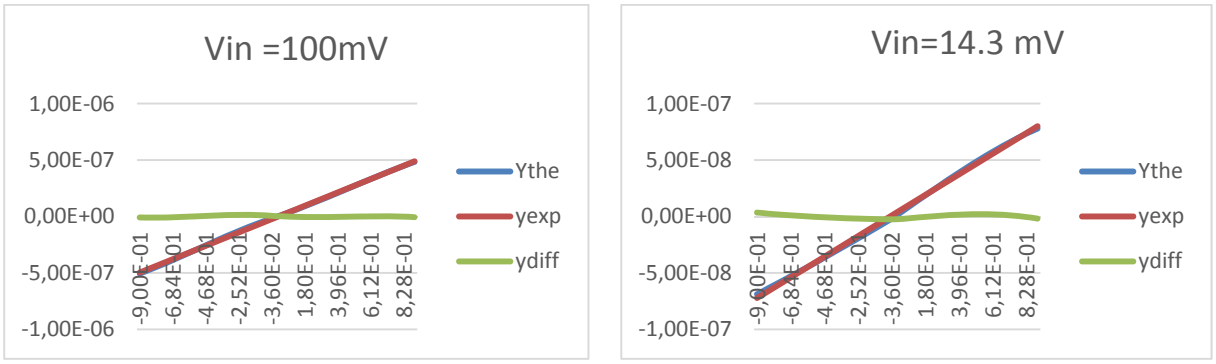


Figure 4.9: I_{out} en fonction de V_W pour différentes valeurs de v_{in}

2.4.2. Vérification de la linéarité du multiplieur

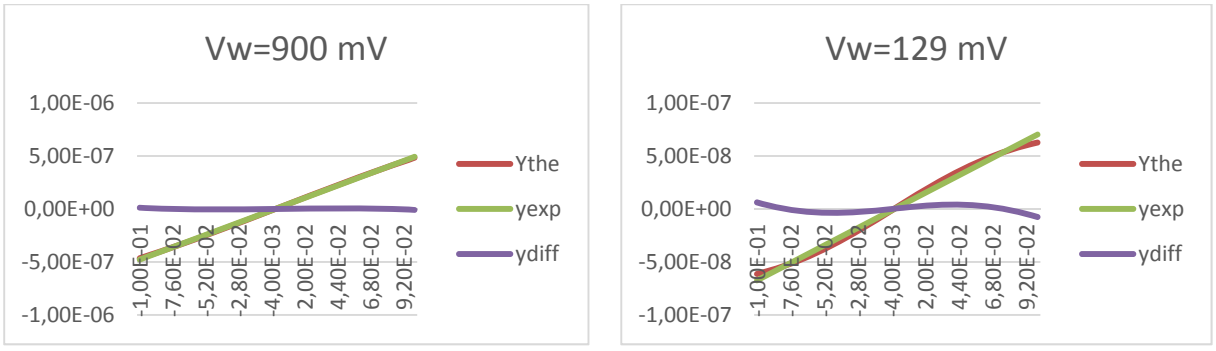
Pour vérifier cette linéarité, nous avons comparé la sortie du multiplieur avec la valeur idéale de sortie (produit des entrées). La figure 4.10 donne l'écart entre ces deux valeurs pour une entrée v_{in} respectivement de 100mV (a) et 14,3mV (b) pour des entrées de poids variant de -900mV à 900mV. Il convient de noter que le courant de sortie est donné en valeur relative, sans unité. L'écart maximum reste toujours inférieure 5 %.



(a) Vin=100mV (b) Vin=14,3mV

Figure 4.10 : Erreur en sortie du multiplieur en fonction de V_W

De façon similaire, la figure 4.11, donne l'écart entre ces deux valeurs pour un poids V_W respectivement de 900mV et 129mV pour des entrées (Cancer) variant de -100mV à 100mV. L'écart maximum, atteint pour V_W=900mV (valeur de la tension d'alimentation), reste toujours inférieure à 10%. Dans le cas de notre application finale, une dynamique de 800mV sera suffisante, diminuant ainsi cette erreur maximale.



(a) V_W=900mV (b) V_W=129mV

Figure 4.11 : Erreur en sortie du multiplieur en fonction de v_{in}

Dans les deux cas, les erreurs restent très inférieures aux valeurs limites déterminées dans le chapitre précédent sur l'influence de la précision des grandeurs analogiques sur la qualité du système final de classification. En conclusion, cette linéarité et cette dynamique sont suffisantes pour notre application.

2.4.3. Analyse en transitoire et consommation d'énergie

L'analyse transitoire est présentée pour deux configurations différentes, avec une dynamique d'entrée maximale (i.e. 900mV pour la première entrée du multiplieur (poids V_W) et 100mV pour la seconde entrée (entrée Cancer) v_{in}). Dans le premier cas, illustré par la figure 4.12, la fréquence de v_{in} est de 100kHz et celle de V_W est de 1MHz . Dans la seconde configuration (cf. Fig 4.13), les fréquences des entrées ont été interverties (i.e. 1MHz pour v_{in} et 100kHz pour V_W). Les résultats obtenus son conformes à ceux attendus.

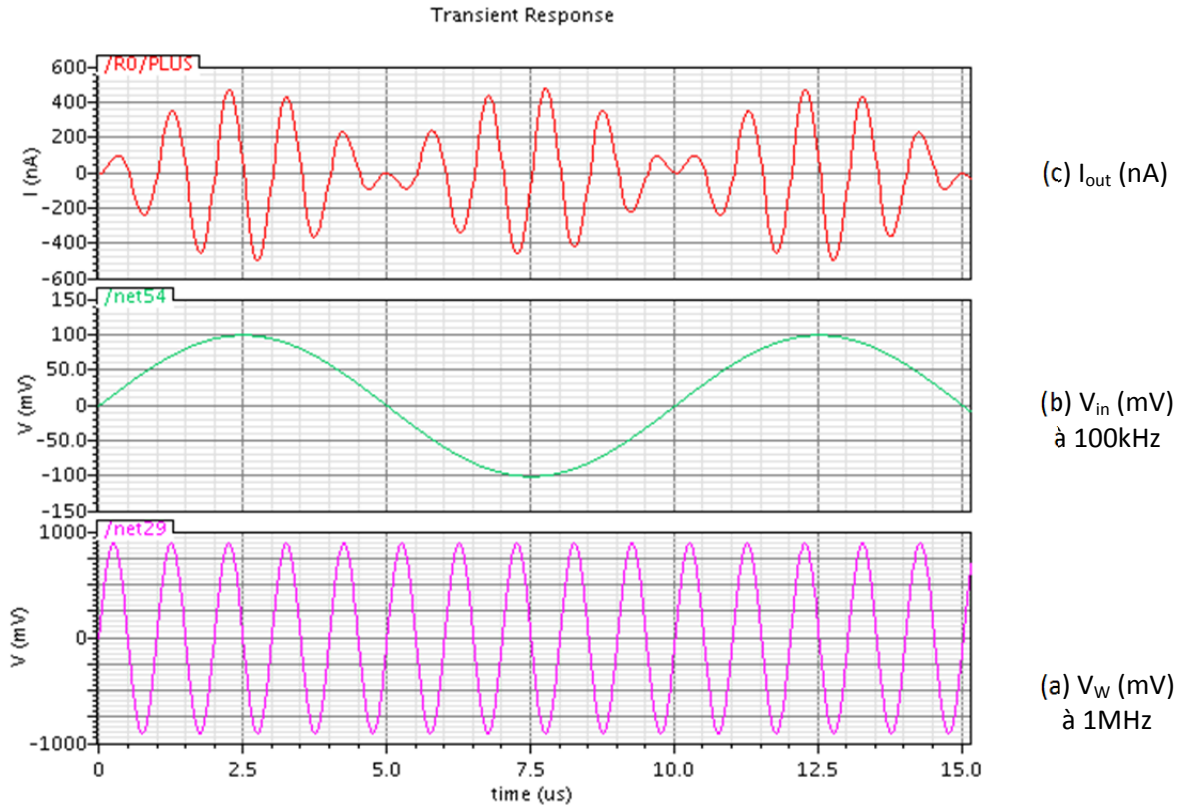
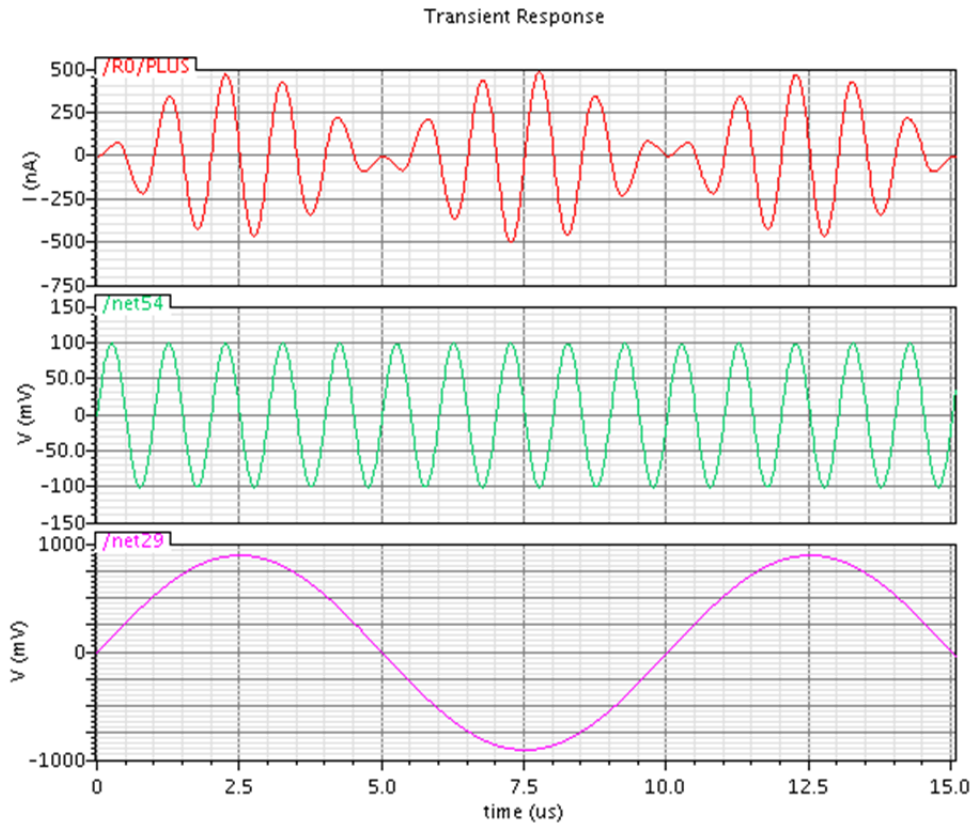


Figure 4.12 : Sortie I_{out} du multiplieur, réponse transitoire pour Vin à 100kHz et V_W à 1MHz

Pour confirmer ce résultat, nous avons calculé, pour les deux configurations, le spectre du signal de sortie, I_{out}. Les résultats, sensiblement identiques, sont présentés sur les figures 4.14 et 4.15. Comme prévu, nous obtenons deux raies à 900kHz et 1,1MHz (produits ou mélanges des deux entrées à 100kHz et 1MHz). Leur amplitude est dans les deux cas d'environ -132dB. Les harmoniques dus aux non-linéarités du multiplieur demeurent inférieures à -170dB, ce qui nous donne une marge de bruit d'envrion 38dB. Cette valeur est là encore largement suffisante pour notre application.

Les différentes simulations transitoires ont montré que la consommation du multiplieur ne dépassait jamais 4µW (environ 3,8µW en fait).



(c) I_{out} (nA)

(b) V_{in} (mV) à 1MHz

(a) V_W (mV) à 100kHz

Figure 4.13 : Sortie I_{out} du multiplieur, réponse transitoire pour Vin à 1MHz et V_W à 100kHz

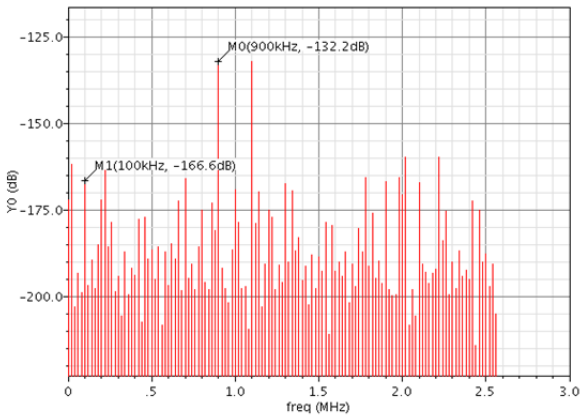


Figure 4.14 : Spectre de I_{out} configuration 1 (Fig 4.12)

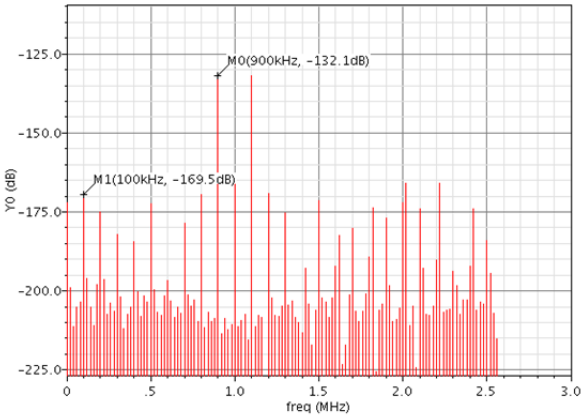


Figure 4.15 : Spectre de I_{out} configuration 2 (Fig. 4.13)

2.4.4. Estimation de la bande passante, analyse AC

L'analyse AC a été effectuée pour différentes valeurs de poids (Vw) et d'entrées Cancer (V_{in}) afin de déterminer le gain du multiplieur, I_{out}/V_{in}, en fonction de la fréquence. Un résultat typique, présenté sur la figure 4.16, montre que le circuit est fonctionnel pour des fréquences allant juqu'à 1 MHz (voir 10MHz, atténuation de l'ordre de 0,2dB seulement), ce qui est plus que suffisant pour l'application visée [1].

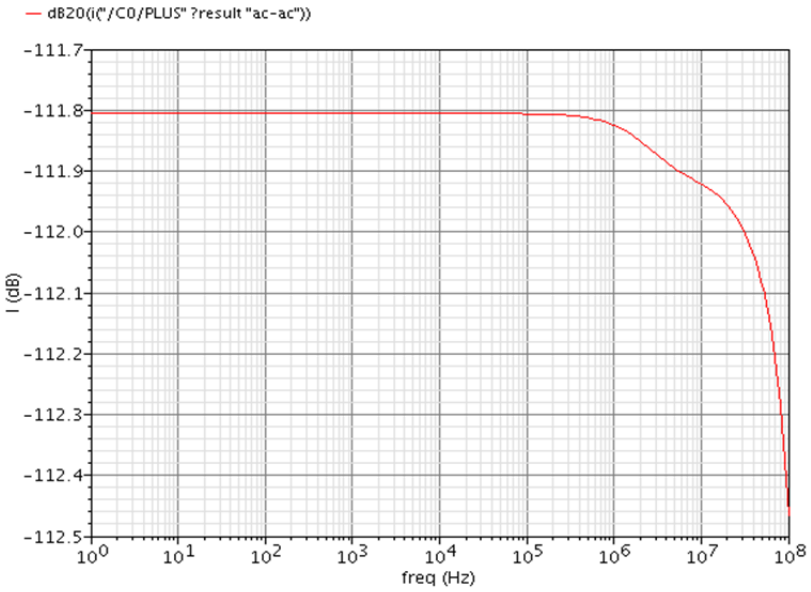


Figure 4.16 : Réponse en fréquence du multiplieur

Lors de cette simulation, pour tracer la figure 4.16, nous avons fixé V_W à 800mV_{DC} et v_{in} à 100mV_{AC}. La fréquence de coupure à -3dB est proche de 100MHz.

2.5. Conclusion

Dans ce paragraphe, nous avons présenté la conception d'un multiplieur répondant à notre cahier des charges. Les différents résultats de simulation ont permis de valider le choix de la topologie ainsi que le dimensionnement des transistors. Il devrait pouvoir fonctionner jusqu'à une fréquence supérieure à 10MHz et une dynamique d'entrée respective de ±100mV et ±800mV (voir ±900mV), ce qui est largemenent suffisant voire au-dessus de nos spécifications.

Le second bloc, très important pour la réalisation d'un neurone puis d'une réseau complet, concerne la fonction d'activation. Il est présenté au paragraphe suivant.

3. Conception de la fonction d'activation et de sa dérivée

3.1. Introduction

La fonction d'activation la plus efficace pour notre application est la fonction logsigmoid, comme il a été démontré au chapitre précédent. Toutefois, il est difficile d'exclure à ce niveau de conception la fonction tangente hyperbolique. Notre étude va donc porter sur ces deux fonctions.

Avant de définir des topologies permettant de les réaliser, revenons sur l'amplificateur de transconductance, en polarisant les transistors en faible inversion. Ainsi le paragraphe suivant est dédié à l'étude d'une paire différentielle où les transistors MOS sont polarisés en faible inversion. Cette méthode permet d'une part d'obtenir un gain important avec une faible consommation et d'autre part de réaliser facilement une fonction tangente hyperbolique entre la sortie et l'entrée de l'amplificateur. En effet, la caractéristique d'un transistor MOS en faible inversion est identique à celle d'un bipolaire, à savoir une loi exponentielle.

Rappelons que la quantité d'inversion est contrôlée par le coefficient d'inversion IC [9]. En principe, n'importe quelle valeur de ce coefficient IC peut être obtenue pour n'importe quelle valeur de courant de saturation en ajustant le courant spécifique I_{spec} à W/L. Les seules limites sont la largeur W du transistor (et la fuite drain-substrat associée) pour I_{spec} très grand, et sa longueur L (et la fuite associée sous le canal) pour I_{spec} très petit.

3.2. Paire différentielle en faible inversion

De nombreux circuits prennent un signal d'entrée représenté comme une différence entre deux tensions. Ces circuits utilisent tous une variante de la paire différentielle représentée sur la figure 4.17 en tant qu'étage d'entrée [10], [11]. Rappelons quelques caractéristiques de ce montage.

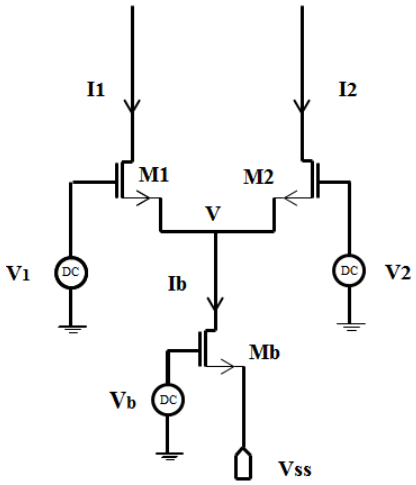


Figure 4.17 : Schéma d'une paire différentielle

Le courant de polarisation I_b est fixé par la tension de polarisation V_b, il est divisé entre I_1 et I_2 en fonction de la différence entre V_1 et V_2. Ainsi, le transistor M_b, utilisé comme une source de courant, est polarisé pour être en régime saturé en forte inversion. Les transistors M_1 et M_2 seront en régime de faible inversion. Par conséquent leur courant sera représenté par une loi exponentielle et non quadratique (cf. équation 4.14).

$$I_D = I_0 e^{\frac{V_G - V_S}{nU_T}} = I_S e^{\frac{2nU_T - V_{T0}}{nU_T}} e^{\frac{V_{GS}}{nU_T}} \quad \text{(rel 4.14)}$$

Où I_S est le courant spécifique, donné par : $\mathrm{I_s} = 2\mathrm{n}\mu\mathrm{C_{ox}}\mathrm{U_T^2}\frac{\mathrm{W}}{\mathrm{L}}$

U_T=kT/q est la tension thermique (26mV), V_{T0} est la tension de seuil à V_{BS}=0.

Le coefficient n, appelé effet de corps, est donné par la relation suivante $n=1+C_D/C_{OX}$, où C_D est la capacité de la couche d'appauvrissement et C_{OX} la capacité de la couche d'oxyde.

Dans le cas de la figure 4.17 où $V_{GS1}=V_1-V$ et $V_{GS2}=V_2-V$, en appliquant la relation 4.14, on obtient pour les courants de drain :

$$I_1 = I_0 e^{\frac{V_1 - V}{nU_T}} \quad \text{et} \quad I_2 = I_0 e^{\frac{V_2 - V}{nU_T}} \quad \text{(rel 4.15)}$$

La somme de ces deux courants nous donne I_b, soit

$$I_1 + I_2 = I_0 \left[e^{\frac{V_1 - V}{nU_T}} + e^{\frac{V_2 - V}{nU_T}} \right] = I_b \quad \text{(rel 4.16)}$$

Le courant de sortie différentielle est quant à lui donné par la relation (4.17) :

$$I_1 - I_2 = I_0 \left[e^{\frac{V_1 - V}{nU_T}} - e^{\frac{V_2 - V}{nU_T}} \right] \quad \text{(rel 4.17)}$$

En effectuant la division entre ces deux équations (rel. 4.16 et 4.17), on obtient la fonction de transfert de cette paire différentielle en régime d'inversion faible (cf. rel 4.18).

$$\frac{I_1 - I_2}{I_b} = \frac{e^{\frac{V_1 - V}{nU_T}} - e^{\frac{V_2 - V}{nU_T}}}{e^{\frac{V_1 - V}{nU_T}} + e^{\frac{V_2 - V}{nU_T}}} = tanh\left(\frac{V_1 - V_2}{2nU_T}\right)$$

$$I_{diff} = I_1 - I_2 = I_b tanh\left(\frac{V_1 - V_2}{2nU_T}\right) \quad \text{(rel 4.18)}$$

La fonction de transfert I_{diff}/I_b en fonction de $v_{in}=V_1-V_2$ est ainsi illustrée par la figure 4.18. Cette fonction de type tangente hyperbolique présente des caractéristiques très intéressantes [12-18]. Elle passe par l'origine avec une pente unité et tend vers 1 pour une entrée positive et -1 pour une entrée négative. En outre, le point d'inflexion à l'origine (pour les faibles variations de la tension d'entrée différentielle) offre une plus grande plage de linéarité en comparaison d'un montage à un seul transistor de type Source Commune.

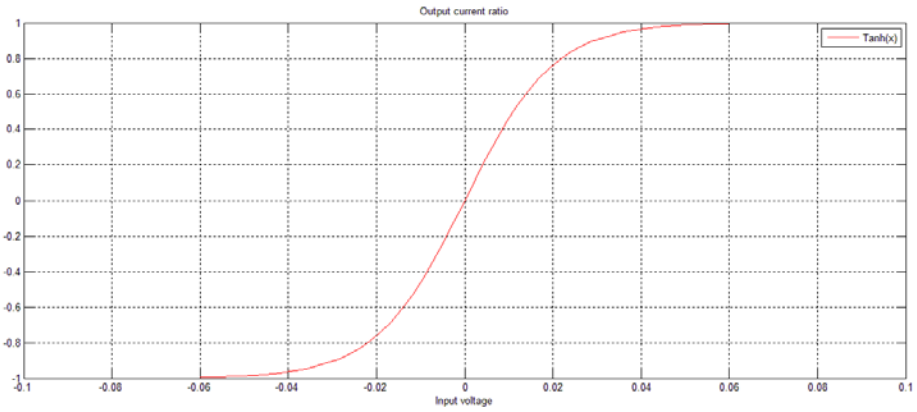
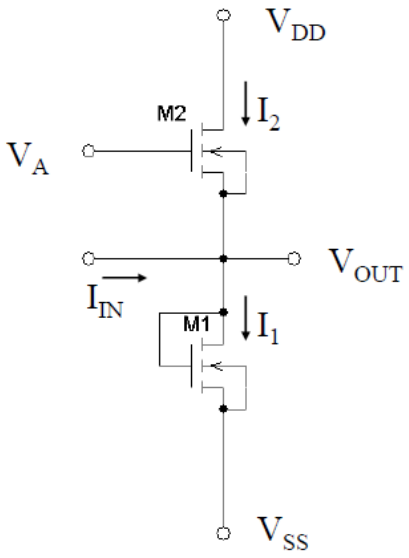


Figure 4.18 : Fonction de transfert de type tangente hyperbolique d'une paire différentielle

Avant de concevoir la fonction d'activation de notre circuit final (neurone), étudions dans un second temps une autre fonction de base de l'électronique analogique, à savoir un convertissseur courant-tension linéaire. Ce circuit sera ensuite implémenté pour réaliser la fonction d'activation.

3.3. Convertisseur courant-tension linéaire

K. Bult et H. Wallinga [19] ont proposé un convertisseur courant-tension linéaire, qui n'avait que deux transistors, comme le montre la figure 4.19. Les deux transistors NMOS, M_1 et M_2 sont polarisés en saturation. Pour éliminer l'effet de corps sur la tension de seuil V_{TH}, les bulks des deux transistors sont connectés à leurs sources.



Les transistors sont de tailles identiques

$W_1=W_2=W$ et $L_1=L_2=L$.

Ainsi $\beta = \frac{1}{2} KP_n \frac{W}{L}$

$V_{GS1}=V_{OUT}-V_{SS}$ et $V_{GS2}=V_A-V_{OUT}$

$$I_1=\beta(V_{out}-V_{SS}-V_{TH})^2 \qquad \text{(rel 4.19)}$$

$$I_2=\beta(V_A-V_{OUT}-V_{TH})^2 \qquad \text{(rel 4.20)}$$

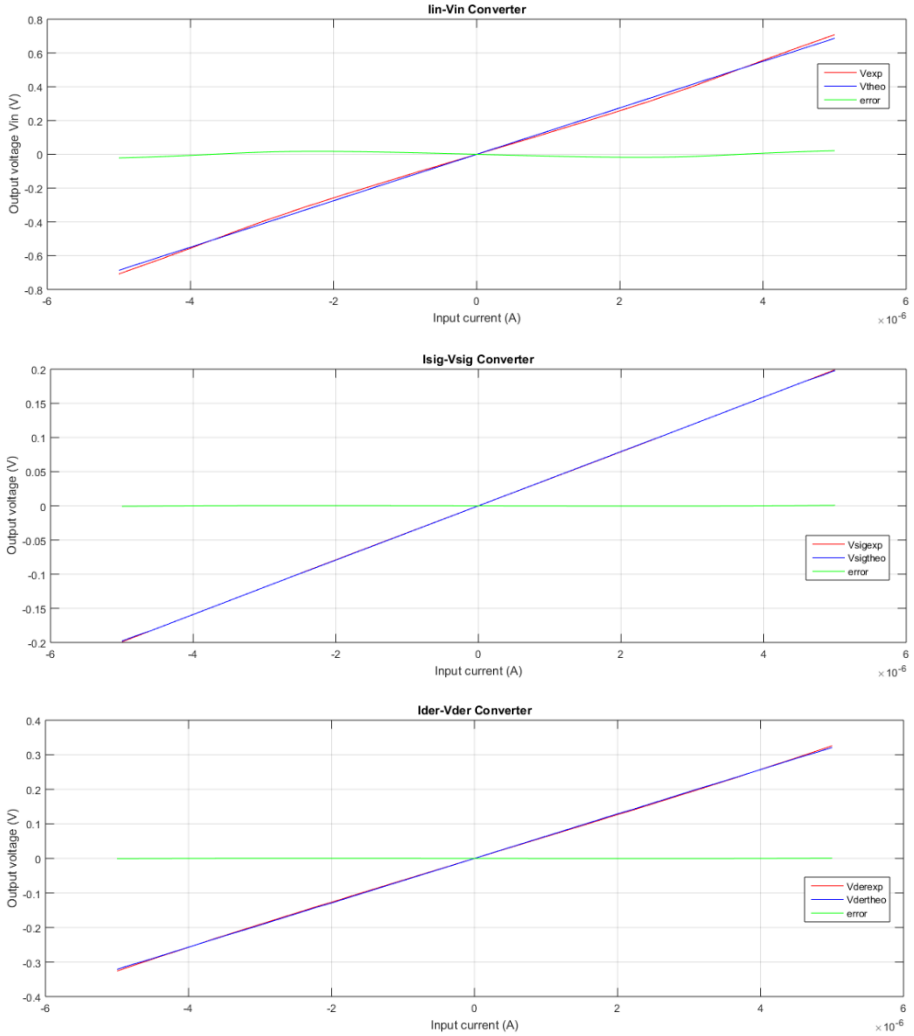
Figure 4.19 : Schéma à 2 transistors du convertisseur courant-tension linéaire

Or $I_{IN}=I_1-I_2$, donc I_{IN} peut être écrit par la relation 4.21 en faisant la différence des deux relations précédentes (4.19 et 4.20). Si on pose $V_A=V_{DD}$ (court-circuit entre la grille et le drain de M_2, charge active à l'instar de M_1), on en déduit la valeur de la tension de sortie en fonction de I_{IN}.

$$I_{IN} = \beta(V_A - V_{SS} - 2V_{TH})(2V_{OUT} - V_{SS} - V_A) \quad \text{(rel 4.21)}$$

Et finalement : $$V_{OUT} = \frac{I_{IN}}{\beta(V_A - V_{SS} - 2V_{TH})} \quad \text{(rel 4.22)}$$

En utilisant toujours la technologie HCMOS9A, nous avons simulé ce circuit pour 3 rapports W/L différents de taille de transistor. Les résultats sont présentés sur la figure 4.20 avec respectivement W/L=290nm/7µm (configuration appelée V_{in}), 150nm/950nm (configuration V_{sig}) et 407nm/4,97µm (V_{der}). Lors des simulations, nous avons là encore choisi V_{DD}=900mV et V_{SS}=-900mV. Pour être compatible avec la sortie du multiplieur présenté au paragraphe 2 de ce chapitre, la dynamique d'entrée I_{IN} est de ±5µA (une dynamique de ±4µA est en fait suffisante).



Configuration V_{in}

W=290nm

L=7µm

Configuration V_{sig}

W=150nm

L=950nm

Configuration V_{der}

W=407nm

L=4,97µm

Figure 4.20 : Caractéristique du convertisseur courant-tension pour 3 tailles de transistor

Sur les trois courbes de la figure 4.20, nous avons également tracé la droite théorique à partir de l'équation (4.22). Dans la pratique, différentes tailles W/L de transistors seront utilisées suivant la fonction d'activation retenue (tangente hyperbolique ou logsigmoid), afin de s'adapter à la charge.

Notons (cf. Figure 4.20) que cela n'a que peu d'influence sur l'erreur, écart entre la sortie réelle et la droite théorique.

3.4. Conception de la fonction d'activation

3.4.1. Fonction d'activation tangente hyperbolique

Dans les réseaux neuronaux à action directe, leur réponse est généralement une fonction non linéaire de l'entrée pondéré, réalisée en sommant les courants de sortie des synapses (sur un nœud de sommation). La fonction non linéaire doit être réalisée par le circuit de la fonction d'activation. De plus, pour les réseaux de neurones avec apprentissage sur puce entièrement analogiques utilisant la rétropropagation, la dérivée de la fonction non linéaire par rapport à l'entrée pondérée est requise [13]. Les types les plus courants de fonctions d'activation sont:

- Linéaire [19] (non retenue)
- Sigmoïde, logistique ou logsigmoïde (logsigmoid) [2], [11]
- Tangente hyperbolique |12-18]

La fonction non linéaire la plus largement utilisée dans les réseaux de neurones à action directe est probablement la fonction tangente hyperbolique. Elle est est constituée de deux convertisseurs courant-tension (I-V converter) et du circuit de fonction non linéaire. Deux convertisseurs I-V sont utilisés pour convertir le courant d'entrée et le courant de sortie en tension d'entrée et en tension de sortie. Un tel circuit est illustré par la figure 4.21.

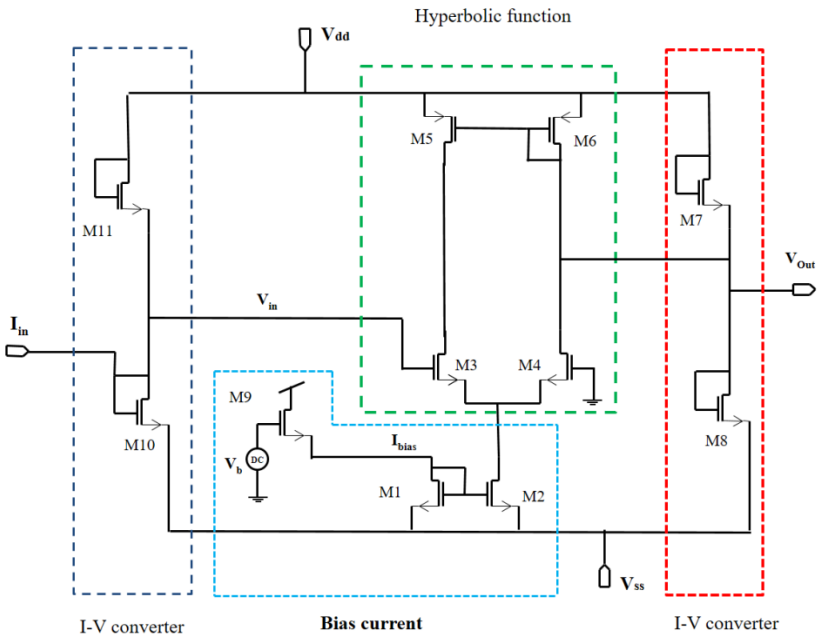


Figure 4.21 : Fonction d'activation de type tangente hyperbolique

L'entrée I_{in} correspond au courant de sortie (I_{out}) du multiplieur (cf. Fig. 4.7), il est ensuite transformé en tension pour attaquer la paire différentielle en faible inversion (cf. Fig. 4 .17). L'entrée V_2 de cet étage est mise à la masse et l'autre entrée correspond à la sortie du convertisseur I-V d'entrée. Pour réaliser le neurone final, nous dirons que le courant de sortie du multiplieur a_j est connecté à l'entrée de la fonction d'activation dans la couche cachée. La sortie de la fonction d'activation X_j est connectée à l'entrée du multiplieur de la couche de sortie, et la sortie de ce multiplieur a_k est connectée à l'entrée de la fonction d'activation de la couche de sortie. Ce point sera repris au chapitre 5.

Le tableau 4.4 donne les valeurs des tailles des transistors de la figure 4.21. Le résultat de simulation Spice sous Cadence de ce circuit, fonction de transfert, est présenté sur la figure 4.22.

Tableau 4.4 : Dimensions des transistors de la fonction d'activation tangente hyperbolique

Transistor	Type	Largeur (nm)	Longueur (µm)
M1	NMOS	240	5,4
M2	NMOS	275	6,1
M3	NMOS	2000	1
M4	NMOS	2000	1
M5	NMOS	300	3
M6	PMOS	300	3
M7	PMOS	150	5
M8	PMOS	150	5
M9	NMOS	230	6
M10	NMOS	150	5
M11	NMOS	150	5

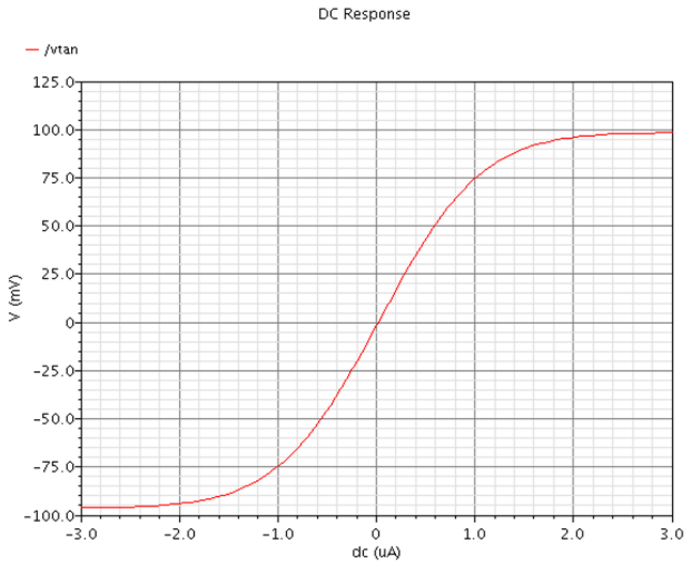


Figure 4.22 : Fonction de transfert de la fonction d'activation de type tangente hyperbolique

On constate sur la figure 4.22 que l'on respecte la dynamique de sortie à savoir ±100mV, tension qui sera réinjectée sur le circuit dérivée, décrit au paragraphe suivant (i.e paragraphe 3.4.2) avec toutefois une légère dissymétrie. La dynamique d'entrée est également respectée, puisque nous saturons à ±100mV avant les valeurs extrèmes de courant d'entrée (i.e. les valeurs extrèmes du courant de sortie (ou dynamique de sortie) du multiplieur : ±3,8µA. La saturation est atteinte à I_{IN}=±3µA. Concernant la dissymétrie observée, elle est de 5mV lorsque la sortie vaut -100mV (-95mV en réalité pour une entrée de -3µA, alors qu'elle est bien de 100mV pour une entrée de 3µA). Cette erreur d'environ 5% (valeur extrème) est bien inférieure aux 10% requis par le cahier des charges.

3.4.2. Circuit dérivée de la fonction tangente hyperbolique

Les réseaux neuronaux d'apprentissage sur puce nécessitent des circuits pour la dérivée de la fonction d'activation non linéaire pour le calcul d'erreur entre la sortie réelle et la sortie désirée. Dans le cas d'une fonction d'activation de type tangente hyperbolique y=tanh(x), sa dérivée est égale à $y'= 1-x^2$. La cellule de Gilbert [5] est utilisée comme circuit de base pour effectuer cette opération, comme le montre la figure 4.23. Les dimensions des transistors, de cette figure, sont résumés sur le tableau 4.5.

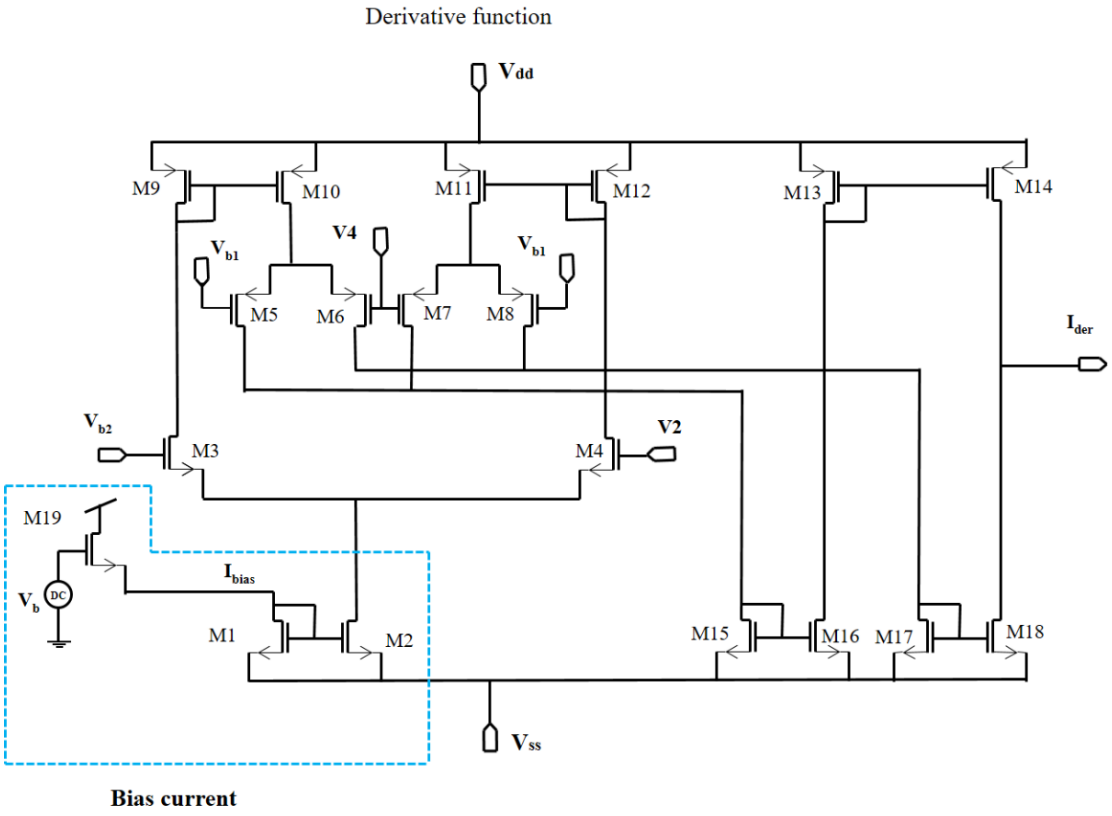


Figure 4.23 : Electronique réalisant le circuit dérivée à base d'une cellule de Gilbert

Le courant de sortie d'une telle cellule de Gilbert (cf. Figure 4.23) est donné par l'équation (4.23) où I_B est le courant de polarisation et V_{b2}, V_2, V_{b1} et V_4 sont les entrées respectives. En outre, en raison des faibles valeurs des tensions d'entrée (<100mV), nous pouvons écrire tanh(θ)#θ.

$$I_{OUT} = I_b tanh\left(\frac{V_{b2}-V_2}{2nU_T}\right).tanh\left(\frac{V_{b1}-V_4}{2nU_T}\right) \# I_b \frac{(V_{b2}-V_2)(V_{b1}-V_4)}{4n^2U_T^2} \quad \text{(rel 4.23)}$$

Si on pose V_{b1}=-V_4=V_r et V_2=-V_{b2}=V_{in}, alors on obtient la relation suivante :

$$I_{OUT} = I_b\left(\frac{V_r-V_{in}}{2nU_T}\right)\left(\frac{V_r+V_{in}}{2nU_T}\right) \quad \text{(rel 4.24)}$$

On obtient finalement l'équation désirée, dérivée de la fonction tangente hyperbolique, comme le montre l'équation (4.25).

$$I_{OUT} = \frac{I_b V_r^2}{4n^2U_T^2}\left(1-\left(\frac{V_{in}}{V_r}\right)^2\right) \quad \text{(rel 4.25)}$$

Tableau 4.5 : Dimensions des transistors de la fonction dérivée de la tangente hyperbolique

Transistor	Type	Largeur (nm)	Longueur (µm)
M1	NMOS	300	1,08
M2	NMOS	300	1,08
M3	NMOS	1000	5,0
M4	NMOS	1000	5,0
M5	NMOS	1000	0,5
M6	PMOS	1000	0,5
M7	PMOS	1000	0,5
M8	PMOS	1000	0,5
M9	NMOS	1000	8,84
M10	NMOS	1000	8,84
M11	NMOS	1000	8,84
M12	NMOS	1000	8,84
M13	PMOS	1000	8,84
M14	PMOS	1000	8,84
M15	PMOS	500	10,0
M16	PMOS	500	10,0
M17	NMOS	500	10,0
M18	NMOS	500	10,0
M19	NMOS	150	6,0

La figure 4.24 présente les résultats de simulation Spice (sous Cadence) de ce circuit en technologie HCMOS9A 130 nm. La dynamique d'entrée de ±100mV est respectée, et le circuit offre en sortie une dynamique en courant de 25 nA.

Nous pouvons passer à l'étude de la fonction d'activation logsigmoid.

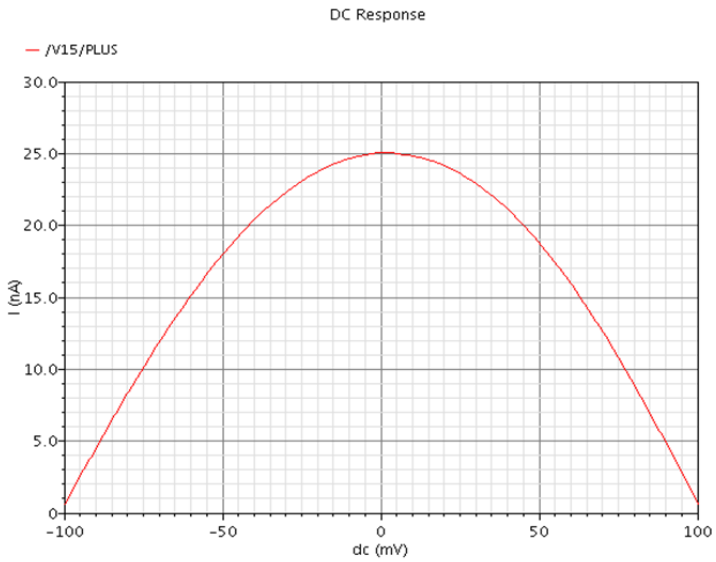


Figure 4.24 : Fonction de transfert du circuit dérivée de la tangente hyperbolique

3.4.3. Fonction d'activation de type logsigmoid

La seconde fonction non linéaire la plus largement utilisée dans les réseaux de neurones à action directe est la fonction sigmoïde ou logsigmoid. Cette fonction est approchée en utilisant là encore une paire différentielle bipolaire (ou MOS en faible inversion) avec une entrée en tension et une sortie en courant. Comme dans le cas de la fonction tangente hyperbolique, la fonction d'activation utilise deux convertisseurs I-V (courant-tension) pour convertir le courant d'entrée (i.e. de sortie du multiplieur) en tension pour le circuit « logsigmoid » lui-même, ainsi que pour son courant de sortie, comme le montre la figure 4.25.

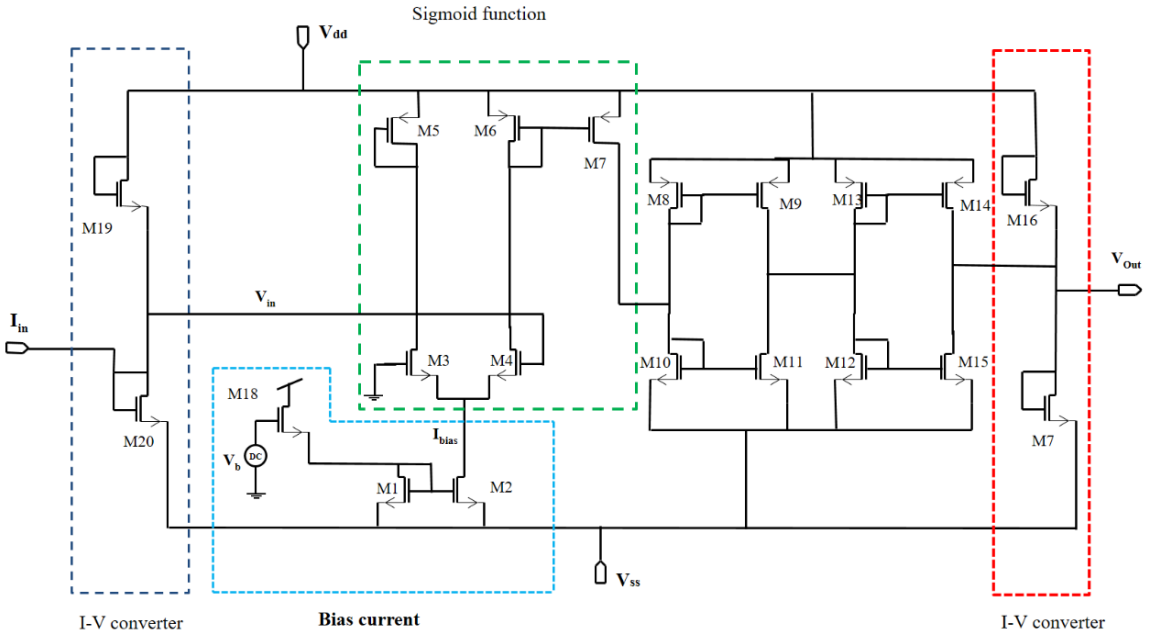


Figure 4.25 : Fonction d'activation de type logsigmoid

Le tableau 4 .6 résume les dimensions, après optimisation (i.e. rapports W / L), des transistors utilisés dans le circuit représenté sur la figure 4.25. Le résultat de simulation Spice sous Cadence de ce circuit, fonction de transfert, est présenté sur la figure 4.26.

Tableau 4.6 : Dimensions des transistors de la fonction d'activation logsigmoid

Transistor	W/L (en nm/µm)	Transistor	W/L (en nm/µm)
M1	240/5,4	M11	150/1,5
M2	275/6,2	M12	150/1,5
M3	250/6,0	M13	1000/1,0
M4	280/5,1	M14	1000/1,1
M5	300/2,0	M15	150/1,5
M6	300/2,0	M16	150/1,1
M7	600/1,6	M17	150/1,1
M8	1000/1,0	M18	230/6,0
M9	1000/1,1	M19	150/10,0
M10	150/1,5	M20	150/10,0

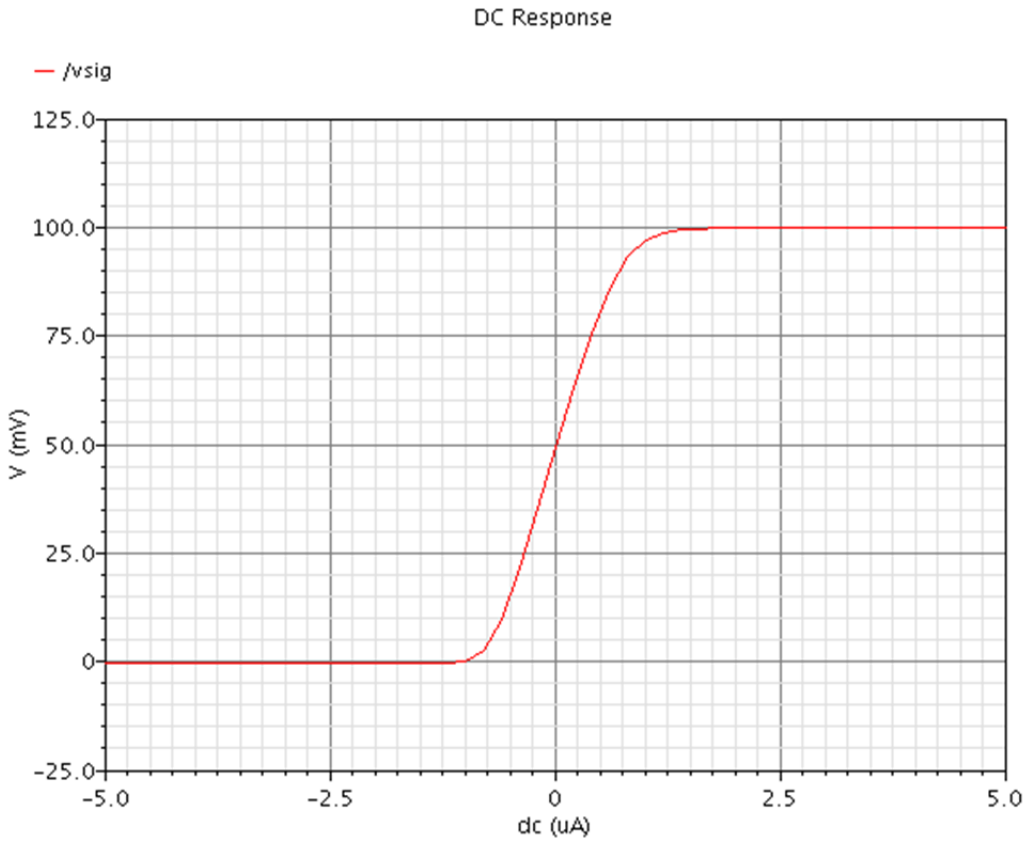


Figure 4.26 : Fonction de transfert de la fonction d'activation logsigmoid

On constate sur la figure 4.26 que l'on respecte la dynamique de sortie à savoir de 0 à 100mV, tension qui sera réinjectée sur le circuit dérivée, décrit au paragrpahe suivant (3.4.4). La dynamique d'entrée est également respectée, puisque nous saturons à 0 ou 100mV avant des valeurs de courant de l'ordre de ±1,5µA, bien inférieures à la dynamique de sortie du convertisseur I-V qui est de ±4µA.

Le dernier circuit, étudié dans ce chapitre, concerne la dérivée de la fonction d'activation logsigmoid. Elle est décrite au paragraphe suivant.

3.4.4. Circuit dérivée de la fonction logsigmoid

Comme dans le cas de la fonction d'activation tangente hyperbolique, le circuit dérivée est basé sur une cellule de Gilbert [5], comme le montre la figure 4.27. Toutefois, pour s'adapter à cette nouvelle fonction, nous devons modifier les tensions de la façon suivante (cf. Tableau 4.7) :

Tableau 4.7 : Tableau de correspondance entre fonctions « dérivée » de la fonction d'activation

Tension (V)	**Dérivée de la fonction tangente hyperbolique**	**Dérivée de la fonction logsigmoid**
V_b	0,4	0,2
V_{b1}	-0,1	0
V_{b2}	0,1	0,1
V_2	V_{hyp}	V_{sig}
V_4	V_{hyp}	V_{sig}

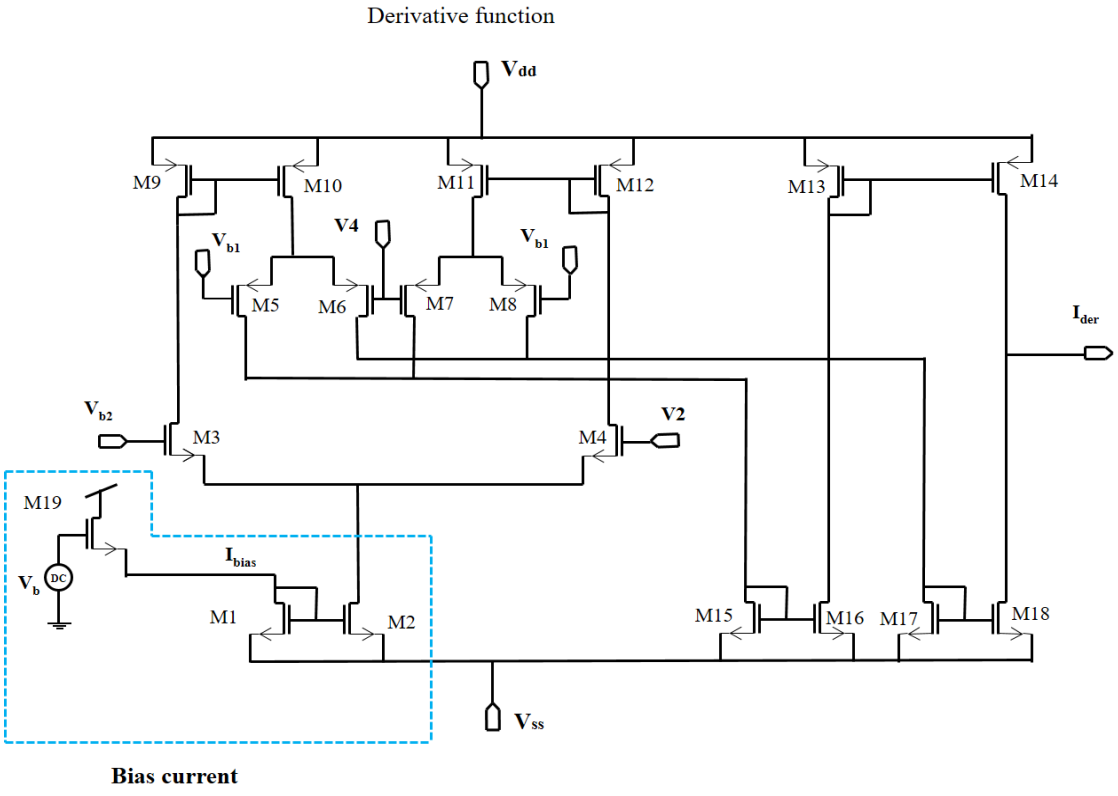


Figure 4.27 : Circuit « dérivée » de la fonction d'activation de type logsigmoid

Le tableau 4 .8 résume les dimensions, après optimisation (i.e. rapports W / L), des transistors utilisés dans le circuit représenté sur la figure 4.27. Le résultat de simulation Spice sous Cadence de ce circuit, fonction de transfert, est présenté sur la figure 4.28.

Tableau 4.8 : Dimensions des transistors du circuit « dérivée » de la fonction d'activation logsigmoid

Transistor	W/L (en nm/µm)	Transistor	W/L (en nm/µm)
M1	300/1,08	M11	1000/8,84
M2	300/1,08	M12	1000/8,84
M3	1000/5,0	M13	1000/8,84
M4	1000/5,0	M14	1000/8,84
M5	1000/500,0	M15	500/1,0
M6	1000/500,0	M16	500/1,0
M7	1000/500,0	M17	500/1,0
M8	1000/500,0	M18	500/1,0
M9	1000/8,84	M19	150/6,0
M10	1000/8,84		

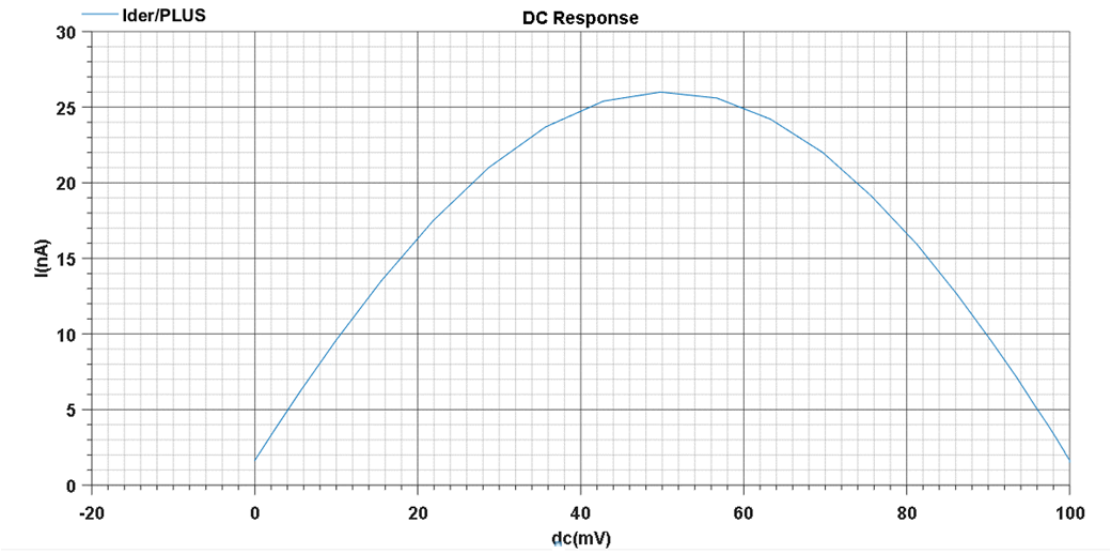


Figure 4.28 : Fonction de transfert de la dérivée de la fonction d'activation logsigmoid

La figure 4.28 montre que le crcuit « dérivée » de la fonction d'activation logsigmoid respecte la dynamique d'entrée, à savoir 0 à 100 mV. Comme dans le cas de la dérivée de la fonction tangente hyperbolique, le circuit offre en sortie une dynamique de 25nA.

Les consommations respectives pour les fonctions d'activation et leurs dérivées sont de 7,8µW et de 230nW pour la fonction tangente hyperbolique et de 10,8µW et de 252nW pour la fonction logsigmoid.

3.4.5. Conclusion

La fonction d'activation d'un neurone est une fonction non-liénaire, son implémentation est très importante. Elle nous permet de représenter une correspondance non-linéaire complexe entre les entrées et les sorties d'un réseau de neurone. Dans ce paragraphe, nous avons conçu et optimisé

les fonctions tangente hyperbolique et logsigmoid, ainsi que leurs dérivées respectives. Dans les deux cas, les resultats de simulation montre que le cahier des charges est respectés.

4. Conclusion

Dans ce chapitre, nous avons présenté la conception des briques de base analogiques, réalisés en technologie CMOS 130nm, d'un neurone artificiel, à savoir le multiplieur, la fonction d'activation et sa dérivée. La sortie du neurone étant représenté sous forme de courant, nous n'avons pas besoin d'implémenter un additionneur. Cette opération sera réalisée en utilisant simplement la loi aux nœuds (loi de Kirchhoff), optimisant ainsi la consommation finale du réseau de neurones complet. Les circuits ont été optimisés, et donc dimensionnés au niveau transistor, afin de respecter notre cahier des charges et d'avoir une consommation la plus faible possible.

Le dernier chapitre décrit l'implémentation finale du réseau de neurones, utilisant ces briques de base, en ajoutant la mémorisation et la mise à jour des coefficients. Rappelons que ce réseau est basé sur une architecture multicouche perceptron avec un algorithme de rétro-propagation pour la phase d'apprentissage.

5. Bibliographie

[1] H. Jouni, M. Issa Mariam, A. Harb, G. Jacquemod, Y. Leduc, "Neural Network architecture for breast cancer detection and classification," in *2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, Beirut, 2016.

[2] C.A.Mead, M. Ismail, "Analog VLSI Implementation of Neural Systems," in *Boston:Kluwer Academic Publisher*, Boston, 1989.

[3] G. Han, S. Edgar, "CMOS Transconductance Multipliers: A Tutorial," *IEEE Trans. On Circuits and Systems - II,* vol. 45, no. 12, pp. 1550-1563, 1998.

[4] E.A.Vittoz, "Analog VLSI signals processing:Why, Where and How?," *Journal of VLSI Signal Processing,* vol. 8, pp. 27-44, 1994.

[5] S. Panda, S. Srimani, B. Maji, and A. K. Mukhopadhysay, "High Performance CMOS Four Quadrant Analog Multiplier in 45 nm Technology," *International Journal of Application or Innovation in Engineering & Management, vol. 2, no. 7, 2013.,* Vols. 2,no.7, 2013.

[6] M. Dei, N. Nizza, P. Bruschi, M. Pioto, "A Four Quadrant CMOS Analog Multiplier Based on the Non Ideal MOSFET I-V Charactersitics," in *IEEE Conference Publications*, 2008.

[7] C. Sawigun, J. Mahattanakul, "A 1.5 V,Wide Input range, high bandwidth, CMOS four-quadrant analog multiplier," *Proc. of the IEEE International Symposium Circuits and Systems,* pp. 2318-2321, 2008.

[8] H. Chible, H. Jouni, "Analog Multiplier For Feed Forward Neural Network Signal Processing," in *Proceeding of the 7th IASTED International Conference Signal Processing,Pattern Recognition and Applications*, Innsbruck,Austria, 2010.

[9] D. Colombo, C. Fayomi, F. Nabki, L.F. Ferreira, G. Wirth, S. Bampi, «A Design Methodology Using the Inversion Coefficient for Low-Voltage Low-Power CMOS Voltage References,» Journal of Integrated Circuits and Systems , vol. 6, n° %11, pp. 7-17, August 2011.

[10] «Analog VLSI Signal Processing:Why, Where and How?,» Journal of VLSI Signal Processing,and in Analog Integrated Circuits and Signal Processing, vol. 8, pp. 27-44, July 1994.

[11] R.C. Chang, B.J. Sheu, J. Choi, D. Cheng-Hsiung Chen, «Programmable-Weight Building Blocks for Analog VLSI Neural Network Processors,» nalog Integrated Circuits and Signal Processing, vol. 9, pp. 215-230, 1996.

[12] Madhumitha G.B., Vijayalatha Devadiga, «Analog VLSI Implementation of Artificial Neural Network,» International Journal of Innovative Research in Computer and Communication Engineering, vol. 3, n° %15, pp. 72-80, May 2015.

[13] Cyril Prasanna Raj, S.L. Pinjare, «Design and Analog VLSI Implementation of Neural Network Architecture for Signal Processing,» European Journal of Scientific Research, vol. 27, n° %12, pp. 199-216, 2009.

[14] S. R. Kshirsagar, A. O. Vyas, «Signal Processing in Neural Network using VLSI Implementation,» International Journal Of Engineering And Computer Science, vol. 2, n° %16, pp. 2086-2090, June 2013.

[15] Bapuray.D. Yammenavar, Vadiraj.R. Gurunaik, Rakesh.N. Bevinagidad, Vinayak.U. Gandage, «DESIGN AND ANALOG VLSI IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORK,» International Journal of Artificial Intelligence & Applications (IJAIA), vol. 2, n° %13, pp. 96-109, July 2011.

[16] Neeraj Chasta, Sarita Chouhan,Yogesh Kumar, «Analog VLSI Implementation of Neural Network Architecture for Signal Processing,» International Journal of VLSI design & Communication Systems (VLSICS), vol. 3, n° %12, pp. 243-259, April 2012.

[17] L. Gatet, H. Tap-Beteille, M. Lescure, D. Roviras, A. Mallet, «Functional tests of a 0.6 lm CMOS MLP analog neural network for fast on-board signal processing,» Analog Integr Circ Sig Process , vol. 54, p. 219–227, 2008.

[18] Rachana R. Patil, Dinkar L. Bhombe, «Review: “Implementation of Feedforward and Feedback Neural Network for Signal Processing Using Analog VLSI Technology”,» Journal of Engineering Research and Applications, vol. 5, n° 11(Part 5), pp. 115-119, January 2015.

[19] R. Wojtyna, T. Talaska, «Transresistance CMOS neuron for adaptive neural networks implemented in hardware,» Bulletin of the Polish Academy of Sciences and Technical Sciences, vol. 54, n° 14, pp. 443-448, 2006.

[20] I. Bayraktaroglu, A. Selc¸ U. Ogrenci, G. Dundar, S. Balkır, E. Alpaydın, «ANNSyS: an Analog Neural Network Synthesis System,» Neural Networks, vol. 12, pp. 325-338, 1999.

Chapitre V – Conception du réseau de neurones

1. Introduction

Dans les chapitres précédents nous avons retenu l'algorithme de classification, défini l'architecture du RNA et les spécifications des blocs de base. Enfin au quatrième chapitre, nous avons donné les schémas à transistors dimensionnés de ces blocs de base et avons vérifié que les spécifications étaient atteintes. La dernière étape consiste à mettre en œuvre un réseau de neurones à partir de ces différents circuits. Dans un premier temps, nous rappellerons l'architecture de notre RNA (i.e. agencement des blocs de base), puis nous proposerons une électronique de sauvegarde des poids de chaque neurone. Enfin nous pourrons définir le circuit final réalisant ce réseau de neurones artificiels.

Rappelons que l'objectif de ce travail est de réaliser une architecture de réseau neuronal artificiel en plateforme VLSI analogique pour classer le cancer du sein comme malin ou bénin [1]. Le cœur de cette étude est la mise en œuvre de l'architecture de réseau de neurones analogiques avec apprentissage sur puce en VLSI analogique [2]. Les composants analogiques de base de l'architecture du réseau de neurones artificiels sont :

- Multiplieur : deux types de multiplieurs ont été étudiés, le multiplieur à quatre quadrants à grande dynamique d'entrée [3] et le multiplieur de Gilbert [4], [5].
- Bloc d'addition : somme des sorties des multiplieurs sous forme de courant, basée sur la loi de Kirchhoff en courant.
- Fonction d'activation et sa dérivée : dans ce cas, nous utilisons la tangente hyperbolique ou la fonction sigmoïde (ou logsigmoid) basée sur la paire différentielle.
- Mémorisation et mise à jour des coefficients : ces étapes seront décrites dans la suite de ce chapitre.

Par conséquent, l'architecture du réseau neuronal est une conception analogique complète. La performance de cette structure est vérifiée pour l'application analogique comme la détection et la classification du cancer du sein. Dans cette conception, l'accent est mis sur la mise en œuvre de la conception de puces pour le réseau de neurones à propagation avant avec un algorithme de propagation inverse en VLSI analogique. Le circuit MLP analogique possède 9 entrées, 10 neurones cachés et 2 neurones de sortie. Le circuit a été conçu et alimenté par une alimentation égale à ±0,9 V.

Rappelons que dans les chapitres précédents, nous avons déterminé les dynamiques d'entrées et de sortie des neurones de la façon suivante :

- Entrées du multiplieur : tensions de ±100mV pour l'entrée « Cancer » (i.e. attributs de la base de données Wisconsin, cf. Tableau 1.3 du chapitre 1) et de ±900mV pour les coefficients (poids des neurones)
- Sortie du multiplieur : courant de ±4µV
- Entrée de la fonction d'activation (tangente hyperbolique ou logsigmoid) : courant de ±4µV
- Sortie de la fonction d'activation : tension de ±100mV
- Circuit dérivée de la fonction d'activation : en entrée tension de ±100mV et en sortie courant de 25nA maximum.

2. Mémorisation des poids analogiques

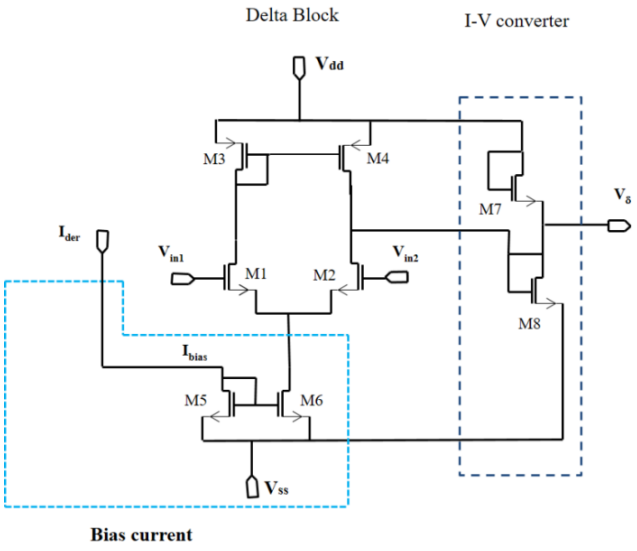
2.1. Introduction

Le réseau neuronal artificiel apprend à partir des modèles (données d'entrée et de sortie) obtenus à partir de la base de données de l'application souhaitée. Dans notre cas, la détection et la classification du cancer du sein a été choisie et l'algorithme le plus approprié est le perceptron MLP avec rétro-propagation [6] , [7], [8], [9]. Le premier réseau de neurones perceptron multicouche a été créé par Hopfield et Tank en 1986 [10]. Dans cet algorithme, l'ajustement des poids est fait en comparant la sortie du neurone avec la cible de chaque entrée. Les poids sont continuellement ajustés jusqu'à ce que la différence entre la sortie du neurone et la cible soit minimale. Cette erreur est renvoyée pour mettre à jour les poids et les éléments de biais. Cet algorithme a été développé de manière continue et la mise en œuvre de divers circuits VLSI analogiques du réseau de neurones perceptron multicouche a été réalisée [4], [11], [12], [13].

La mise à jour des poids (ou coefficients W_{ij}) est réalisée en utilisant un circuit nommé **Bloc Delta** dont le rôle est le suivant. Il convient dans un premier temps de déterminer la multiplication du courant dérivé I_D, appelé δ_k, de la fonction d'activation du neurone de sortie avec l'erreur en Volt. Cette erreur est la différence entre la tension à la sortie du neurone et la tension cible, X_T-X_k.

Ainsi, l'erreur de propagation inverse dans la couche cachée est donnée par la relation suivante : $\delta_j = \sum_{j=1}^{10} \delta_k W_{kj}$. Rappelons que ce mécanisme de mise à jour des coefficients a été décrit, en partie, au chapitre 2 (paragraphe 9), Il sera repris dans le paragraphe 3 de ce chapitre décrivant sa mise en œuvre.

La figure 5.1 illustre le circuit « dimensionné » nommé **Bloc Delta** permettant de calculer ces différences. Les transistors sont polarisés sous le seuil pour avoir une relation exponentielle. La sortie V_δ est alors fonction de la différence des deux entrées V_{in1} et V_{in2}. En toute rigueur, cette relation est là encore une tangente hyperbolique. Toutefois, l'erreur (V_{in1}- V_{in2}) étant très faible (inférieure à notre critère d'arrêt), on peut écrire tanh(V_{in1}- V_{in2})#V_{in1}- V_{in2}.



Dimension des transistors

Transistor	W/L (en nm/µm)
M1	290/1,8
M2	290/1,8
M3	350/0,65
M4	350/0,65
M5	290/1,8
M6	290/1,8
M7	150/5,0
M8	150/5,0

Figure 5.1 : Conception du circuit Bloc Delta

2.2. Stockage des coefficients

La façon la plus simple de mémoriser une tension analogique est d'utiliser un condensateur. La figure 5.2 représente l'initialisation et la procédure de mise à jour de ces poids. Pour chaque synapse ou poids, deux condensateurs de 2pF, C_W et C_{Wu}, sont utilisés pour mémoriser et mettre à jour le poids respectivement.

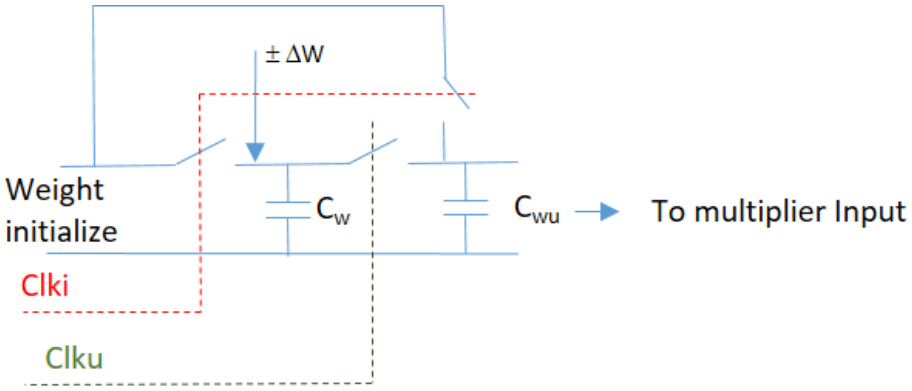


Figure 5.2 : Mécanisme de mémorisation et changement des poids

Sur cette figure, le signal d'horloge, C_{lki}, est utilisé pour initialiser les poids avant de commencer la phase d'apprentissage, i.e. détermination des valeurs finales de ces poids (W_{ij}). L'horloge C_{lku} est utilisée pour contrôler le début de la phase d'apprentissage et le test (classification). Durant le processus d'apprentissage, le signal d'horloge est élevé tandis que durant la phase de test, ce signal est plus faible.

3. Construction du RNA

3.1. Mise à jour des poids

Pour ajuster ou mettre à jour les poids, la méthode d'apprentissage supervisé est utilisée. Ce type d'apprentissage est un ajustement de l'erreur qui se propage vers le neurone de sortie, puis vers le neurone caché, respectivement. Rappelons que notre architecture de réseau neuronal propagation directe est constituée de 9 neurones en entrée, 10 dans la couche cachée et 2 en sortie (9-10-2) avec l'algorithme Propagation-Inverse. Ainsi, les 9 entrées X_{in1}, X_{in2} ... X_{in9} sont connectées aux 10 neurones de la couche cachée à travers 90 poids W^1_{ij} où i = 1 à 9 et j = 1 à 10. Ensuite, la sortie de ces neurones est connectée aux 2 neurones de la couche de sortie à travers 20 poids W^2_{jk} où j = 1 à 10 et k = 1, 2. L'entrée de la fonction d'activation dans la couche cachée s'écrit:

$$a_j = \sum_{i=1}^{9} w_{ij} * X_i + X_\theta \theta_j \quad \text{j=1, 2…10} \qquad \text{(rel 5.1)}$$

Où X_i est la $i^{ième}$ sortie de la couche d'entrée connecté au $j^{ième}$ neurone de la couche cachée et X_θ est une constante égale à 100mV, connecté au $j^{ième}$ élément de polarisation dans la couche cachée. W_{ij} est le poids qui connecte la $i^{ième}$ entrée de la couche d'entrée au $j^{ième}$ neurone de la couche cachée. Nous pouvons alors calculer la valeur X_j de la $j^{ième}$ sortie de la couche cachée en utilisant la relation 5.2.

$$X_j = \tanh(a_j) \qquad \text{j=1, 2…10} \qquad \text{(rel 5.2)}$$

De façon similaire, l'équation 5.3 donne la fonction d'activation pour la couche de sortie. De même , la sortie effective, Xk, de la couche de sortie est donnée par la relation 5.4.

$$a_k = \sum_{j=1}^{10} w_{jk} * X_j + X_\theta \theta_k \quad \text{k=1, 2} \qquad \text{(rel 5.3)}$$

$$X_k = \tanh(a_k) \qquad \text{k=1, 2} \qquad \text{(rel 5.4)}$$

Lors de la première passe durant la phase d'apprentissage, ces poids sont initialisés par une valeur aléatoire (comprise entre -900mV et 900mV). L'algorithme de propagation inverse permet de

mettre à jour ces poids en calculant l'erreur de propagation (circuit Bloc Delta). En commençant par la couche de sortie (propagation inverse), cette erreur de propagation est donnée par la relation 5.5.

$$\delta_k=(X_T-X_O)D_k \quad k=1, 2 \qquad \text{(rel 5.5)}$$

Où X_T est la cible (Valeur désirée), X_O est le neurone de sortie et D_k est la dérivée de la $k^{ième}$ fonction d'activation dans la couche de sortie.

L'écart de la valeur du poids, mise à jour dans la couche de sortie (addition ou soustraction), est déterminé par l'équation 5.6.

$$\Delta W_{kj}=\eta(X_T-X_O)D_kX_{j=}\ \eta\delta_kX_j \quad j=1, 2…10 \text{ et } k=1, 2 \qquad \text{(rel 5.6)}$$

Où η est le taux d'apprentissage, que nous avons fixé à 0,25.

Le nouveau poids est alors donné par la relation 5.7.

$$W_{kj}(new)=W_{kj}(old)+\Delta W_{kj} \quad j=1, 2…10 \text{ et } k=1, 2 \qquad \text{(rel 5.7)}$$

De même, l'erreur de propagation inverse, δ_j, à la couche cachée est donnée par l'équation 5.8.

$$\delta_j=\sum_{j=1}^{10} \delta_k W_{kj} \quad j=1, 2…10 \text{ et } k=1, 2 \qquad \text{(rel 5.8)}$$

De façon similaire à la couche de sortie, la valeur du poids de la couche cachée est mise à jour en utilisant les relations 5.9 et 5.10.

$$\Delta W_{ij}= X_i\ \delta_j \quad i=1, 2…9 \text{ et } j=1, 2…10 \qquad \text{(rel 5.9)}$$

$$W_{ij}(new)=W_{ij}(old)+\Delta W_{ij} \quad i=1, 2…9 \text{ et } j=1, 2…10 \qquad \text{(rel 5.10)}$$

Ces différentes équations nous permettent de définir l'architecture du réseau de neurones implémentant l'algorithme de rétro-propagation (cf. Figure 5.3). Lorsque l'erreur dans la couche cachée devient suffisamment faible, le processus d'apprentissage est terminé. Les poids sont fixés et le processus de test (ou de classification) peut commencer.

Nous avons fixé le taux d'apprentissage, η, en prenant en compte le compromis suivant. Les valeurs des poids sont moins stables et la phase d'apprentissage est rapide si la valeur de η est élevée. A contrario, ces valeurs sont plus stables et la phase d'apprentissage plus lente, si le taux d'apprentissage est faible. Lors des simulations sous Matlab, le meilleur compromis a été trouvé pour une valeur de η=0,25. Ce taux d'apprentissage est représenté par un courant dans notre architecture, sa valeur a ainsi été fixée à une valeur constante de 40nA.

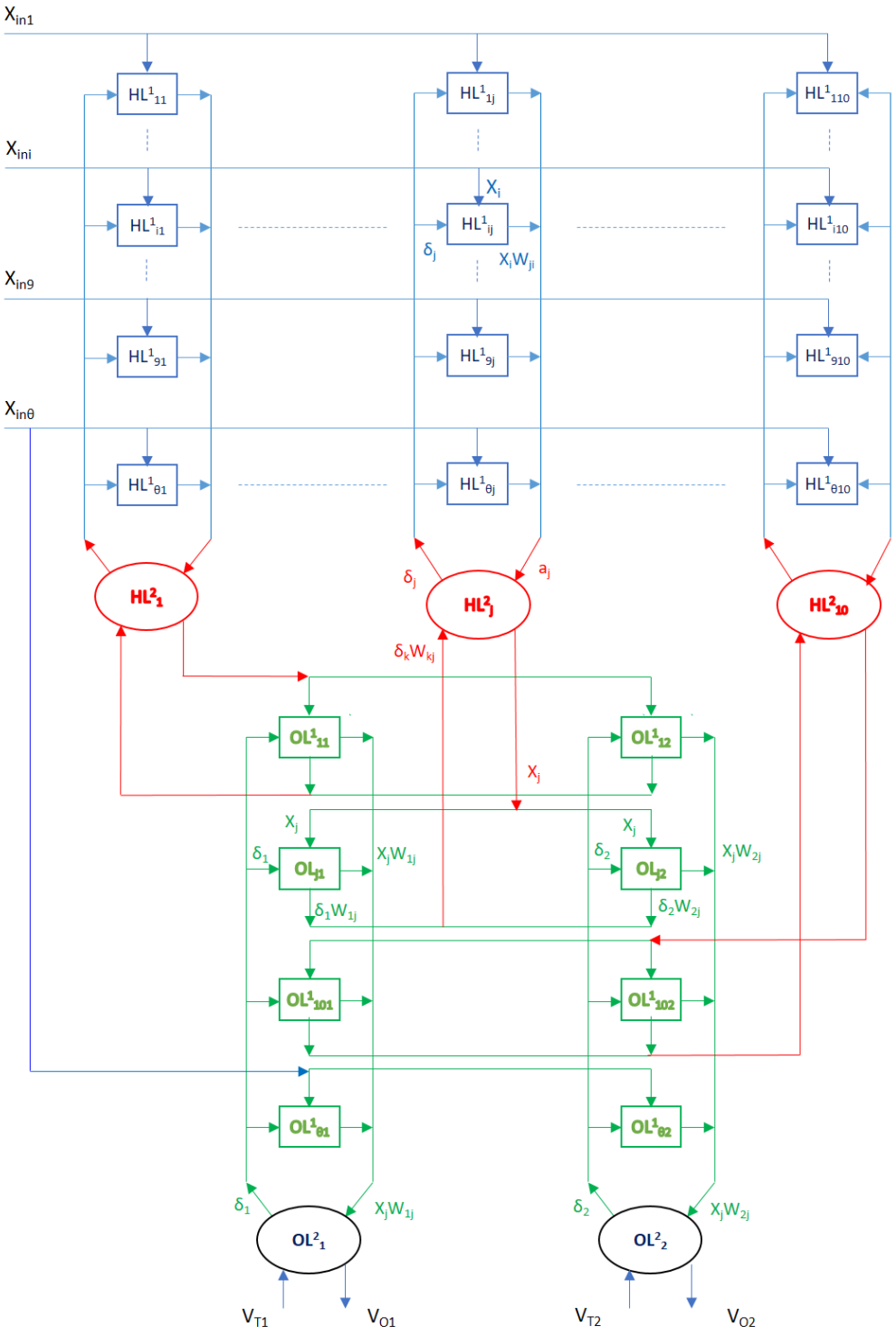


Figure 5.3 : Conception du RNA MLP avec propagation inverse

3.2. Description de l'architecture finale

La figure 5.3 décrit l'architecture du RNA réalisant la classification des cellules cancéreuses. L'objectif de ce paragraphe est de détailler chacun des blocs de cette architecture, qui seront implémentés en utilisant les briques de base conçues précédemment : multiplieur, fonction

d'activation et sa dérivée (tangente hyperbolique ou logsigmoid), bloc delta. La figure 5.4 présente, pour chaque bloc, le détail de son implémentation à partir des blocs de base. Sur cette figure, la description des variables de chaque bloc est résumée sur le tableau 5.1.

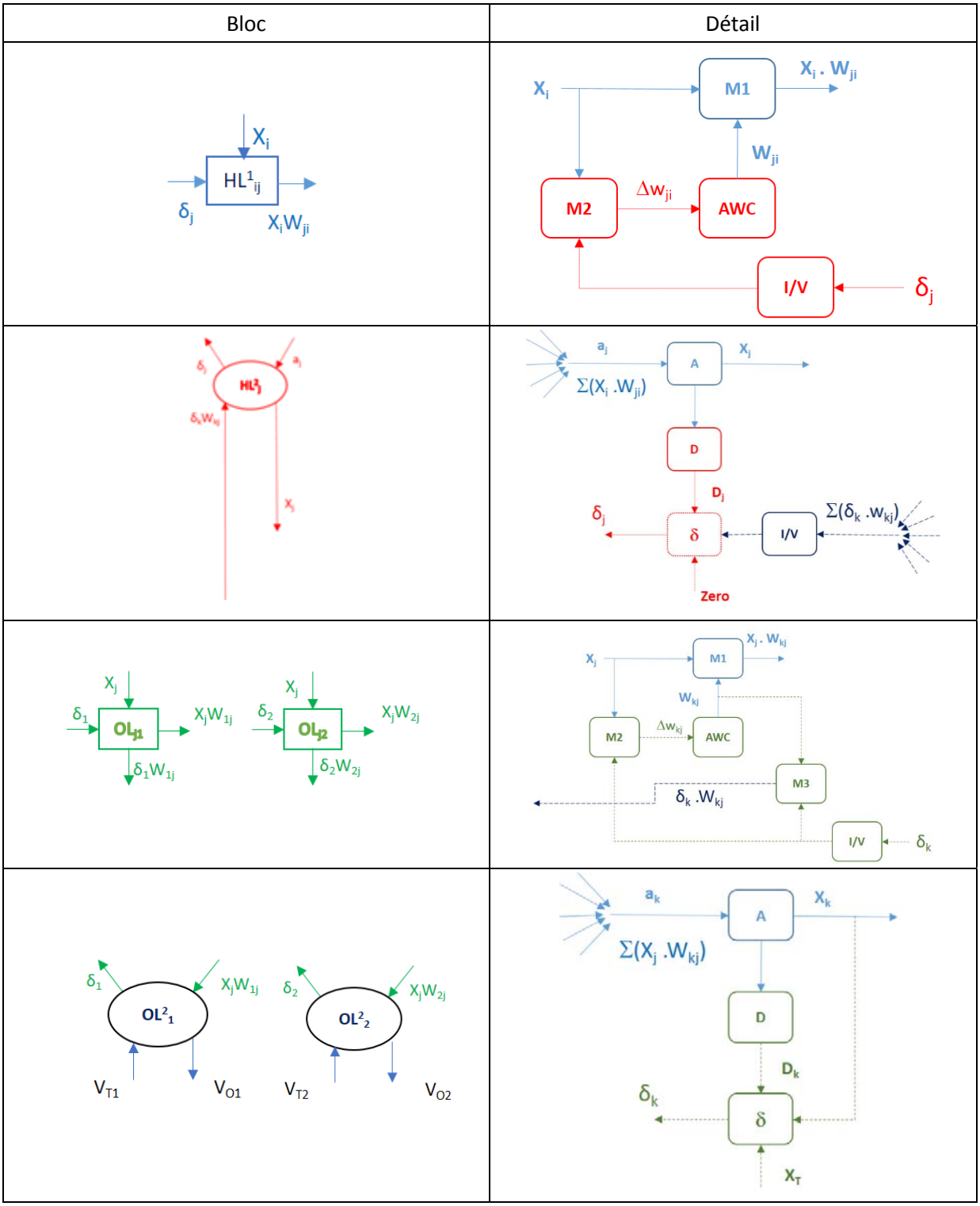


Figure 5.4 : Détail de chaque bloc utilisé dans la figure 5.3

Tableau 5.1 : Variables de la figure 5.4 et leur description respective

Variable	Description	Unité
X_i	Couche d'entrée (Attributs Wisconsin, i=1, 2…9)	V
ΔW_{ji}	Poids dans la couche cachée	V
a_j	Somme des entrées multipliées par le poids dans la couche cachée Neurone j (j=1, 2…10)	A
X_j	Entrée dans la couche de sortie en V	V
W_{kj}	Poids dans la couche de sortie en V	V
a_k	Somme des entrées multipliées par le poids dans la couche de sortie Neurone k (k=1, 2)en A	A
ΔW_{kj}	Poids dans la couche de sortie	V
X_k	Sortie dans la couche de sortie	V
D_j	Dérivée du neurone j de la couche cachée	A
D_k	Dérivée du neurone k de la couche cachée	A
X_T	Sortie désirée (Cible)	V
δ_j	Différence entre la couche de sortie et la cible, multipliée par la dérivée dans la couche de sortie	A
δ_k	Fonction Delta (Somme des $\delta_j W_{kj}$ multipliée par la dérivée dans la couche cachée	A

Tableau 5.2 : Fonction et implémentation de chaque bloc

Bloc	Description	Implémentation
M1	Multiplication de X_j par W_{ji}	Multiplieur large plage
M2	Multiplication de X_j par δ_j	Multiplieur de Gilbert
M3	Multiplication de W_{kj} par δ_k	Multiplieur large plage
A	Fonction d'activation dans les couches cachée et de sortie	Tangente hyperbolique ou logsigmoid
D D	Fonction dérivée de la fonction d'activation	Circuit « dérivée »
δ δ	Fonction dérivée de la fonction d'activation	Bloc Delta
I/V	Convertisseur Courant-Tension	Convertisseur V-I
AWC	Changement de poids adaptatif	Charge du condensateur

Dans la figure 5.4, les détails de chaque bloc sont implémentés par les circuits analogiques décrits au chapitre précédent. Pour faciliter la lecture et faire le lien avec l'architecture globale et les équations 5.1 à 5.10, le tableau 5.2 présente la description de chaque bloc.

Le circuit final, RNA complet, ayant été conçu, l'étape suivante consiste à réaliser des simulations Spice, au niveau transistor, des phases d'apprentissage et de classification, afin de valider la topologie retenue.

4. Simulations du RNA

4.1. Introduction

Pour réaliser les simulations, nous avons traduit les attributs de la base de données Wisconsin, sous forme de tensions comprises entre 10 et 100mV. La figure 5.5 présente ces 9 attributs pour les 699 biopsies (valeurs d'entrée Cancer du réseau de neurones, cf. Tableau 1.3, page 18) dont 458 cas bénins et 241 cas malins. Ces signaux seront utilisés en entrée pour les différentes simulations, dont les paramètres sont résumés dans le tableau 3.1 (page 69).

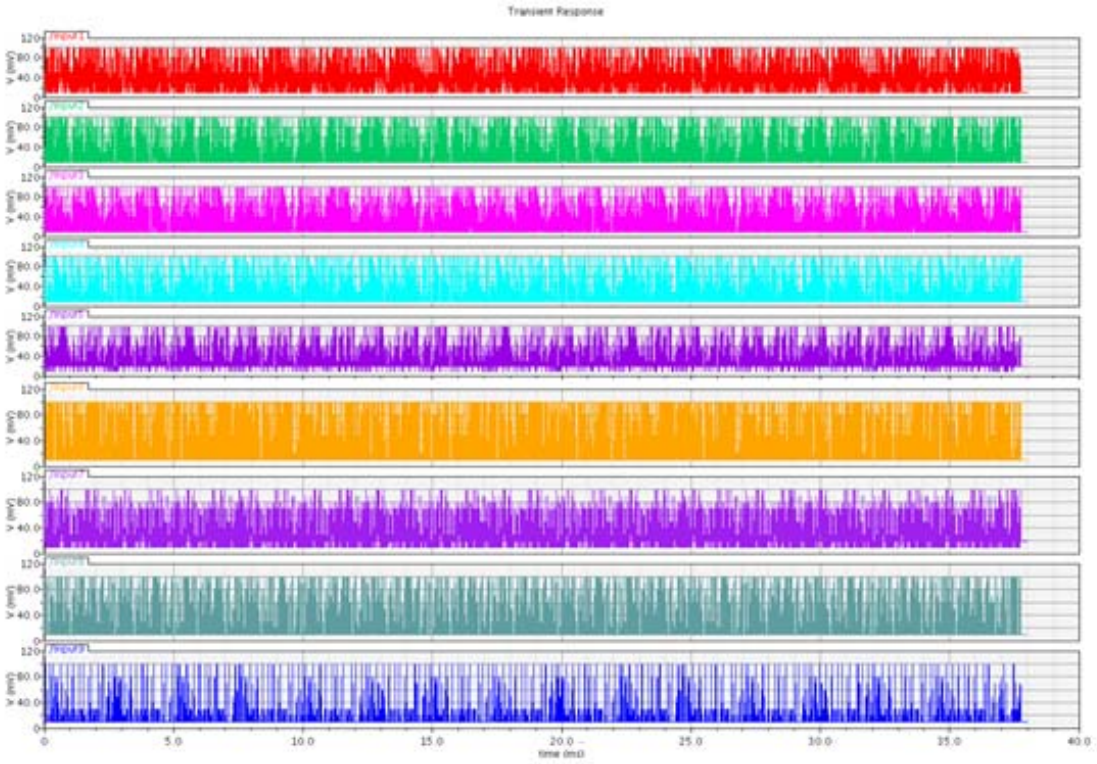


Figure 5.5 : Détail de chaque bloc utilisé dans la figure 5.2

Lors de la phase d'apprentissage, le nombre d'échantillons, le nombre d'époques et le taux d'apprentissage sont respectivement de 489, 14 et 0,25, comme dans le cas des simulations Matlab

réalisées au chapitre 3. La durée des simulations est de 34,25ms pour la phase d'apprentissage et de 37,73ms au total (phase d'apprentissage et phase de test ou classification, soit (6845+699)*5µs). En effet, le temps de traitement entre deux échantillons est d'environ 5µs (soit une fréquence maximale d'environ 200kHz).

Durant cette phase d'apprentissage, deux options sont possibles : Apprentissage en temps continu ou Apprentissage en temps discret. Pour la première option, le commutateur commandé par le signal d'horloge C_{lku} est toujours fermé jusqu'à l'obtention de la convergence de l'architecture. À ce moment, la phase de test peut commencer.

Nous avons testé (simulé) le réseau de neurones en utilisant soit la fonction tangente hyperbolique, soit la fonction logsigmoid, comme fonction d'activation, à la fois pour la couche cachée et la couche de sortie.

4.2. Fonction d'activation logsigmoid sans biais

La figure 5.6 présente l'évolution des poids W^H_{ij} (i = 1 à 9 et j = 1 à 10) dans la couche cachée, alors que la figure 5.7 illustre celle des poids W^O_{jk} (j = 1 à 10 et k = 1 à 2) dans la couche de sortie. Rappelons que dans le cas de la couche cachée, ils sont au nombre de 90. En effet, la couche cachée est constituée de 10 neurones, reliés chacun aux 9 entrées (9 attributs de la base de données illustrés à la figure 5.5). De même, ce nombre de coefficients est de 20 pour la couche de sortie, constituée de 2 neurones, reliés chacun aux 10 sorties de la couche cachée.

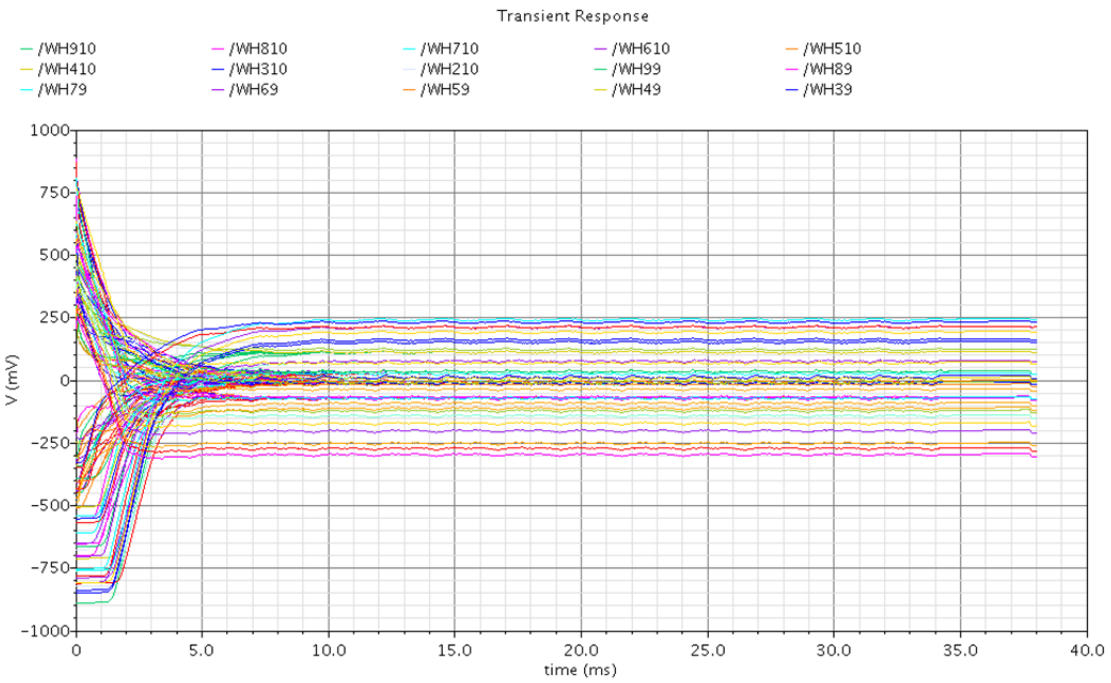


Figure 5.6 : Variations des poids de la couche cachée

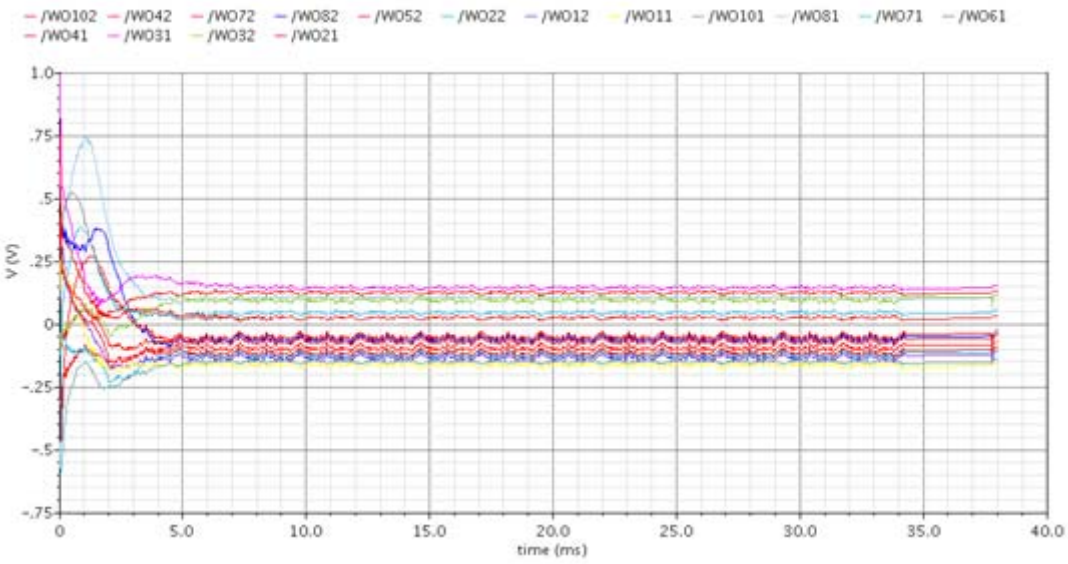


Figure 5.7 : Variations des poids de la couche de sortie

Sur ces deux figures, nous pouvons constater qu'au bout de 34,23ms (durée de la phase d'apprentissage), les poids ne varient plus. Dans les deux cas, les valeurs initiales ont été tirées arbitrairement entre -900 et +900mv (dynamique d'entrée du multiplieur). On peut constater que pour chaque cas les valeurs extrêmes sont comprises dans l'intervalle [-300V, +250mV].

Si la sortie souhaitée est élevée (100 mV) et que la sortie des neurones est supérieure ou égale à 50 mV (seuil de décision) ou si la sortie souhaitée est faible (0 V) et que la sortie des neurones est inférieure à 50 mV, la réponse est correcte. Les figures 5.8 et 5.9 illustrent les sorties du réseau de neurone sans biais avec une fonction d'activation logsigmoid d'une part pour la classe bénigne et d'autre part pour la classe maligne.

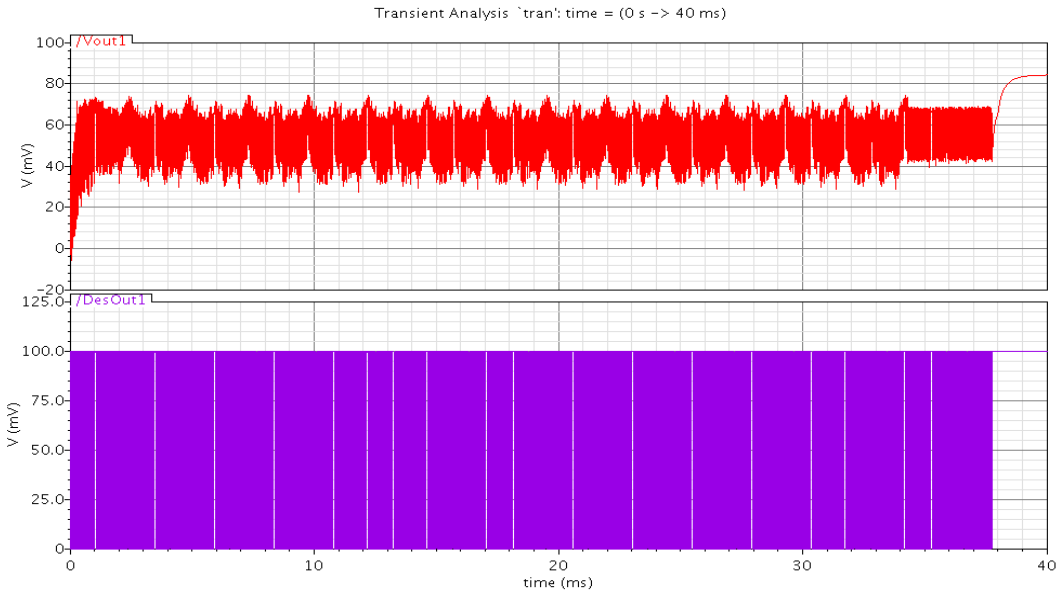


Figure 5.8 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

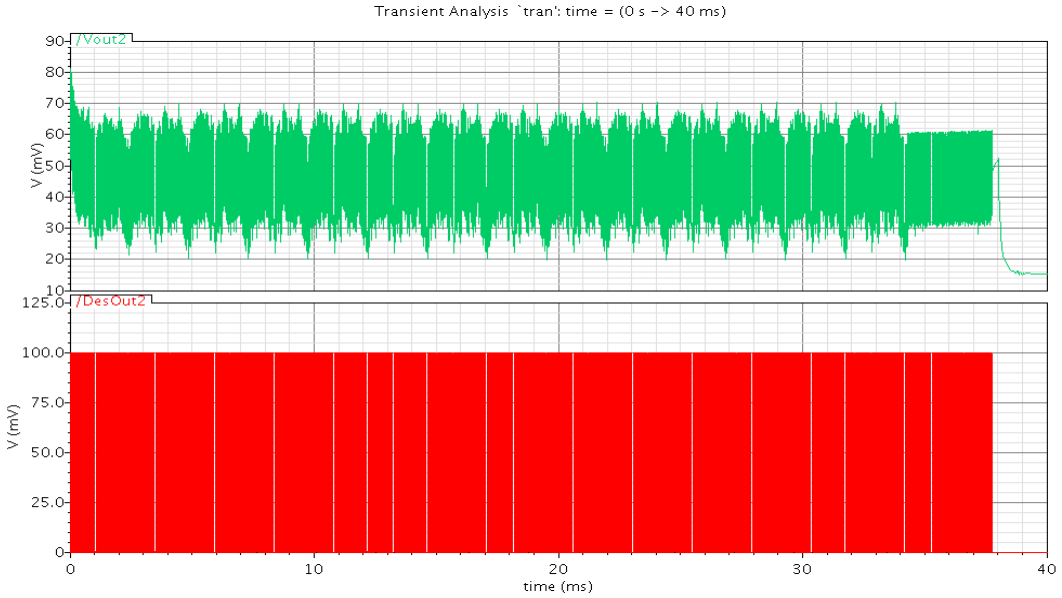


Figure 5.9 : Signal de sortie du RNA pour la classe maligne avec au-dessous la sortie désirée

Comme dans le cas des simulations Matlab, l'interprétation visuelle de ces courbes est relativement difficile, nous avons donc rajouté un bloc de comparaison entre la sortie réelle et la sortie désirée afin de calculer le taux d'erreur du RNA. Ces résultats sont résumés par la matrice de confusion, présentée sur la figure 5.10.

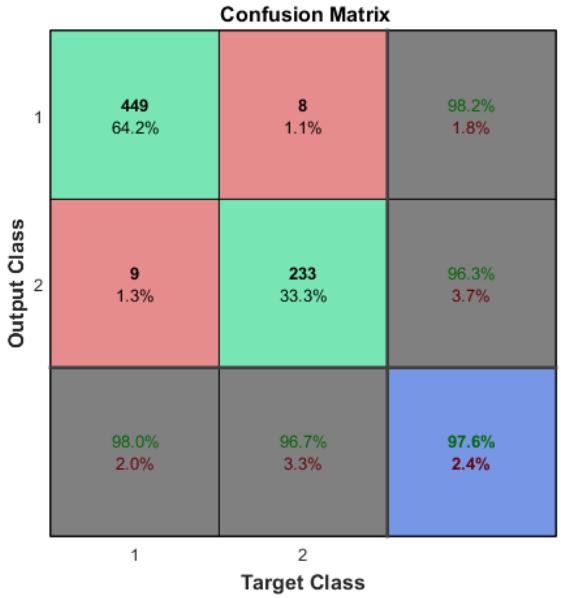


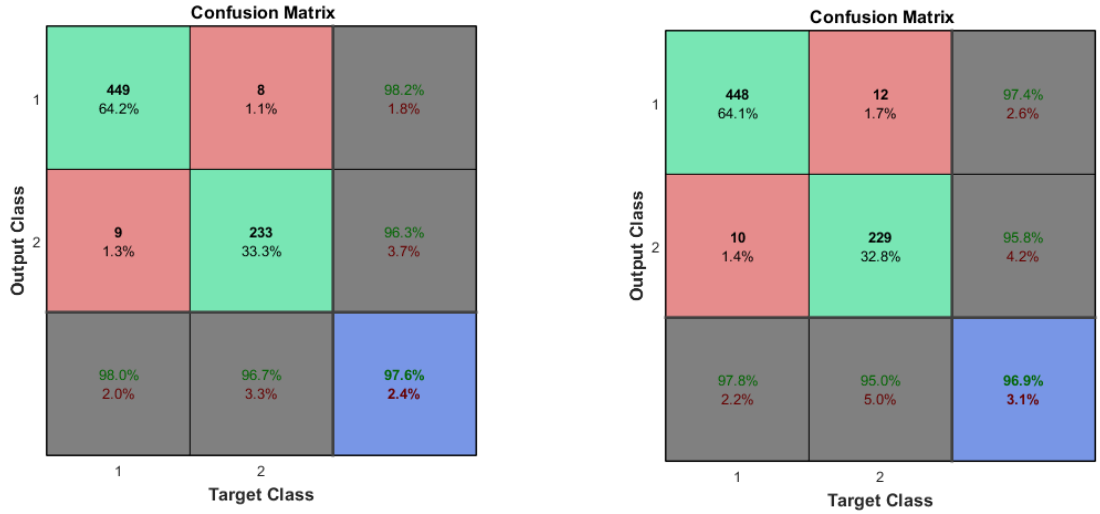
Figure 5.10 : Matrice de confusion du RNA sans biais et avec une fonction d'activation logsigmoid

La valeur du taux d'erreur global de 2,4% est au-delà de nos espérances, car en-dessous des simulations Matlab. A ce stade de nos analyses, et par manque de comparaison avec d'autres bases de données, il est difficile d'en tirer des conclusions exhaustives. Toutefois, ce résultat nous conforte dans le choix de l'architecture du RNA retenu et valide d'une part notre cahier des charges et d'autre part le design des blocs de base.

Afin de confirmer cette conclusion, nous avons réalisé des simulations avec une architecture avec ou sans biais et en utilisant soit la fonction logsigmoid, soit la fonction tangente hyperbolique comme fonctions d'activation, à la fois pour la couche cachée et celle de sortie. Nous ne présenterons que les matrices de confusion pour chaque cas, les résultats de simulation Spice (signaux de sortie) sont reportés en annexe 2.

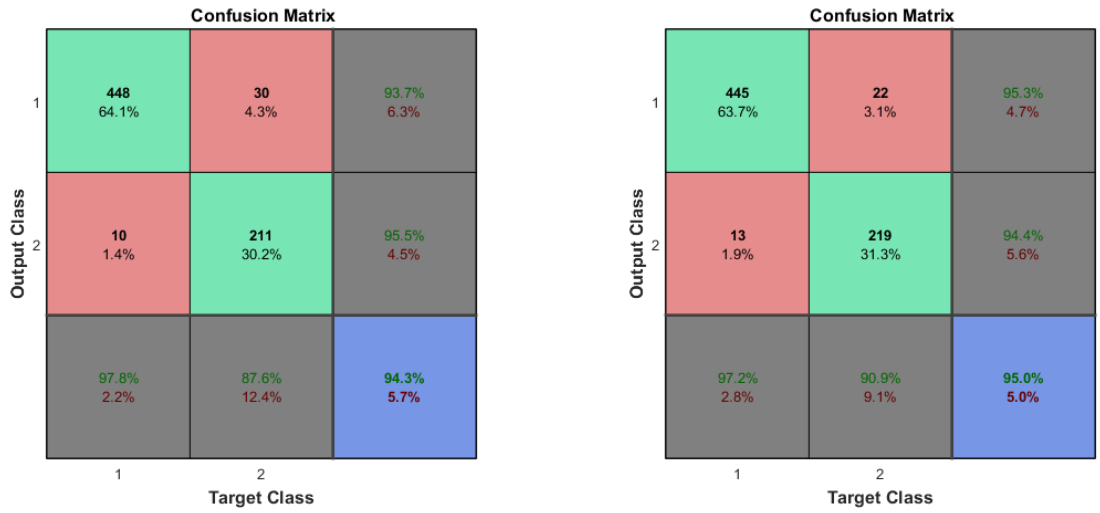
4.3. Différentes fonctions d'activation avec ou sans biais

Les figures 5.11 et 5.12 résument sous forme de tableau les résultats de simulations pour les différentes configurations. Pour faciliter les comparaisons nous avons également repris la matrice de confusion présentée à la figure 5.10.



(a) Sans biais (b) Avec biais

Figure 5.11 : Matrice de confusion du RNA pour une fonction d'activation logsigmoid



(a) Sans biais (b) Avec biais

Figure 5.10 : Matrice de confusion du RNA avec une fonction d'activation tangente hyperbolique

L'ensemble de ces résultats sont d'une part conformes aux simulations Matlab, réalisées au chapitre 3 et d'autre part valide à la fois l'architecture retenue, la conception et le dimensionnement des circuits électroniques réalisés précédemment.

5. Conclusion

Dans ce dernier chapitre, nous avons implémenté l'ensemble des circuits de base pour réaliser un réseau de neurones (9-10-2) pour une application particulière, à savoir la classification de cellules cancéreuses. Ce réseau pourrait être utilisé pour d'autres applications, toutefois sa topologie est figée. Ainsi le nombre d'entrée ne peut pas dépasser 9 et le nombre de sortie est fixé à 2.

Le circuit final comporte environ 5900 transistors. Les consommations estimées, au niveau transistor, sont résumés dans le tableau 5.3. La consommation totale du RNA est d'environ 25 mW.

Tableau 5.3 : Consommation de chaque bloc

Bloc	Consommation
Multiplieur	3 à 5,7 µW et 1,8 à 1,9 µW
Fonction d'activation	7,8 à 10,8 µW
Circuit Dérivée	230 à 250 nW
Bloc Delta	2,8 µW
Total (Neurone)	0,5 à 1,4 mW

Dans ce chapitre, nous avons également validé le choix de notre architecture, le cahier des charges ainsi que le dimensionnement des blocs de base. L'étape suivante consisterait à réaliser le dessin des masques, puis de fabriquer et tester le circuit final.

6. Bibliographie

[1] P. Murphy and D. Aha, "UCI Repository of machine learning databases," 1994.

[2] G.-M. Bo, "Microelectronic Neural Systems: Analog Vlsi For Perception And Cognition," in *Thesis*, 1998 November .

[3] H. Jouni, A. Harb, G. Jackmod and Y. Leduc, "Wide Range Analog CMOS Multiplier for Neural Network Application," in *EEETEM2017*, Beirut, 2017.

[4] C. Lu, B.X. Shi, L. Chen, "An on-chip BP learning neural network with ideal neuron characteristics and learning rate adaption. Analog Integr," in *Circuit. Signal Process.* , 2002, 31, 55–62. .

[5] T. Morie, Y. Amemiya, "An all-analog expandable neural network LSI with on-chip backpropagation learning.," in *IEEE J.Solid-State Circuits* , 1994, 29, 1086–1093..

[6] J. Hertz, A. Kough and R. Palmer, "Introduction to the Theory of Neural Computation," in *Addison-Wesley, Reading*, 1991.

[7] R. P. Lippmann. , “An introduction to computing with neural nets.,” in *IEEE ASSP Magazine*, 1987,April, pp. 4-22 april.

[8] D. E. Rumelhart and J. L. McClelland, “Parallel distribuited processing,” in *MIT Press*, Cambridge, 1986.

[9] M. Valle, D.D. Caviglia, G. Donzellini, A. Mussi, Oddone F. and G.M. Bisio., “ A neural computer based on an analog VLSI neural network.,” in *Proceedings of ICANN'94, Sorrento, vol. 2:pp. 1339{1342*, Italy, 1994., 26- 29 May .

[10] J.J. Hopfield, D.W. Tank, “computing with neural circuits,” in *A model science*, 1986, 233, 625-633.

[11] D. Coue, G. Wilson, “A four quadrant subthreshold mode multiplier for analog neural-network applications.,” in *IEEE Trans. Neural Netw.* , 1996, 7, 1212–1219.

[12] J.B. Lont, W. Guggenbuhl, “Analog CMOS implementation of a multilayer perceptron with nonlinear synapses.,” in *IEEE Trans. Neural Netw.* , 1992, 3, 457–465.

[13] M. Milev, M. Hrstov, “Analog implementation of ANN with inherent quadratic nonlinearity of the synapses.,” in *IEEE Trans. Neural Netw.* , 2003, 14, 1187–1200.

Conclusion générale et perspectives

1. Conclusion

L'objectif principal de ce travail de thèse était de réaliser un réseau de neurones artificiel, par une implémentation de cellules CMOS analogiques, dans le cas de la classification des cellules cancéreuses (bégnines ou malignes) du sein. Nous avons ainsi dans un premier temps étudié différents algorithmes, déjà publiés, afin de déterminer le mieux adapté à cette application. Une étude sous Matlab, utilisant la bibliothèque « Neural Network Toolbox », nous a permis de retenir l'architecture MLP avec rétro-propagation, réalisée à partir de 9 entrées (9 attributs des biopsies, proposés par la base de données Wisconsin), 10 neurones dans la couche cachée et 2 pour la couche de sortie (Réseau nommé 9-10-2). Cette étude au niveau système, nous a également permis de retenir l'algorithme le plus efficace pour notre application, à savoir le choix du nombre 14 d'époques, du taux d'apprentissage de 0,25 et d'une fonction d'activation de type Logsigmoid sans biais. Le cahier des charges et les spécifications de chaque bloc de base (multiplieur, fonction d'activation et sa dérivée) ont également été ainsi affinés.

Les réseaux neuronaux ont une capacité remarquable à dériver du sens des données complexes ou imprécises et peuvent être utilisés pour extraire des modèles et pour détecter des tendances trop complexes pour être remarquées par l'homme ou par d'autres techniques informatiques. Ainsi, la mise à jour des coefficients dépend des valeurs initiales aléatoires ainsi que du choix des exemples (couple entrée-sortie désirées), sans pour autant changer de façon significative le résultat de classification. Dans notre application, nous avons choisi la méthode de propagation inverse continue dans le temps (i.e. le poids est mis à jour en temps continu lors de la phase d'apprentissage). Cette méthode est stable et permet d'éviter le partage et l'injection de charge grâce au commutateur qui contrôle la mise à jour des poids. Notre principale contribution a alors consisté à implémenter un nouveau circuit analogique CMOS VLSI pour la mise en œuvre de la fonction d'activation, de sa dérivée, ainsi que des multiplieurs. Nous avons également réalisé une modélisation et simulation comportementale pour valider et spécifier ce réseau. Enfin nous avons réalisé un circuit faible consommation, permettant l'intégration d'un nombre très important de neurones. Une grande partie des objectifs initiaux a été atteinte :

1 - Analyse et conception de l'implémentation CMOS analogique VLSI de systèmes neuronaux

2 - Définition et analyse des cellules primitives neuronales

3 - Exploitation des briques de base et de périphériques simples pour l'implémentation finale

4 - Utilisation de transistors en faible ou forte inversion pour réaliser des fonctions plus ou moins complexes avec une faible consommation

5 - Validation du circuit final dans le cadre d'une application particulière : classification des cellules cancéreuses.

Le dimensionnement et l'optimisation des circuits de base ont été réalisés en utilisant une technologie CMOS 130nm mature et donc faible coût (technologie HCMOS9A de la société STM). L'accent a été mis sur les circuits suivants :

- Deux multiplieurs : multiplieur de Gilbert et multiplieur quatre quadrants à grande dynamique d'entrée. Ce dernier offre une sortie simple en courant qui élimine le circuit de sommation supplémentaire à la sortie du neurone et réduit ainsi la surface et la consommation du circuit (RNA) final
- Deux versions de la fonction d'activation : logsigmoid et tangente hyperbolique
- La fonction dérivée de cette fonction d'activation pour chacun des deux cas, basée sur une cellule de Gilbert
- Convertisseur courant-tension et bloc Delta pour le calcul d'erreur.

Sous Cadence, nous avons vérifié et validé l'architecture analogique du RNA. Les résultats de classification sont excellents puisque nous avons atteint la valeur référence obtenue sous Matlab, à savoir un taux d'erreur global de 2,4% dans le meilleur cas. Les simulations avec mise à jour continu du poids à temps continu ont également permis de valider ce choix, ainsi que la valeur du taux d'apprentissage global. Le circuit final fonctionne sous une alimentation de ±900mV pour une consommation de seulement 25mW. Seule la réalisation du dessin des masques, la fabrication et le test de la puce n'ont pas été réalisés par manque de temps.

2. Perspectives

Le premier futur travail consistera à réaliser le dessin des masques en vue de la fabrication de la puce. Des simulations « post-layout », incluant des analyses de type Monte-Carlo ou pire cas pour l'étude des variations PVT, devraient également permettre de valider la topologie du circuit avant fabrication. Notons toutefois que la maturité de la technologie (HCMOS9A 130 nm), en plus de d'offrir un coût de fabrication très faible, devrait garantir ce point.

Dans l'objectif du test du premier prototype, nous avons d'ores et déjà réalisé un générateur de signal programmable permettant de fournir en entrée les 9 attributs de la base de données

Wisconsin pour chaque biopsie. On trouvera, en Annexe 3, une description de ce générateur de signaux, dédié à notre application.

Le second effort portera sur l'électronique de sauvegarde et mise à jour des coefficients ou poids des neurones. Des circuits à capacités plus performants ou l'utilisation de synapses à « porte flottantes » ou mémoire non volatiles à long terme, devraient permettre d'améliorer ce point. Le dernier point et non le moindre consiste à rendre ce réseau reconfigurable, nombre de neurones variables dans les différentes couches afin de s'adapter à différentes classes d'application.

Publications liées à cette thèse

1. H. Jouni, M. Issa, A. Harb, G. Jacquemod & Y. Leduc, «Neural Network Architecture for Breast Cancer Detection and Classification», IMCET, Beirut, Lebanon, 2016

2. H. Jouni, A. Harb, G. Jacquemod & Y. Leduc, «Wide Range Analog CMOS Multiplier for Neural Network Application», EEETEM, Beirut, Lebanon, 2017

3. H. Jouni, A. Harb, G. Jacquemod & Y. Leduc, «Programmable signal generator for neural network application», ICM, Beirut, Lebanon, 2017

4. H. Jouni, A. Harb, G. Jacquemod & Y. Leduc, «Creation of Real Blocks for Neural Network using Simulink», ICCA, Beirut, Lebanon, 2018

Annexe 1 : Mise à jour des coefficients

Résultats avec Biais

	Fonction d'activation Couche cachée	**Fonction d'activation Neurones de sortie**	**- Poids de la couche cachée - Poids des Biais pour la couche cachée Valeurs min et max**	**- Poids pour les neurones de sortie - Poids pour les biais des neurones de sortie Valeurs min et max**	**Erreur Moyenne**
Premier Algorithme	Tangente hyperbolique	Tangente hyperbolique	-2.72 et 1.90 -3.8 et 3.7	-0.66 et 1.39 -0.31 et 1.03	5.9%
Second Algorithme	Tangente hyperbolique	Tangente hyperbolique	-1.09 et 2.33 -3.65 et 1.97	-1.29 et 1.06 -0.22 et 0.06	6.2%
Premier Algorithme	LogSigmoid	LogSigmoid	-1.71 et 1.3 -3.21 et 2.72	-3.21 et 3.36 -1.18 et 0.58	3.1%
Second Algorithme	LogSigmoid	LogSigmoid	-1.81 et 1.48 -1.55 et 1.95	-2.9 et 2.58 -0.34 et 0.95	2.9%

Résultats sans Biais

	Fonction d'activation Caché Couche	**Fonction d'activation Neurones de sortie**	**-Poids de la couche cachée Valeurs min et max**	**-Poids pour les neurones de sortie Valeurs min et max**	**Erreur Moyenne**
Premier Algorithme	Tangente hyperbolique	Tangente hyperbolique	-7.24 et 6.92	-7.59 et 8.39	6.7%
Second Algorithme	Tangente hyperbolique	Tangente hyperbolique	-1.65 et 1.55	-3.97 et 1.83	6.6%
Premier Algorithme	LogSigmoid	LogSigmoid	-7.09 et 5.4	-10.83 et 10.78	3%
Second Algorithme	LogSigmoid	LogSigmoid	-1.37 et 1.17	-3.54 et 3.48	2.6%

Annexe 2 : Simulation Spice des différentes configurations

Fonction d'activation Logsigmoid avec biais

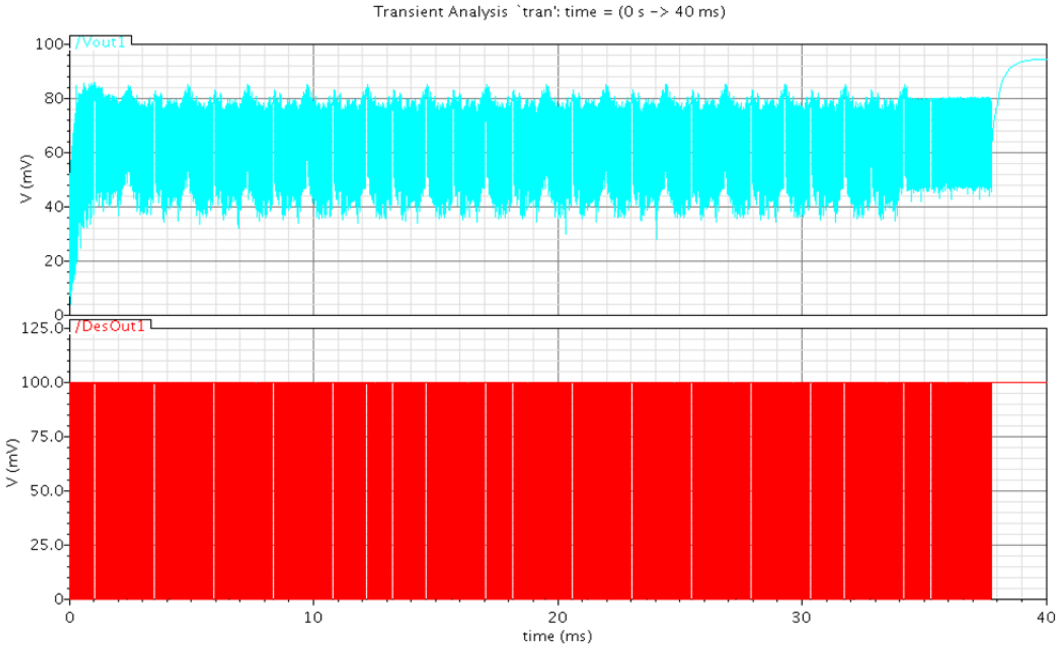


Figure A2.1 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

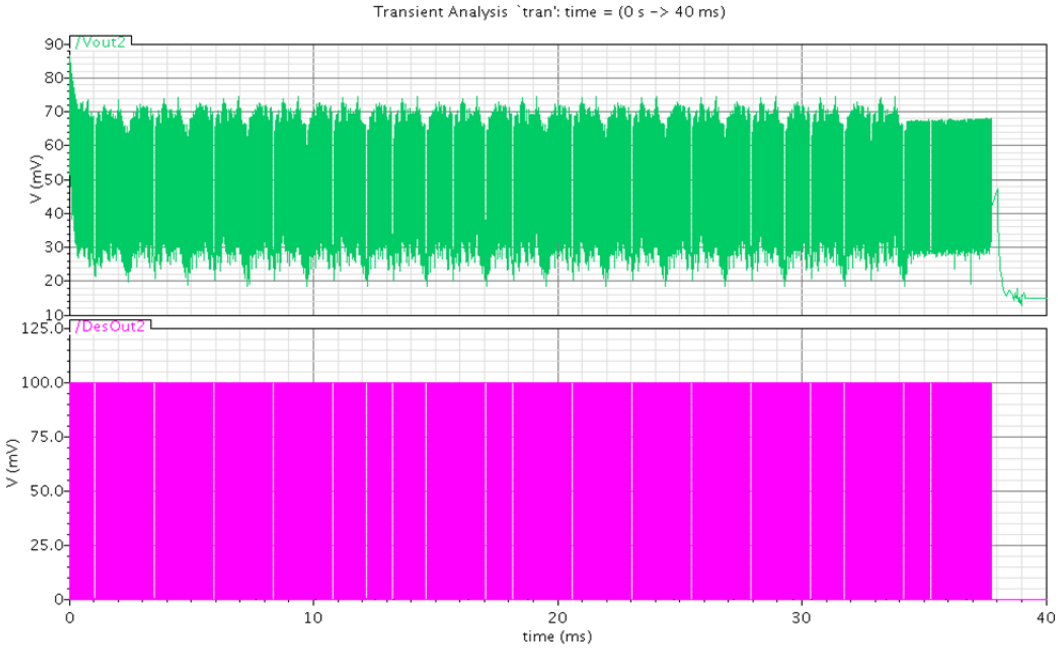


Figure A2.2 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

Fonction d'activation Tangente hyperbolique sans biais

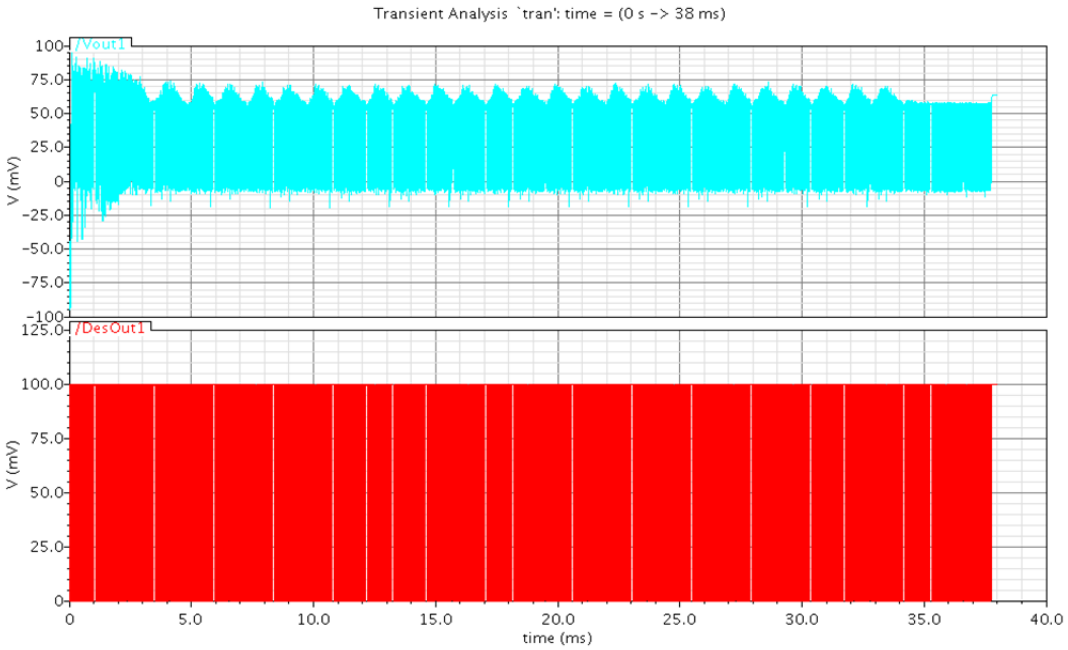


Figure A2.3 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

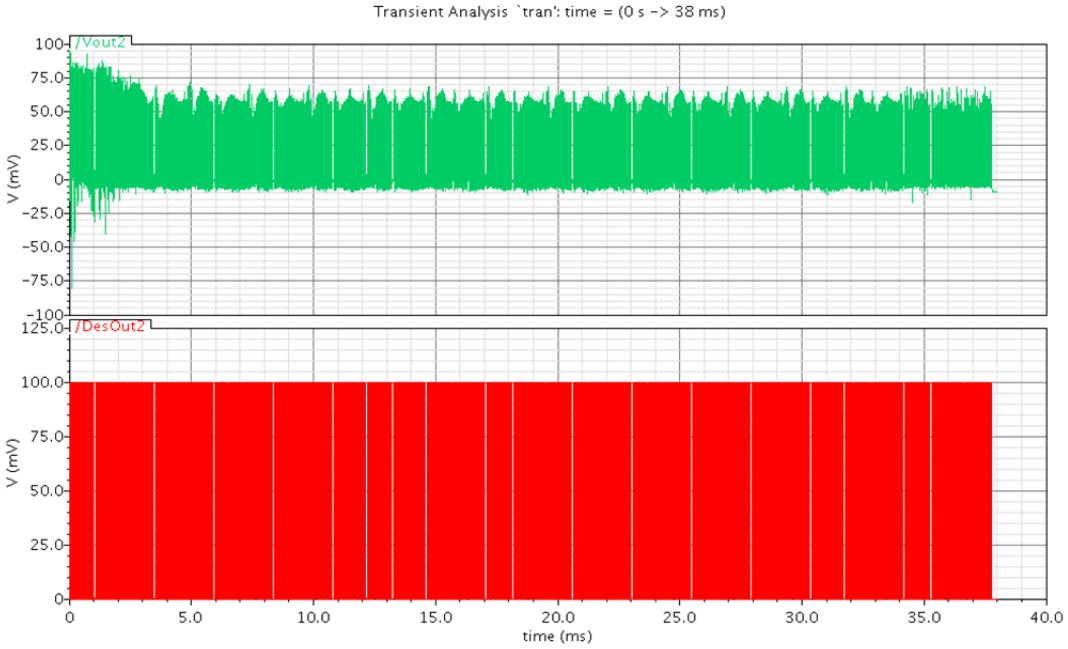


Figure A2.4 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

Fonction d'activation Tangente hyperbolique avec biais

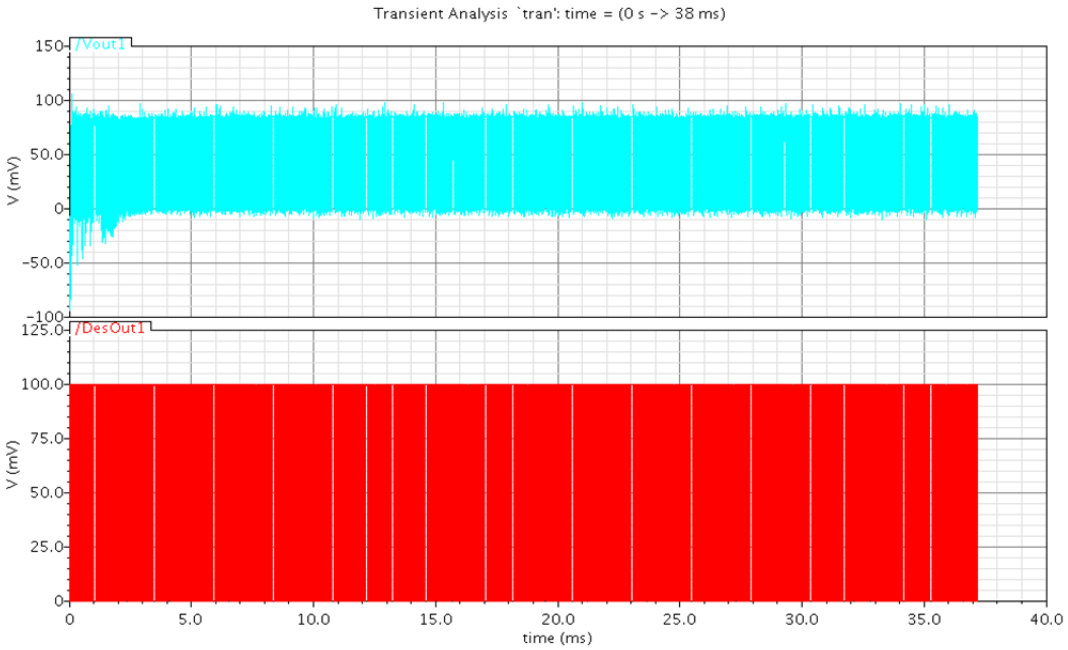


Figure A2.5 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

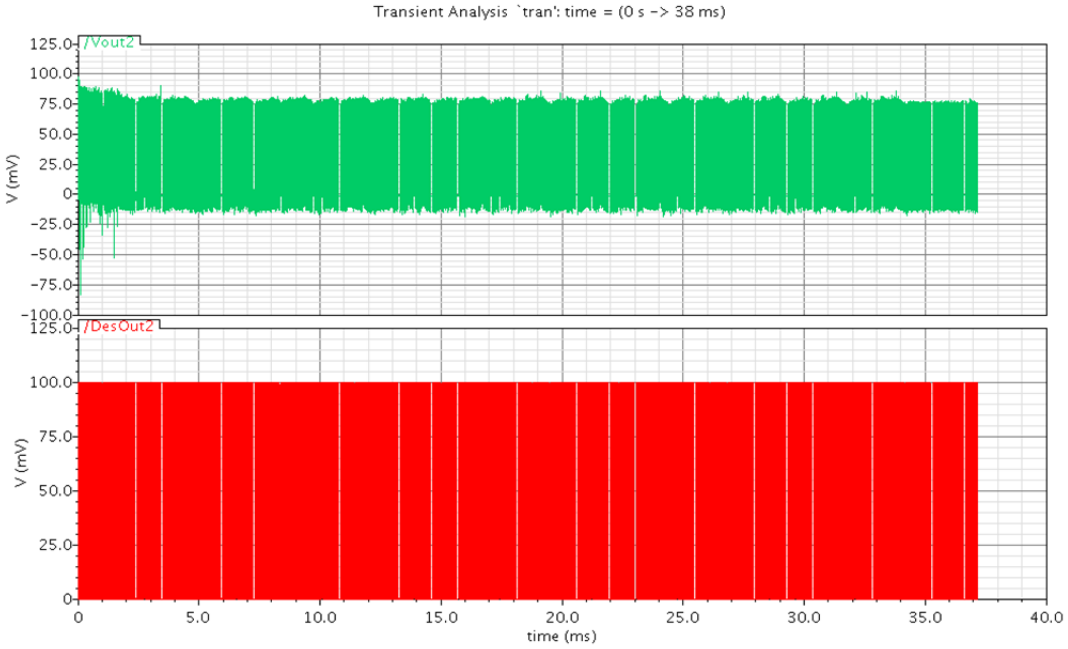


Figure A2.6 : Signal de sortie du RNA pour la classe bénigne avec au-dessous la sortie désirée

Résumé des résultats sans biais

	Fonction d'activation **Couche cachée et couche de sortie**	**-Poids de la couche cachée** **Valeurs min et max**	**-Poids pour les neurones de sortie** **Valeurs min et max**	**Erreur Moyenne**
Second Algorithme	Tangente hyperbolique	-210mV et 180mV	-290mV et 140mV	5.7%
Second Algorithme	LogSigmoid	-160mV et 180mV	-300mVet 290mV	2.4%

Résumé des résultats avec biais

	Fonction d'activation **Couche cachée et couche de sortie**	**-Poids de la couche cachée** **-Poids des Biais pour la couche cachée** **Valeurs min et max**	**- pour les neurones de sortie** **- Poids pour les biais des neurones de sortie** **Valeurs min et max**	**Erreur Moyenne**
Second Algorithme	Tangente hyperbolique	-310mV et 200mV -280mV et 150mV	-175mV et 200mV -50mV et 20mV	5.0%
Second Algorithme	LogSigmoid	-230mV et 270mV -190mV et 160mV	-220mV et 240mV -80mV et 110mV	3.1%

Annexe 3 : Générateur de signal programmable pour une application de réseau neuronal

L'objectif de ce travail était de réaliser un générateur de signaux programmable qui puisse servir de test pour le réseau de neurones analogique. En effet, ce générateur, basé sur l'utilisation de microcontrôleurs (MCU) PIC 18F680, doit fournir les signaux d'entrées (attributs Wisconsin) du circuit analogique VLSI implémentant le RNA. Pour réaliser cette fonction, nous avons utilisé 12 MCU, 11 multiplexeurs 4067 et un afficheur MCD LM032L. La figure A3.1 présente le synoptique de ce générateur de signaux, où les entrées sont modifiables par l'intermédiaire de résistances variables.

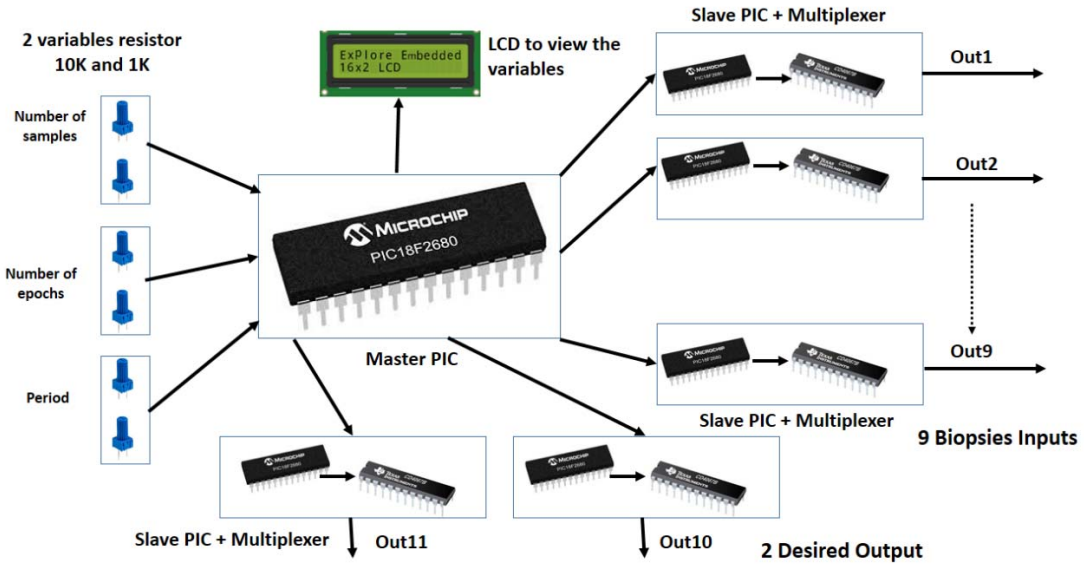


Figure A3.1 : Principaux circuits du générateur de signaux

Les entrées (résistances) du MCU maître sont les suivantes : la période, le nombre d'échantillons et l'époque ; elles sont transmises via le port A en utilisant les broches RA0, RA1 et RA2. Ces entrées sont converties en numérique avec le convertisseur analogique-numérique 10 bits. Les broches RB2, RB3, RB4, RB5, RB6 et RB7 du port B sont utilisées pour afficher ces valeurs.

La tâche de la broche RC0 du microcontrôleur maître est d'activer ou de désactiver tous les esclaves en même temps. Pour réinitialiser le microcontrôleur maître, appuyez sur le bouton d'arrêt (Fig. A3.3-a) pour connecter la broche de réinitialisation RE3 à la masse.

Le port C est utilisé comme sortie sauf la broche RC4. Lorsque le bouton de démarrage est enfoncé, le microcontrôleur maître transmet le nombre d'échantillons et le nombre d'époques à tous les microcontrôleurs esclaves, via les broches RC1 et RC2 respectivement en mettant RC1 haut et bas (période T) EP et RC2 haut et bas (Période T) NS fois pour chaque époque.

La figure A3.2 présente l'organigramme d'utilisation de ce générateur. Pour démarrer, il convient d'appuyer sur le bouton start (cf. Figure A3.3.a). La figure A3.3 illustre les entrées-sorties utilisées pour le MCU Maitre (cf. Figure A3.3.a) et les MCU esclaves (cf. Figure A3.3.b). Le MCU lit alors les entrées réalisées par les résistances variables (cf. Figure A3.4). Par exemple, pour le nombre d'échantillons, nous avons 2 résistances variables connectées ensemble en mode série montré à la Figure A3.4. Pour obtenir la valeur requise, la résistance 10-kΩ est utilisée pour le raffinement grossier et le 1-kΩ est utilisé pour de beaux changements.

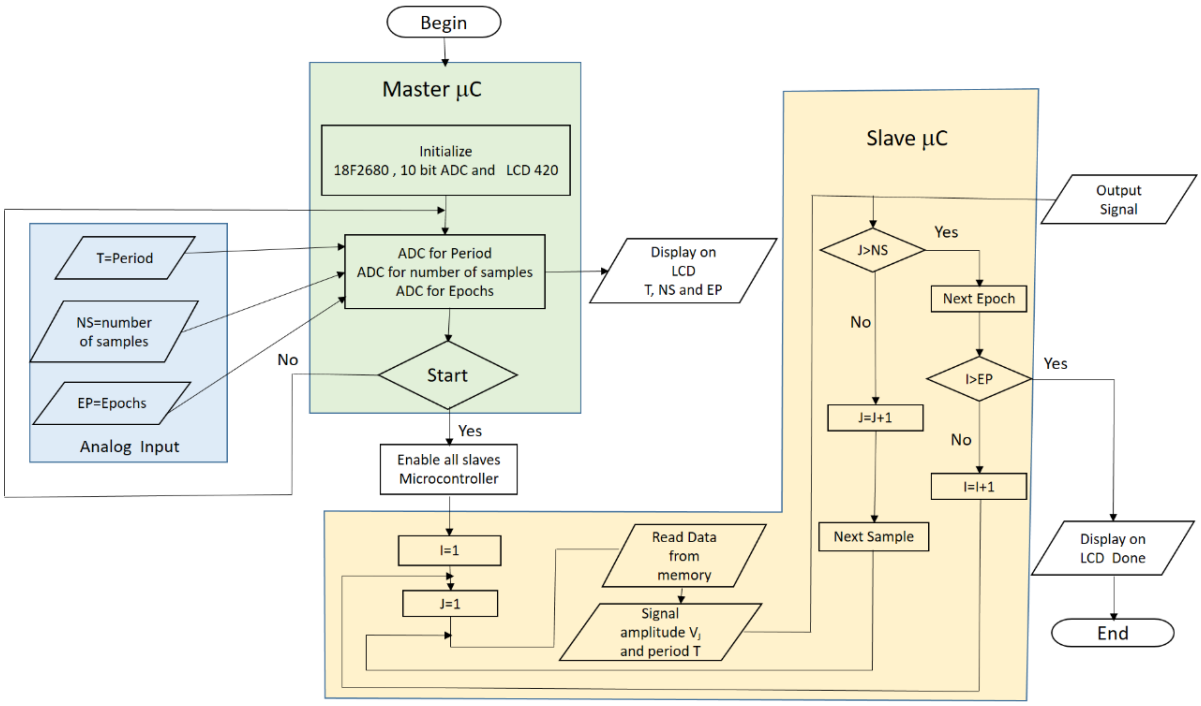
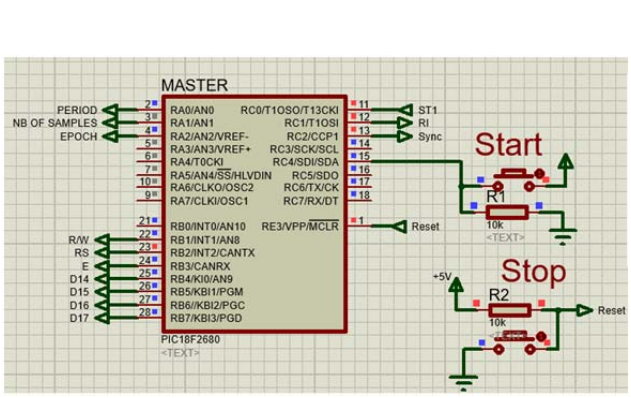
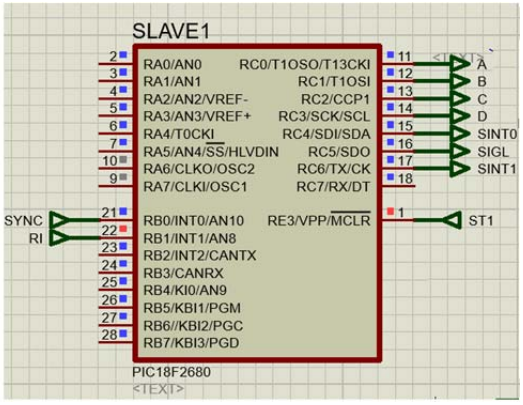


Figure A3.2 : Organigramme du générateur de signaux



(a) MCU Maître



(b) MCU Esclave

Figure A3.3 : Microcontrôleurs PIC 18F2680

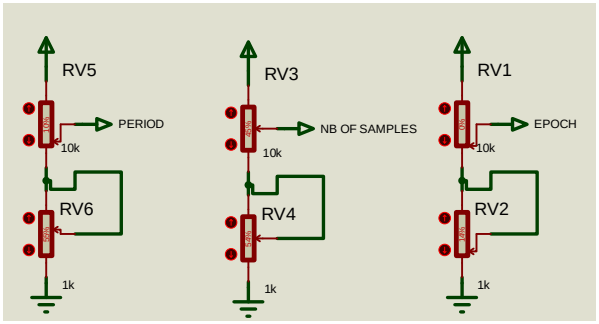


Figure A3.4 : Circuits pour les variables PERIOD, NB OF SAMPLES et EPOCH

Avant de créer le réseau neuronal, nous devons avoir l'ensemble des données par exemple dans notre cas, cet ensemble est pour la détection du cancer du sein : 699 exemples (9 entrées et 2 sorties désirées).

Après avoir construit la structure du réseau de neurones, nous choisissons un certain nombre d'exemples pour l'entraîner. Ce nombre est égal au nombre d'échantillons. Nous répétons cette opération plusieurs fois pour tous les exemples d'entraînement pour terminer le taux d'apprentissage. Le nombre d'opérations est égal aux époques (ou EPOCH), et enfin sur la période, nous sélectionnons la période du signal électrique pour chaque valeur.

Différentes valeurs pour la période, le nombre d'échantillons et l'époque peuvent être choisies. Mais dans notre référence, les nombres choisis d'échantillons = 489, et epochs = 14, et la période choisie = 200 μS. Les valeurs choisies sont affichées sur le LCD LM032L (cf. Fig. A3.5) par le microcontrôleur maître. Lorsque le bouton de démarrage est pressé, 11 signaux provenant des microcontrôleurs esclaves sont produits.

Chaque MCU esclave stocke ses propres données dans sa mémoire. Ces données sont liées à chaque biopsie et à la sortie désirée extraite de la base de données du cancer du sein. Dans le microcontrôleur esclave, les broches RB0 et RB1 connectées aux broches RC2 et RC1 du microcontrôleur maître fonctionnent respectivement comme entrées. Pour chaque impulsion reçue via RB0, la valeur de mémoire suivante provenant du microcontrôleur esclave est transmise par l'intermédiaire de RC0, RC1, RC2 et RC3 aux broches A, B, C et D du multiplexeur respectivement. L'entrée d'adresse ABCD du multiplexeur commande quelle entrée analogique, valeur comprise entre 0 et 1 V, sera transmise à la sortie (cf. Figure A3.6). Cette opération est répétée NS fois pour chaque époque lue à partir de RB1.

Le logiciel Proteus 4.8 SF0 Pro a été utilisé pour concevoir et simuler l'architecture du générateur de signaux tel que présenté sur la figure A3.7.

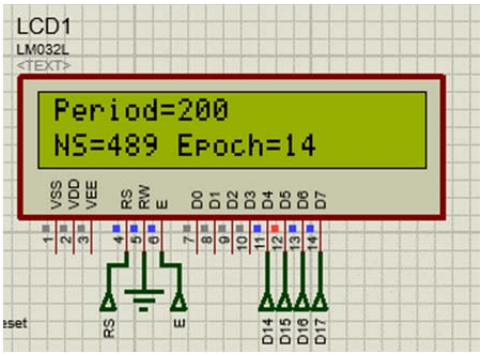


Figure A3.5 : Circuit d'affichage LCD LM032L

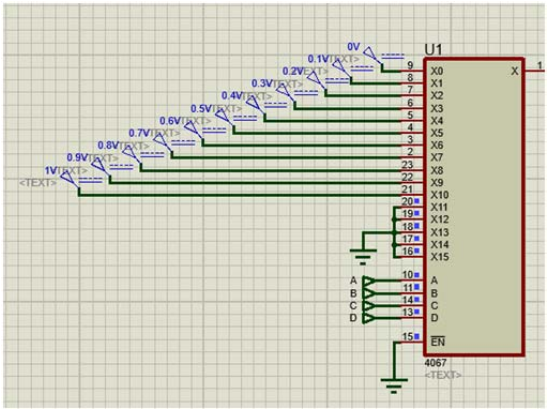


Figure A3.6 : Multiplexeur analogique 16 canaux

Tous les microcontrôleurs sont liés à des fichiers hexadécimaux créés par le logiciel PIC C via les programmes master.c, slave1.c, slave2.c, ... et slave11.c. Ces programmes contrôlent le microcontrôleur maître et tous les microcontrôleurs esclaves respectivement. Tous les programmes sont similaires sauf le master.c. La seule différence entre le slave1.c, slave2.c, ..., et slave11.c concerne les données à l'intérieur de la mémoire.

Les données pour les 8 microcontrôleurs esclaves varient de 0 à 1 par pas de 0,1. Elles sont stockées à l'intérieur des 8 valeurs de données de microcontrôleurs esclaves entre 0 et 10. Le multiplexeur permet de choisir la sortie appropriée. Par exemple, lorsque nous avons la valeur 5 dans la mémoire, nous avons une sortie égale à 0,5 V. Ainsi, les changements de données du microcontrôleur esclave numéro 6 ont la plage [0 0.1 0.2 0.3 0.35 0.4 0.5 0.6 0.7 0.8 0.9 1], mais les données équivalentes stockées dans ce microcontrôleur sont équivalentes à [0 1 2 3 4 5 6 7 8 9 10 11], dans les 2 derniers microcontrôleurs esclaves pour la sortie désirée, nous stockons dans la mémoire 0 et 1.

Si l'on veut visualiser 4 (par exemple) courbes sur « l'oscilloscope », on choisit sur la figure A3.7 quatre microcontrôleurs esclaves (3 biopsies d'entrée et 1 sortie désirée). En sélectionnant une période = 200 μs, un nombre d'échantillons = 489, et un nombre d'époque = 14, nous obtenons les graphiques présentés sur la figure A3.8 où les trois premières courbes représentent les trois premières entrées de l'ensemble de données représentent l'épaisseur de l'amas, l'uniformité de la taille des cellules et l'uniformité de la forme des cellules, et la dernière courbe indique la sortie souhaitée pour la catégorie bénigne.

Ce générateur de signal programmable peut être utilisé comme entrées et sortie désirée pour circuit intégré IC pour la détection du cancer du sein (sur puce d'apprentissage).

Son inconvénient principal concerne sa basse fréquence (5 kHz maximum), qui n'est toutefois pas critique dans notre cas.

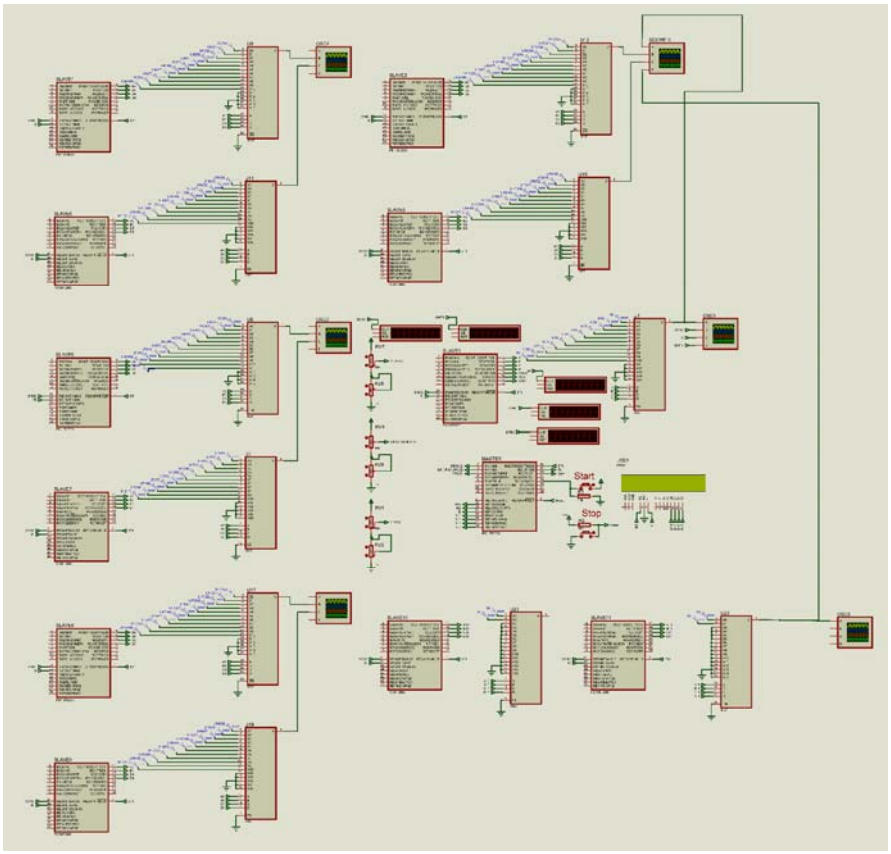


Figure A3.7 : Conception finale du générateur de signaux

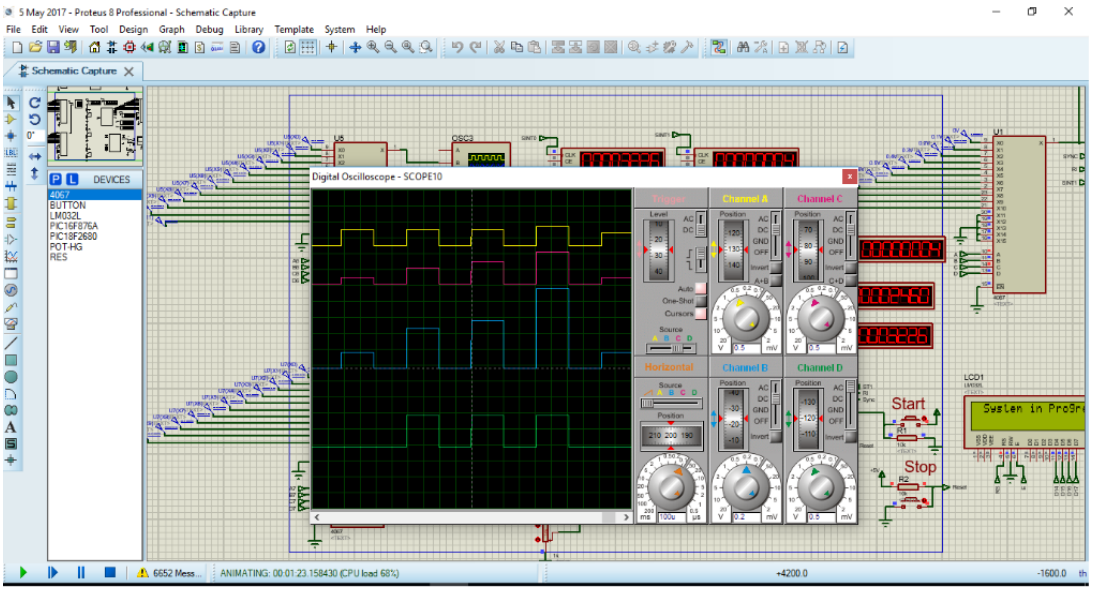


Figure A3.8 : Exemples de résultats de simulation

Résumé

Les réseaux de neurones artificiels sont particulièrement intéressants pour les implémentations CMOS VLSI car chaque élément parallèle (neurone ou synapse) est relativement simple, permettant l'intégration complète de grands réseaux sur une seule puce. Les multiplieurs, la fonction non-linéaire et sa dérivée sont des éléments clés essentiels dans le traitement du signal analogique et notamment dans la mise en œuvre VLSI analogique de réseaux neuronaux artificiels. Les principales conditions de ce type de circuits sont les suivantes : une faible surface de Silicium et une faible consommation électrique. Pour valider notre approche, nous avons choisi comme type d'application, la classification de cellules cancéreuses (malignes ou bénignes) du sein. Le réseau de neurones étudié dans cette thèse est basé sur l'architecture Multi-Layer Perceptron, formé par la rétro-propagation.

L'objectif principal est de trouver les meilleurs compromis et optimisations pour réaliser des circuits dans une technologie mature CMOS 130 nm afin d'avoir le coût le plus faible possible. Après avoir choisi le meilleur algorithme (le plus simple et efficace) pour une implémentation VLSI simple, nous avons défini une architecture analogique efficiente. Enfin les briques de base ont été conçues et réalisées avant l'intégration finale sur une faible surface de silicium et une faible consommation de puissance. Pour vérifier et valider le projet de la puce VLSI avant fabrication, une méthodologie de vérification a été proposée dans cette thèse. Elle nous a également permis de définir le cahier des charges de la puce, ainsi que celui des blocs de base. La topologie du RNA final est alors réalisée à partir des différents blocs de base puis validée par des simulations Spice au niveau transistor. Les résultats montrent que le taux d'erreur global atteint une valeur de 2,4% pour un RNA sans biais avec une fonction d'activation Logsigmoid, équivalent à celui obtenu sous Matlab. Le circuit final fonctionne sous une alimentation de ±900mV pour une consommation d'environ 25mW.

Mots clés : Réseau de neurones artificiels, Circuits VLSI analogiques, CMOS 130nm, Classification de cellules cancéreuses du sein

Abstract

The artificial neural networks are particularly interesting for CMOS VLSI implementations because every parallel element (neuron or synapsis) is relatively simple, allowing the complete integration of big networks on a single chip. Multipliers, non-linear function and its derivative are essential key elements in the analog signal processing in particular for analog VLSI implementation of artificial neuronal networks. The main conditions of this kind of circuits are the following ones: a low surface of Silicon and a low electric consumption. To validate our approach, we chose as type of application, the classification of cancer cells (malignant or benign) of the breast. The neural network studied in this thesis is based on Multi-Layer Perceptron with back-propagation.

The main objective is to find the best compromises and the optimizations to realize circuits in a mature CMOS 130nm technology to have the lowest cost. Having chosen the best algorithm (the simplest and most effective) as a simple VLSI implementation, we defined efficient analog architecture. Finally building blocks were designed and realized before the final integration on a low surface of silicon and low power consumption. To verify and validate the project of the VLSI chip before manufacturing, a methodology of check was proposed in this thesis. It also allowed us to define the specifications of the full chip, as well as that of the building blocks. The Spice simulated results show that the global error rate reaches 2,4 % for a RNA without bias with a Logsigmoid activation function, equivalent to that obtained under Matlab. The final circuit, with a ±900mV supply, exhibits a power consumption about 25mW.

Key words: Artificial Neuronal Network, Analog VLSI circuits, CMOS 130nm, Breast Cancer Classification

Résumé

Les réseaux de neurones artificiels sont particulièrement intéressants pour les implémentations CMOS VLSI car chaque élément parallèle (neurone ou synapse) est relativement simple, permettant l'intégration complète de grands réseaux sur une seule puce. Les multiplieurs, la fonction non-linéaire et sa dérivée sont des éléments clés essentiels dans le traitement du signal analogique et notamment dans la mise en œuvre VLSI analogique de réseaux neuronaux artificiels. Les principales conditions de ce type de circuits sont les suivantes : une faible surface de Silicium et une faible consommation électrique. Pour valider notre approche, nous avons choisi comme type d'application, la classification de cellules cancéreuses (malignes ou bénignes) du sein. Le réseau de neurones étudié dans cette thèse est basé sur l'architecture Multi-Layer Perceptron, formé par la rétro-propagation.

L'objectif principal est de trouver les meilleurs compromis et optimisations pour réaliser des circuits dans une technologie mature CMOS 130 nm afin d'avoir le coût le plus faible possible. Après avoir choisi le meilleur algorithme (le plus simple et efficace) pour une implémentation VLSI simple, nous avons défini une architecture analogique efficiente. Enfin les briques de base ont été conçues et réalisées avant l'intégration finale sur une faible surface de silicium et une faible consommation de puissance. Pour vérifier et valider le projet de la puce VLSI avant fabrication, une méthodologie de vérification a été proposée dans cette thèse. Elle nous a également permis de définir le cahier des charges de la puce, ainsi que celui des blocs de base. La topologie du RNA final est alors réalisée à partir des différents blocs de base puis validée par des simulations Spice au niveau transistor. Les résultats montrent que le taux d'erreur global atteint une valeur de 2,4% pour un RNA sans biais avec une fonction d'activation Logsigmoid, équivalent à celui obtenu sous Matlab. Le circuit final fonctionne sous une alimentation de ±900mV pour une consommation d'environ 25mW.

Mots clés : Réseau de neurones artificiels, Circuits VLSI analogiques, CMOS 130nm, Classification de cellules cancéreuses du sein

Abstract

The artificial neural networks are particularly interesting for CMOS VLSI implementations because every parallel element (neuron or synapsis) is relatively simple, allowing the complete integration of big networks on a single chip. Multipliers, non-linear function and its derivative are essential key elements in the analog signal processing in particular for analog VLSI implementation of artificial neuronal networks. The main conditions of this kind of circuits are the following ones: a low surface of Silicon and a low electric consumption. To validate our approach, we chose as type of application, the classification of cancer cells (malignant or benign) of the breast. The neural network studied in this thesis is based on Multi-Layer Perceptron with back-propagation.

The main objective is to find the best compromises and the optimizations to realize circuits in a mature CMOS 130nm technology to have the lowest cost. Having chosen the best algorithm (the simplest and most effective) as a simple VLSI implementation, we defined efficient analog architecture. Finally building blocks were designed and realized before the final integration on a low surface of silicon and low power consumption. To verify and validate the project of the VLSI chip before manufacturing, a methodology of check was proposed in this thesis. It also allowed us to define the specifications of the full chip, as well as that of the building blocks. The Spice simulated results show that the global error rate reaches 2,4 % for a RNA without bias with a Logsigmoid activation function, equivalent to that obtained under Matlab. The final circuit, with a ±900mV supply, exhibits a power consumption about 25mW.

Key words: Artificial Neuronal Network, Analog VLSI circuits, CMOS 130nm, Breast Cancer Classification