REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

ECOLE NATIONALE POLYTECHNIQUE
Département d'Electronique

Projet de fin d'études en vue de l'obtention du Diplôme d'Ingénieur d'Etat en Electronique

Implémentation sur FPGA d'un classificateur de chromosomes par utilisation de réseaux de Kohonen

Réalisé par : ***Melle BADACHE Faïza***
Mr DERDOUR Islam Houssam

Proposé et dirigé par : ***Pr. HAMAMI Latifa***
Melle ABID Faroudja

Juillet 2010

Ecole Nationale Polytechnique, 10 Av. Hassan Badi, El Harrach, Alger, Algérie

Résumé :

Notre travail consiste en la réalisation d'un système portable (sur puce) qui permet d'établir le caryotype humain et la détection d'éventuelles aberrations chromosomiques numériques. L'architecture proposée est formé de deux réseaux de Kohonen, Chacun des deux réseaux opère sur une des deux caractéristiques morphologique ou texturale du chromosome. Ceci permet d'obtenir en sortie une classification pondérée sur l'une des caractéristiques.

L'implémentation hardware de notre système sur circuit FPGA SPARTAN 3E (XC3S500E-4fg320) de XILINX a permis un traitement en temps réel. Un certain nombre de simplifications comme l'utilisation de la distance de Manhattan au lieu de la distance Euclidienne ont permis de diminuer la surface de la puce nécessaire
. Cette architecture a consommé 86% des ressources internes de la puce FPGA, avec un fonctionnement piloté par une horloge de 50 Mhz.

La performance de notre projet a été testée sur une base de données comprenant des cas normaux et anormaux d'aberration numériques.

Mot clés : chromosome, aberration, réseau de Kohonen, apprentissage, classification, FPGA, distance de Manhattan…

Abstract:

Our work consists of designing a portable system which detects chromosomal aberrations from a human caryotype. It is based on a multi-networks architecture made up of two Kohonen networks. The first carries out the classification of chromosomes according to their morphologies, and the second classifies them according to their band patterns.

The hardware implementation of our system in FPGA circuit SPARTAN3E (XC3S500E-4fg320) from XILINX allowed a real-time treatment. Some simplifications as the use of the distance of Manhattan instead of the Euclidian distance allowed to restrict the used surface of the chip. This architecture consumed 86 % of the internal FPGA chip resources, piloted with a 50 MHz clock.

The performance of our project was tested on a database includes normal and abnormal numeric cases of aberration.

Keywords: chromosomes, aberration, Kohonen, learning phase, classification, FPGA, Manhattan distance …

ملخص :

يتعلق عملنا بانجاز نظام محمول (على رقاقة) يسمح بالحصول على النمط النووي و الكشف على الانحرافات الكروموزومية العددية.

النظام المقترح مشكل من شبكتين من نوع كوهونان، كل شبكة تعمل على إحدى الميزتين المورفولوجية و النسيجية للكروموزوم هذا يسمح بالحصول على تصنيف مرجح لإحدى الميزتين.

التطبيق المادي لنظامنا على رقاقة FPGA SPARTAN 3E (XC3S500E-4fg320) المقترحة من طرف XILINX سمحت بزيادة وقت التنفيذ.

إن استخدام عدد من التبسيطات مثل استخدام مسافة مانهاتن عوضا عن المسافة الإقليدية مكننا من تقليل مساحة الرقاقة المستعملة. هذه الهندسة المادية استهلكت 86% من الموارد الداخلية من رقاقة FPGA مع ساعة تشغيل تقدر ب : 50 Mhz

قمنا بتجربة نتيجة نظامنا على قاعدة تظم صور كروموزومية تحتوي على حالات عادية و حالات انحرافات عددية.

مفاتيح: كروموزوم، انحرافات، شبكة عصبية، كوهونان، طور التعلم، طور التصنيف،FPGA، مسافة مانهاتن...

REMERCIEMENT

Nous tenons à remercier tout particulièrement et très chaleureusement notre Directrice de thèse, Pr L. Hamami, avec qui nous avons eu tant de plaisir à travailler et qui nous a fait profiter de son expérience. Qu'elle trouve ici notre sincère gratitude.

Nous remercions Melle ABID Faroudja, notre Co-promotrice pour l'aide qu'elle a toujours voulu nous apporter.

Nous exprimons notre profonde reconnaissance aux membres du jury d'avoir accepté de juger notre travail.

Que tous nos professeurs qui ont contribué à notre formation trouvent ici notre plus profonde gratitude.

Nous adressons nos plus sincères remerciements à tous ceux qui ont contribué, de près ou de loin, à l'aboutissement de ce travail.

Enfin, nous souhaitons dédier ce mémoire à nos parents. Rien n'aurait été possible sans leur soutien, confiance et générosité.

DEDICACE

A l'homme qui a consacré sa vie pour que la mienne soit meilleure,

par l'aide et le soutient qu'il ma réservé, Mon très cher père,

A ma très chère mère, pour sa patience, son aide et sa générosité,

A Naima, Sihem, Sara, Kenza, Ikram, Lyly, Maya,

A ma promotrice L. HAMAMI,

A mon binôme Houssam,

A tous mes amis,

A tous ceux qui me sont chers,

Je dédie ce travail.

Faïza.

DEDICACE

A ma mère, ma mère, ma mère et mon père

A mes grands-parents

A mes tantes : Samou, Ghanou et Naima

A mes frères : Akram et Aymen

A ma promotrice : L.HAMAMI

A mon binôme : Faiza

A tous mes amis

A tous ceux qui ont su croire en moi

A tous ceux qui me sont chers

Je dédie ce modeste travail

Houssam.

Sommaire

CHAPITRE III : EXTRACTION DES PARAMETRES CHROMOSOMIQUES

Liste des figures

Table des notations

ANN	Artificial Neural Network
CDTA	Centre de Développement des Technologies Avancées
ADN	Acide Désoxyribonucléique
PLD	Programmable Logic Device
CPLD	Complexe Programmable Logic Device
ASIC	Appliction Speific Integrated Circuit
FPGA	Field Programmable Gate Array
CMOS	Complementary Metal Oxide Semiconductor
RAM	Random Access Memory
SRAM	Static Random Access Memory
ROM	Read Only Memory
ISCN	International Standing Comittee on humain Cytogenetic Nomenclature
ISE	Integrated Software Environment
RPG	Rétro-Propagation du Gradient
CLB	Configurable Logic Bloc
IOB	Input Output Bloc
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VGA	Video Graphics Array
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
SOM	Self Organizing Map
W_j	Vecteur des poids synaptiques du j-ième neurone
X_j	Vecteur d'entrée du j-ième neurone
g	Fonction d'activation

r	Rayon de voisinage
$V(j,r)$	Voisinage de rayon *r* du neurone *j*
$h(j,n)$	Fonction qui décrit le mouvement de correction des neurones à la proximité du neurone j et à l'itération n
J^*	Neurone gagnant
$\eta(n)$	Fonction taux d'apprentissage
$min( )$	Fonction *min*
W_0	Coefficient de biais
N	Nombre d'itérations

Introduction generale

Introduction générale:

La classification automatique des chromosomes est l'un des plus importants problèmes sur lequel se sont penchés les chercheurs durant ce dernier quart du siècle.

La découverte des chromosomes comme support physique du matériel génétique de la cellule et l'implication de ce dernier dans la transmission du patrimoine héréditaire a révolutionné les techniques de diagnostic des maladies héréditaires et le dépistage des anomalies chromosomiques. L'établissement d'un caryotype humain est devenu une routine classique dans les laboratoires médicaux pour différents besoins tels le dépistage des malformations génétiques et le diagnostic du cancer à caractère héréditaire. Cette opération laborieuse, délicate, coûteuse en temps et en argent et nécessitant un personnel qualifié est réalisée le plus souvent manuellement par des cytogénéticiens entraînés.

La définition complète de la carte du génome humain et la formidable révolution ces vingt dernières années dans la manipulation génétique et ses implications directes sur le développement de produits nouveaux tels que des thérapies , des vaccins et le diagnostic de maladies génétiques n'ont fait que renforcer le besoin des cytogénéticiens pour les techniques modernes qui permettront de travailler individuellement sur chaque chromosome et d'exploiter toute l'information qu'il peut contenir.

En effet, qu'est ce qu'un chromosome dans l'imagerie médicale si ce n'est une entité ayant une forme et une texture caractéristiques du potentiel génétique qu'il véhicule.

Les chromosomes apparaissent sous forme de bâtonnets dans le noyau de la cellule au moment de la division (mitose ou méiose). Ils résultent de la segmentation et de la condensation du réseau de chromatine constituée d'un complexe d'ADN et de protéines dont la structure est aujourd'hui l'objet de recherches considérables à l'échelle mondiale.

La possibilité de leur analyse pendant la phase de la mitose où ils adoptent presque tous la forme d'un X avec deux (02) bras longs et deux (02) bras courts est facilitée grâce à des techniques de colorations permettant de subdiviser chaque bras en zones appelées régions, elles mêmes subdivisées en bandes et sous bandes.

Malgré tous les efforts déployés, les systèmes mis au point pour la réalisation d'un caryotype humain sont loin de concurrencer le savoir faire d'un cytogénéticien, le principal handicap résidant dans la séparation préalable des chromosomes se touchant ou se chevauchant.

La robustesse des réseaux de neurones artificiels (ANN) à opérer en milieu bruité ou même en présence d'un déficit des données a motivé leur utilisation dans le domaine de la cytogénétique automatisée.

Les réseaux de neurones les plus connus (perceptron multicouches, Kohonen, Hopfield) ont été utilisés seuls, combinés entre eux ou à d'autres techniques telles que la logique floue, etc.

Notre travail consiste à réaliser un système portable (sur puce) d'aide à la classification des chromosomes humains tout en introduisant un type particulier de réseaux de neurones qui est le réseau de Kohonen comme ossature principale du système de classification. Dans un premier temps, seules les caractéristiques géométriques dites morphologiques (taille et indice centromériques) sont prises en considération dans l'établissement du caryotype, par la suite on introduira la densité de profil, en supposant que le vecteur de caractéristiques ait déjà été extrait auparavant, il sera donc stocké dans une mémoire de notre système pour le traitement. Ce système sera implémenté sur un type particulier de circuits programmables : FPGA qui constitue le support d'intégration des modules de notre système. La plate forme de développement nous a été fournie par le Centre de Développement des Technologies Avancées (CDTA) où nous avons effectué un stage pratique d'un mois sous la direction de Melle ABID Faroudja.

CHAPITRE I :

LA CYTOGÉNÉTIQUE

1. Introduction

Avant tout, il nous paraît nécessaire de donner un aperçu de la matière première de notre projet. Le tout premier matériel dont disposent les cytogénéticiens, est un échantillon de quelques millilitres de sang prélevés sur un patient. Après plusieurs traitements, on doit obtenir des lames de belles mitoses nettement visibles au microscope photonique.

Dans la présente section, nous décrirons succinctement ce qui se trouve sur ces lames, comment l'obtient-on et dans quel but.

2. Mitose

La mitose est le mode de division le plus fréquent des cellules, au cours duquel le noyau subit de très importantes modifications. Le résultat d'une mitose est la constitution de deux cellules filles absolument semblables à la cellule mère qui leur a donné naissance [1]. Il existe un autre mode de division : la méiose, qui aboutit à la réduction de moitié du nombre de chromosomes dans le noyau des cellules filles.

3. Métaphase

La métaphase est une étape statique de la mitose. Le chromosome atteint sa condensation maximale, ses deux chromatides restent disposées côte à côte et reliées au niveau du centromère [2]. Les chromosomes sont répartis selon les espèces à la périphérie du fuseau ou distribués sur tout le cercle équatorial, ils sont indépendants les uns des autres (Figure I-1).

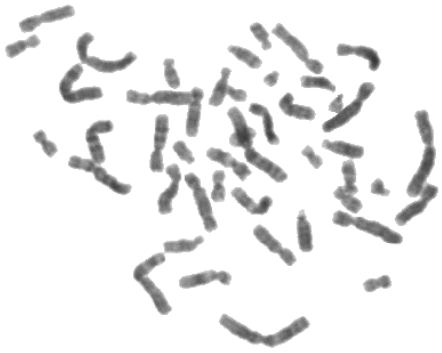


Figure I.1 Mitose (métaphase) d'une cellule d'homme normal.

4. Chromosome

Structure en forme de bâtonnet, située dans le noyau de toute cellule vivante et servant de support aux caractères génétiques propres à la cellule [3].

Les chromosomes représentent le matériel héréditaire des êtres vivants. Ils transmettent le message héréditaire de la cellule parentale à la cellule fille, c'est-à-dire le code à partir duquel sont réalisés les caractères phénotypiques de l'être vivant. Le nombre de chromosomes est immuable pour une espèce déterminée. A la mitose, chaque chromosome se présente comme un double filament plus ou moins long, probablement spiralé, présentant une constriction, le centromère. Le centromère sépare donc le chromosome en deux parties, de taille égale ou non [2]. Longueur totale du chromosome et position du centromère permettent de différencier les chromosomes les uns des autres et de les grouper. Chez l'homme (46 chromosomes), toutes les cellules, à l'exception des cellules gonadiques (ovule, spermatozoïde), sont dites diploïdes, car elles possèdent 44 chromosomes qui peuvent se ranger par paires identiques (les autosomes). La 23^{e} paire (chromosomes sexuels ou hétérochromosomes) est faite de chromosomes dissemblables chez l'homme (X et Y), de chromosomes similaires chez la femme (X et X).

5. Les télomères

Les télomères marquent l'extrémité des chromosomes. Ce sont des structures spécialisées qui confèrent aux chromosomes leur stabilité. En effet, lorsqu'un télomère est perdu, l'extrémité du chromosome devient très instable, et tend à fusionner avec les extrémités d'autres chromosomes « cassés », à subir des recombinaisons, ou tout simplement à être dégradée [4].

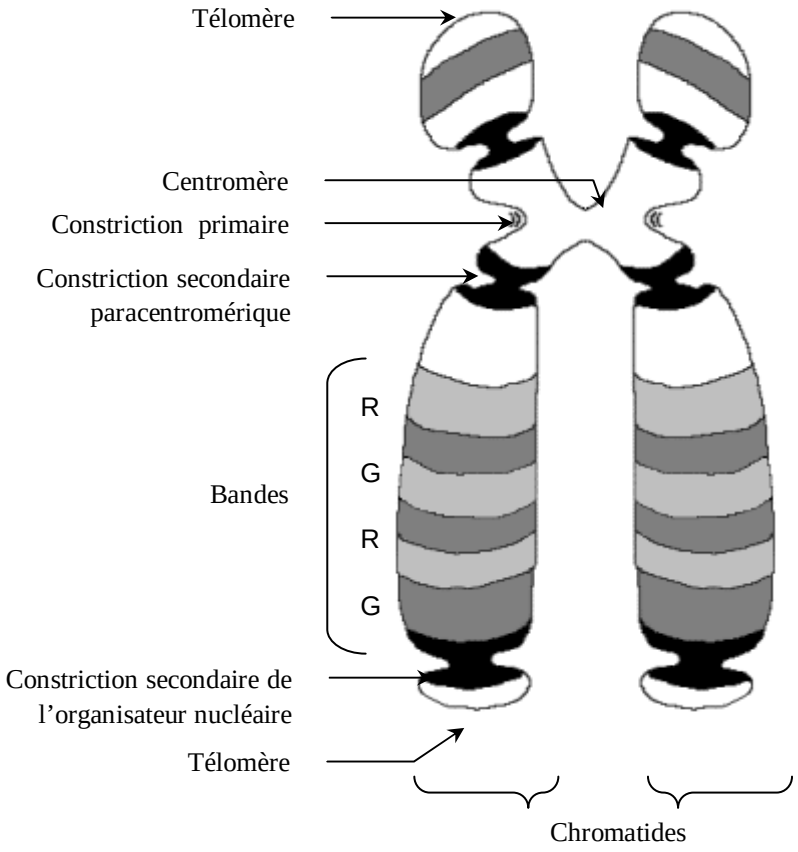


Figure I.2 Organisation d'un chromosome métaphasique.

6. Les centromères

Les centromères sont les régions de constriction et d'association entre les deux chromatides nouvellement répliquées d'un chromosome [4].

La position du centromère sur un chromosome étant fixe, on considère qu'une séquence nucléotidique précise définit le centromère. Les centromères permettent les migrations chromosomiques au cours de la mitose. Ils assurent donc le bon déroulement de la mitose et une répartition « en lots égaux » des chromosomes à chaque pôle de la cellule [2].

Selon la place du centromère, on définit trois types de chromosomes [4] :

- Les chromosomes métacentriques : dont le centromère est en position médiane (chromosomes 1, 3, 16, 19, 20 et X) ;
- Les chromosomes acrocentriques : dont le centromère est en position terminale ou distale (chromosomes 13, 14, 15, 18, 21, 22 et Y) ;

- Les chromosomes submétacentriques **:** dont le centromère est en position intermédiaire (chromosomes 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, et 17).

7. Le caryotype

C'est l'ensemble des chromosomes d'un individu classés par paires d'homologues, par morphologie identique, par la position du centromère, par bandes identiques (sombres/claires) et par ordre décroissant de taille [3]. Le nombre et l'aspect des chromosomes dépendent de l'espèce à laquelle ils appartiennent. Le caryotype sert à évaluer la place de l'espèce d'un point de vue évolutif, déceler des pathologies génétiques (trisomie, polyploïdie, etc..) et différencier le sexe de l'individu. C'est un examen très important pour la taxonomie notamment.

Etablir le caryotype correspond au dénombrement et à l'identification de tous ses chromosomes. Le tableau complet des 46 chromosomes humains en mitose est appelé **caryotype** humain (Figure I.3).

Sur les représentations schématiques ou les caryotypes, par convention, les bras courts (p) sont orientés vers le haut et les bras longs (q) dirigés vers le bas.

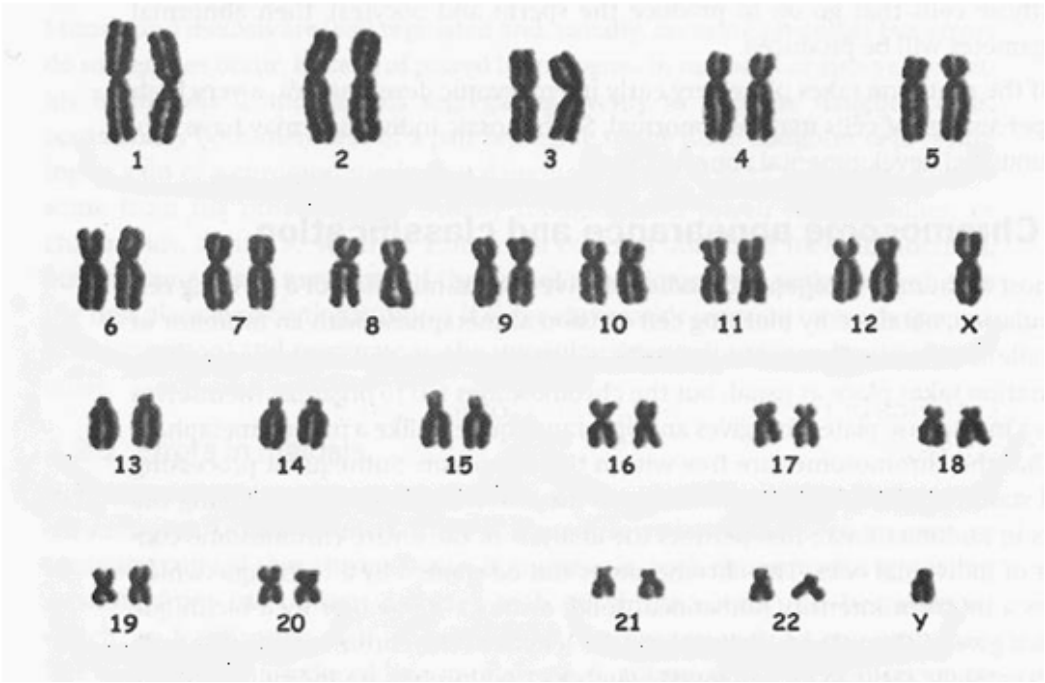


Figure I.3 Caryotype selon la classification de l'ISCN

8. Comment distinguer les chromosomes entre eux ?

Observés en métaphase, c'est-à-dire lorsqu'ils sont à leur état de condensation maximale, les chromosomes sont parfois difficilement différentiables (même forme, même longueur). Classiquement on les distingue par leur longueur et par la position de leur centromère.

Pour faciliter la comparaison, on utilise des colorants qui imprègnent certaines régions chromosomiques plus intensément que d'autres. Ces colorants distinguent principalement l'ADN riche en paires nucléotides A-T (bandes G), de l'ADN riche en paires nucléotides G-C(bandes R). Les bandes G sont les bandes sombres, les claires sont les bandes R.

On obtient ainsi une succession de bandes claires et de bandes sombres dont la résolution est plus ou moins forte. (Figure I.4)

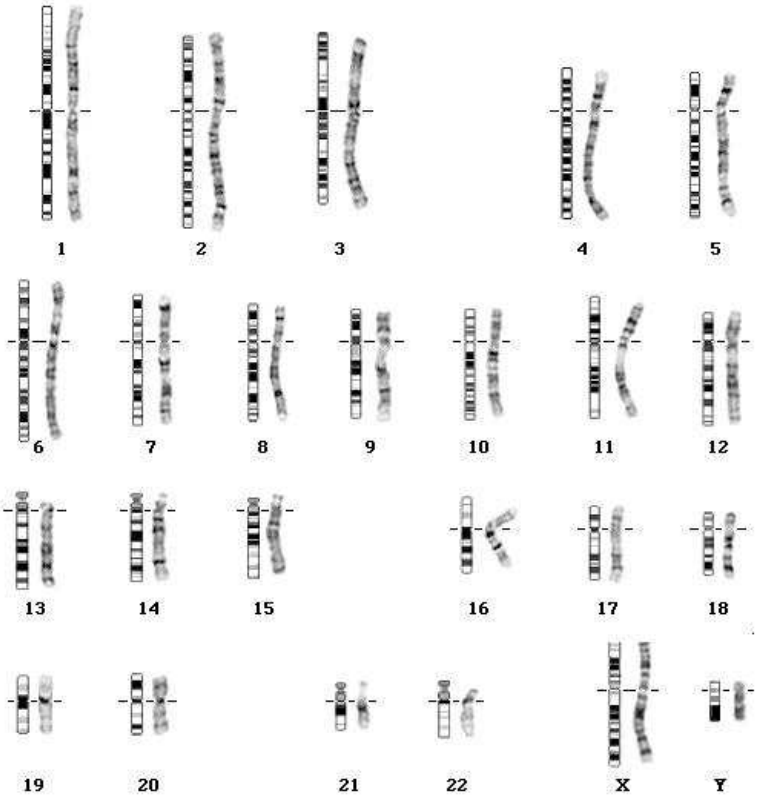


Figure I.4 Carte standard de distribution de la densité de profil de chaque chromosome

9. Les anomalies chromosomiques

On appelle anomalie chromosomique tout remaniement du nombre et / ou de la structure des chromosomes.

Ces remaniements peuvent s'observer de manière constitutionnelle (présents dès la naissance), qu'elles soient transmises par les parents ou apparues « de novo » chez le sujet porteur. Ils résultent d'un accident survenant soit au cours de la méiose, soit au cours d'une mitose. Ils peuvent impliquer un ou plusieurs chromosomes.

9.1. Les anomalies de nombre

Il s'agit le plus souvent de trisomies (présence de 3 chromosomes d'une même paire au lieu de deux [2]. Elles peuvent affecter certaines paires d'autosomes (13, 18, 21) et les chromosomes sexuels.

9.2. Les anomalies de structure

Elles résultent de cassures chromosomiques portant sur un ou plusieurs chromosomes homologues ou non [3]. Elles correspondent soit à des délétions (perte de matériel chromosomique), soit à des inversions ou à des translocations réciproques (échange de matériel chromosomique entre 2 chromosomes) ou robertsonniennes (fusion de 2 chromosomes par leurs centromères).

10. Conclusion

La cytogénétique a enregistré une avancée considérable ces deux dernières décennies. On est ainsi passé de la cytogénétique classique qui travaillait à l'échelle de la structure externe du chromosome à la cytogénétique moléculaire qui s'intéresse au séquençage des gènes.

Toutefois le caryotype reste une technique de base, de plus en plus simple à élaborer en raison des nouvelles techniques de coloration et de traitement numérique d'image, pour la détection précoce d'anomalies chromosomiques.

CHAPITRE II :

LES RÉSEAUX DE NEURONES

1. Introduction

Depuis des années, les films de science fiction tout droit venus d'Hollywood tels que «I, Robot » ont annoncé l'arrivée de l'Intelligence Artificielle (Artificial Intelligence) comme un signe avant-coureur de l'Armageddon. Toutefois, ces films ne pouvaient être aussi éloignés de la vérité. Alors qu'Hollywood nous régale de films d'horreur annonçant notre déclin imminent, une communauté de scientifiques n'ayant aucune volonté de détruire notre espèce, exploite l'Intelligence Artificielle afin de rendre notre vie plus simple, plus productive, plus longue, et d'une manière générale plus agréable.

Comment l'homme fait-il pour raisonner, parler, calculer, apprendre... ?

Un réseau de neurones artificiel est une structure composée d'entités interconnectées entre elles : les neurones. Il permet de traiter, par le biais de l'informatique, des problèmes de différentes natures que les outils classiques ont du mal à résoudre. En effet, son fonctionnement s'inspire de celui des cellules neuronales animales, et est donc différent des méthodes de calcul analytiques que l'on utilise ordinairement. Il s'avère très puissant dans des problèmes de reconnaissance, classification, approximation ou prévision.

2. Historique

C'est en 1943, dans un article resté fondamentale, McCulloch et Pitts ont émis l'idée simplificatrice du neurone formel. C'est à dire, un opérateur binaire interconnecté à ses semblables par des « synapses » excitatrices ou inhibitrices. Une assemblée de tels opérateurs en interaction devrait être capable de certains « calculs » que chacun d'eux séparément est incapable d'exécuter. [5].

Les travaux de McCulloch et Pitts n'ont pas donné d'indication sur la méthode pour adapter les poids synaptiques. Cette question au cœur des réflexions sur l'apprentissage a connu un début de réponse, grâce aux travaux du physiologiste Donald Hebb en 1949 [5], sur l'apprentissage, décrit dans son ouvrage : « The Organisation of Behaviour ». Hebb a proposé une règle simple qui permet de modifier la valeur des coefficients synaptiques, en fonction de l'activité des unités qu'ils

relient. Cette règle aujourd'hui connue sous le nom de « Règle de Hebb », est presque partout présente dans les modèles actuels, même les plus sophistiqués.

Cela resta quand même dans la théorie, jusqu'à 1957, F. Rosenblatt proposa un modèle pratique : le perceptron [6]. C'est le premier système artificiel capable d'apprendre par expérience.

En 1969, la recherche sur les réseaux de neurones a connu un coup grave : Minsky et Papert ont pu montrer lors d'une publication, quelques limitations du perceptron, notamment l'impossibilité de traiter les problèmes de non linéarité, et ont généralisé cela sur tous les réseaux de neurones [7]. De ce fait, la communauté scientifique s'est désintéressée à la recherche sur les réseaux neuronaux, et ainsi, une grande partie de l'investissement publique et privé a été déplacée sur d'autres sujets de recherche en intelligence artificielle.

Ce n'est qu'en 1982, que le physicien J.J.Hopfield, donna un nouveau modèle de réseau de neurones (complètement récurrent) [7], cet article eut du succès pour plusieurs raisons, dont la principale était de teinter la théorie des réseaux de neurones de la rigueur propre aux physiciens. Ainsi le neuronal devient un sujet d'étude acceptable, bien que le modèle de Hopfield souffrit des limitations des modèles des années 60. La recherche fut relancée et l'industrie reprit quelque intérêt au neuronal.

Depuis, les réseaux de neurones ont connu un essor considérable, et une multitude d'industries, de la télécommunication aux processus de qualité des produits s'y sont intéressés.

3. Le neurone biologique

3.1. Définition

Les cellules nerveuses appelées neurones, sont les éléments de base du système nerveux. Celui-ci en posséderait environ cent milliards.

Les neurones possèdent de nombreux points communs dans leur organisation générale et leur système biochimique avec les autres cellules. Ils présentent cependant

des caractéristiques qui leur sont propres et se retrouvent au niveau des cinq fonctions spécialisées qu'ils assurent :

- recevoir des signaux en provenance de neurones voisins,
- intégrer ces signaux,
- engendrer un influx nerveux,
- le conduire,
- le transmettre à un autre neurone capable de le recevoir.

3.2. Structure des neurones

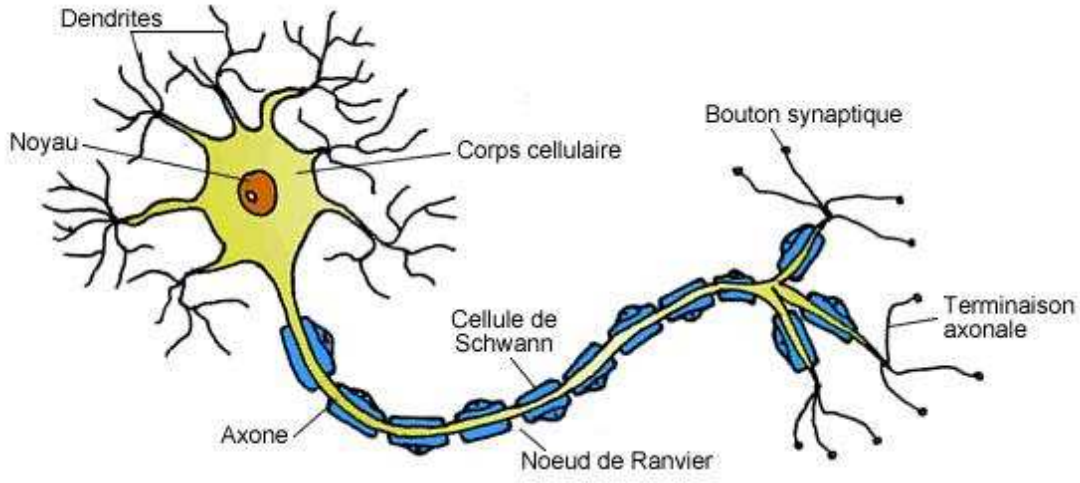


Figure II.1 Structure d'un neurone biologique.

Le neurone est constitué de trois parties (figure II.1) :

- Le corps cellulaire : il contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la synthèse des enzymes et des autres molécules qui assurent la vie du neurone [6]. Sa forme est pyramidale ou sphérique dans la plupart des cas et il fait quelques microns de diamètre.
- Les dendrites : chaque neurone possède une 'chevelure' de dendrites. Celles ci sont de fines extensions tubulaires, de quelques dixièmes de microns de diamètre et d'une longueur de quelque dizaine de microns [6]. Elles se ramifient, ce qui les amène à former une espèce d'arborescence autour du corps cellulaire. Elles sont les récepteurs principaux du neurone pour capter les signaux qui lui parviennent.

- L'axone : qui est à proprement parler la fibre nerveuse, sert de moyen de transport pour les signaux émis par le neurone [8]. Il se distingue des dendrites par sa forme et par les propriétés de sa membrane externe. En effet, il est généralement plus long (sa longueur varie d'un millimètre à plus d'un mètre) que les dendrites, et se ramifie à son extrémité là ou il communique avec d'autres neurones. Pour former le système nerveux, les neurones sont connectés les uns aux autres suivant des répartitions spatiales complexes. Les connexions entre deux neurones se font en des endroits appelés synapses où ils sont séparés par un petit espace synaptique de l'ordre d'un centième de microns.

3.3. Fonctionnement des neurones :

D'une façon simple, on peut dire que le corps cellulaire du neurone traite les courants électriques qui lui proviennent de ses dendrites, et qu'il transmet le courant électrique résultant de ce traitement aux neurones auxquels il est connecté par l'intermédiaire de son axone. Le schéma classique présenté par les biologistes et celui d'un corps cellulaire effectuant une sommation des influx nerveux transmis par ses dendrites. Si la sommation dépasse un seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone. Si la sommation est inferieure à ce seuil, le neurone reste inactif. L'influx nerveux qui se propage entre différents neurones est, au niveau de ces neurones, un phénomène électrique.

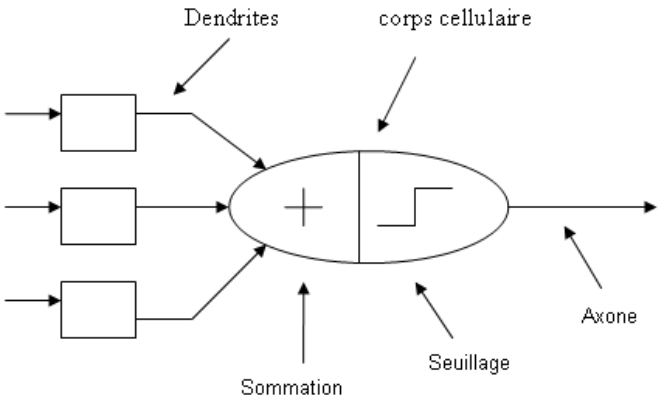


Figure II.2 Représentation d'un neurone biologique.

3.4. Informations diverses sur les neurones du cerveau humain :

- Le cerveau contient environ 100 milliards de neurones.
- On ne dénombre que quelques dizaines de catégories distinctes de neurones.
- La vitesse de propagation des influx nerveux est de l'ordre de 100m/s [9], c'est à dire bien inférieure à la vitesse de transmission de l'information dans un circuit électronique.
- On compte de quelques centaines à plusieurs dizaines de milliers de contacts synaptiques par neurone. Le nombre total de connexions est estimé à environ 10^{15} [6]
- Le nombre de neurones décroit après la naissance. Cependant, cette affirmation semble remise en question.
- On observe par contre une grande plasticité de l'axone, des dendrites et des contacts synaptiques. Celle-ci est surtout très importante après la naissance (on a observé chez le chat un accroissement des contacts synaptiques de quelques centaines à 12 000 entre le 10ème et le 35ème jour) [9]. Cette plasticité est conservée tout au long de l'existence, bien qu'affaiblie lors du processus de vieillesse, on parle alors de "rigidité synaptique".
- Les synapses entre des neurones qui ne sont pas simultanément actifs sont affaiblies puis éliminées [9].

4. Le neurone formel

Prenons un exemple de la reconnaissance de formes. Plus particulièrement, la reconnaissance des chiffres (0, 1, ..., 9). Imaginons un programme qui devrait reconnaître, depuis une image, un chiffre. On présente donc au programme une image d'un "1" manuscrit par exemple et lui doit pouvoir nous dire "c'est un 1". Supposons que les images que l'on montrera au programme soient toutes au format 200x300 pixels. On aurait alors 60000 informations à partir desquelles le programme déduirait le chiffre que représente cette image. L'utilisation principale des réseaux de neurones est justement de pouvoir, à partir d'une liste de *n* informations, pouvoir déterminer à

laquelle des *p* classes possibles appartient cette liste. Les classes dans cet exemple sont les chiffres.

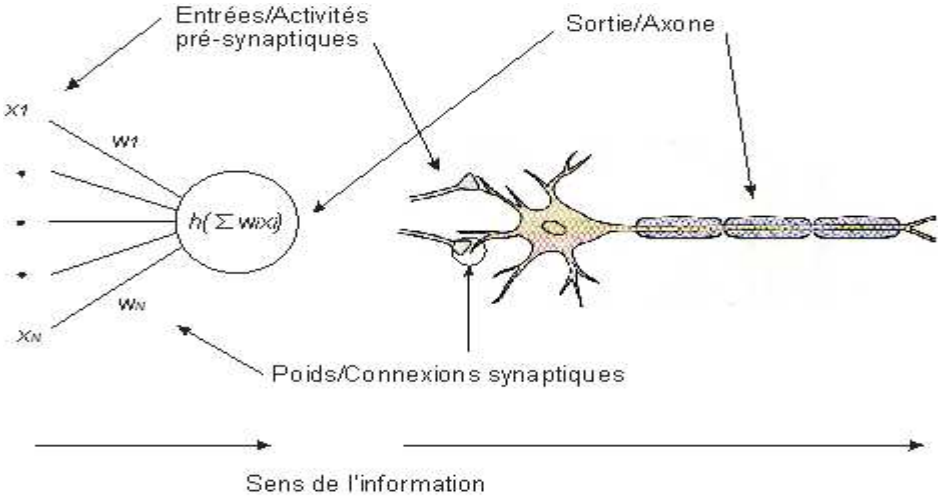


Figure II.3 Neurone formel – Neurone biologique

4.1. Les entrées :

On note x_i (i allant de 1 à *n*) les *n* informations parvenant au neurone. De plus, chacune sera plus ou moins valorisée vis à vis du neurone par le biais d'un poids. Un poids est simplement un coefficient w_i lié à l'information x_i. La i–ème information qui parviendra au neurone sera donc en fait $(w_i \times x_i)$. Il y a toutefois un "poids" supplémentaire, qui va représenter ce que l'on appelle le coefficient de biais noté w_0 .

Le neurone artificiel (qui est une modélisation des neurones du cerveau) va effectuer une somme pondérée de ses entrées plutôt que de considérer séparément chacune des informations. On définit une nouvelle donnée, *in* par :

$$\text{in} = \sum_{i=0}^{n} w_i \times x_i = \left(\sum_{i=1}^{n} w_i \times x_i\right) - w_0 \qquad (2.1)$$

C'est en fait cette donnée-là que va traiter le neurone. Cette donnée est passée à la fonction d'activation, qui fait l'objet du prochain paragraphe. C'est d'ailleurs pour ça que l'on peut parfois appeler un neurone une unité de traitement.

4.2. Fonction d'activation :

La fonction d'activation, ou fonction de transfert, est une fonction qui doit renvoyer un réel proche de 1 quand les "bonnes" informations d'entrée sont données et un réel proche de 0 quand elles sont "mauvaises". On utilise généralement des fonctions à valeurs dans l'intervalle réel [0,1]. Quand le réel est proche de 1, on dit que le neurone est actif alors que quand le réel est proche de 0, on dit que le neurone est inactif. Le réel en question est appelé la sortie du neurone et sera noté *a*.

En notant *g* la fonction d'activation, on obtient donc la formule donnant la sortie d'un neurone:

$$a = g(in) = g\left(\left(\sum_{i=1}^{n} w_i * x_i\right) - w_0\right) \qquad (2.2)$$

Il y a beaucoup de fonctions d'activations possibles, toutefois dans la pratique il y en a principalement 4 qui sont utilisées [9] :

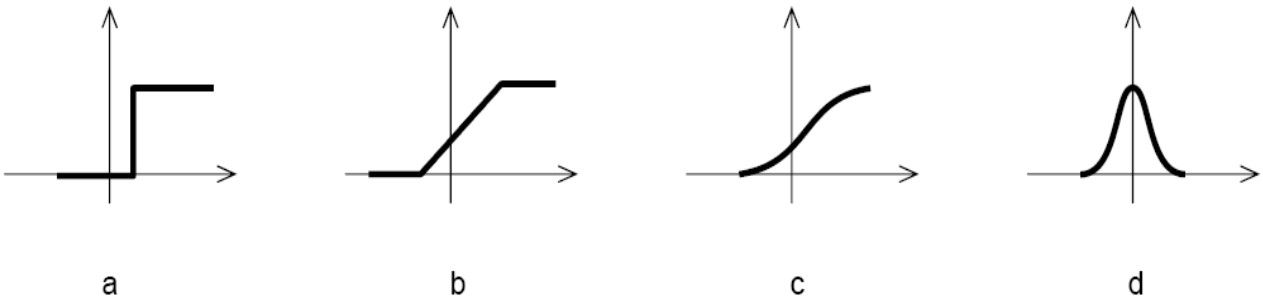


Figure II.4 Quelques fonctions d'activation couramment utilisées.
a) Fonction à seuil. c) Sigmoïde standard.
b) Linéaire par morceaux. d) Gaussienne.

4.3. Coefficient de biais

Si on prend le cas de la fonction sigmoïde définie comme l'indique la figure II.5 :

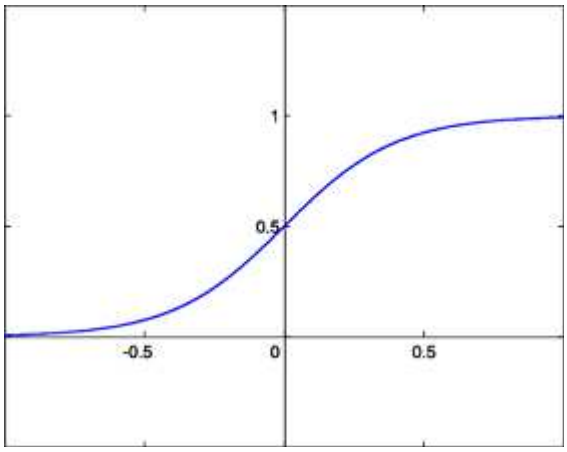


Figure II.5 Graphe de la fonction Sigmoïde.

Cette fonction d'activation possède un seuil en 0, qui vaut ½. Revenons maintenant à notre neurone et demandons-nous quand est-ce que le seuil est atteint, ou dépassé. Il est dans tous les cas atteint quand *in* vaut 0.

$$in = 0 \iff \sum_{i=1}^{n} w_i * x_i - w_0 = 0 \iff \sum_{i=1}^{n} w_i * x_i = w_0 \qquad (2.3)$$

C'est là qu'intervient réellement le coefficient de biais. Nous voyons de l'équation (2.3) que le seuil de la fonction d'activation est atteint lorsque la somme pondérée des informations d'entrée vaut le coefficient de biais.

5. Réseau de neurones artificiel

Les réseaux de neurones se distinguent par leur mode d'apprentissage et par leur topologie (leur architecture), dans le courant paragraphe, nous aborderons la structure, le fonctionnement et l'apprentissage des réseaux de neurones.

5.1. Architecture des réseaux de neurones :

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

5.1.1. les « Feed-Forward networks » :

Ce sont des réseaux à circulation de l'information vers l'avant et dans lesquels l'organisation des neurones est en couches successives [9]. Le calcul se fait en propageant les données de l'entrée vers la sortie. Dans cette catégorie on distingue les réseaux à une seule couche, les réseaux multicouches (possédant une couche d'entrée, une couche de sortie, et une, ou plusieurs couches cachées).

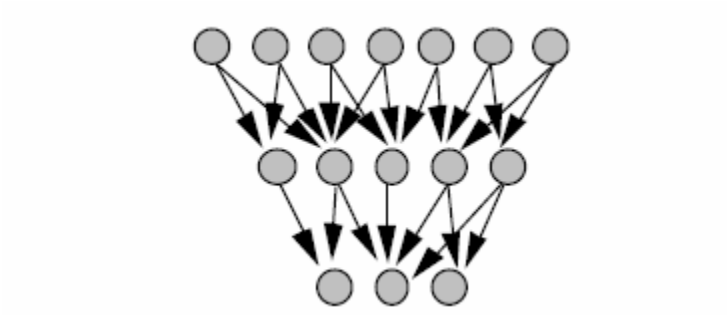


Figure II.6 Schéma d'un réseau Feed-Forward

5.1.2. les « Feed-back networks » ou les réseaux de neurones récurrents:

Les réseaux récurrents ramènent l'information en arrière par rapport au sens de propagation défini dans les réseaux du type Feed-Forward [9]. Dans cette catégorie, on trouve les réseaux de Hopfield.

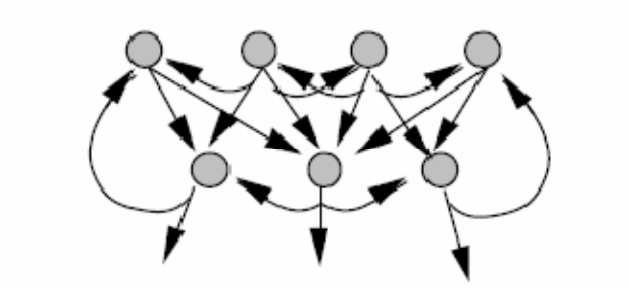


Figure II.7 Schéma d'un réseau Feed-Back

5.2. L'apprentissage :

L'apprentissage est défini comme étant la phase du développement du réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de

comportement. Dans le cas des réseaux de neurones artificiels, on ajoute souvent à la description du modèle l'algorithme d'apprentissage.

Dans un réseau de neurones, l'information est codée par les poids liés aux connexions.

L'apprentissage est réalisé par des algorithmes de calcul dont le but est d'adapter ces poids en fonction des stimuli présentés à l'entrée du réseau. Une fois l'apprentissage fini, les poids ne sont plus modifiés, c'est alors la phase d'utilisation. Les procédures d'apprentissage peuvent être classées en deux catégories :

• L'apprentissage supervisé.

• L'apprentissage non supervisé.

5.2.1. L'apprentissage supervisé :

Dans la phase de développement, un système qui connait parfaitement la sortie correcte (désirée) du stimulus présent à l'entrée du réseau, va guider le réseau en lui indiquant à chaque étape le bon résultat, l'apprentissage dans ce cas consiste à comparer le résultat obtenu avec le résultat désiré, puis on ajuste les poids synaptiques pour diminuer la différence.

Exemple d'apprentissage supervisé : l'algorithme de rétro-propagation du gradient (RPG).

5.2.2. L'apprentissage non supervisé :

Dans ce cas on fournit au réseau (autonome) une quantité suffisante d'exemples, et c'est au réseau de dégager les régularités automatiquement [5].

Exemple d'apprentissage non supervisé : la carte auto organisatrice de Kohonen.

5.3. Les réseaux multicouches :

Les réseaux multicouches sont aujourd'hui les modèles les plus employés, plusieurs couches de traitement leurs permettent de réaliser des associations non linéaires entre l'entrée et la sortie. L'algorithme utilisé pour son apprentissage est celui de la rétro-propagation du gradient.

5.3.1. Structure du réseau multicouche :

Les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales.

Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc de définir les concepts de neurone d'entrée, neurone de sortie.

Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelés couches cachées.

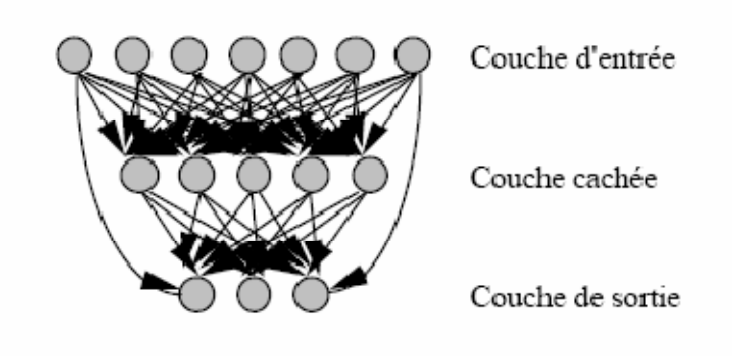


Figure II.8 Schéma d'un réseau multicouche

5.3.2. Apprentissage : l'algorithme de retro-propagation du gradient :

L'apprentissage est supervisé, l'algorithme de retro propagation du gradient, est un algorithme itératif, conçu pour minimiser l'erreur entre la sortie obtenue du réseau multicouche et la sortie désirée. Cette minimisation est réalisée par une configuration adéquate des poids.

L'algorithme :

Les poids synaptiques $W_{ji}^{(n)}$ du réseau multicouche sont au préalable initialisés avec des valeurs aléatoires, généralement comprises entre $[\frac{-0.5}{C\_inf}, \frac{+0.5}{C\_inf}]$, ou

C_inf représente le nombre de neurones dans la couche directement inferieure, et *n* le numéro de la couche [10].

On considère ensuite un ensemble de données qui vont servir à l'apprentissage, l'algorithme de retro-propagation du gradient se présente comme suit :

1- soit un échantillon X, que l'on met à l'entrée du réseau de neurones, la sortie désirée pour cet échantillon est Z.

2- on propage l'information en avant, dans les couches du réseau, la propagation vers l'avant se calcule à l'aide de la fonction d'activation *g* :

$$x_j^{(n)} = g^{(n)}(h_j^{(n)}) = g^{(n)}\left(\sum_k w_{jk}^{(n)} \times x_k^{(n-1)}\right) \qquad (2.4)$$

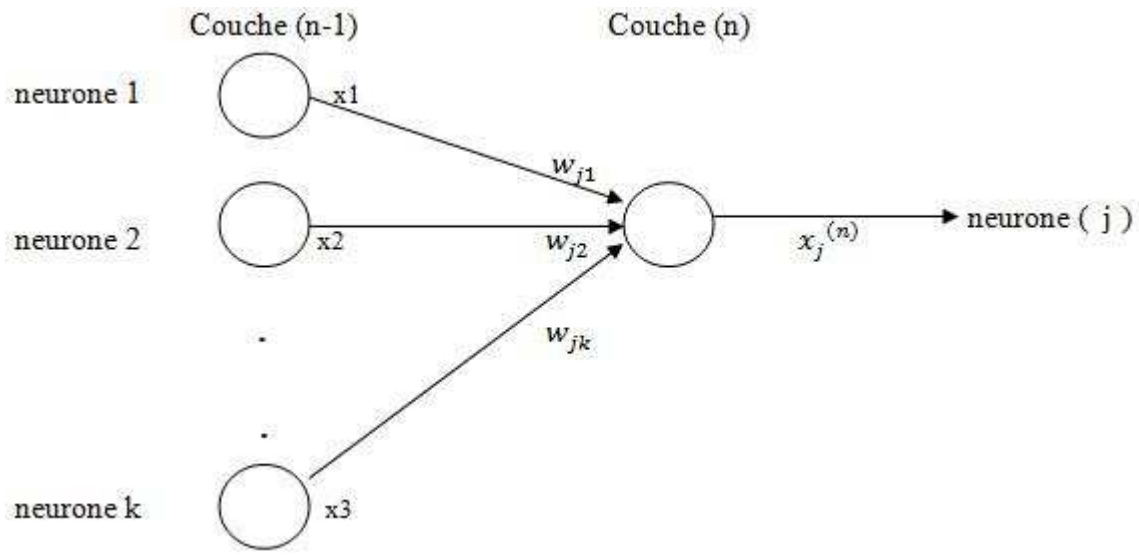


Figure II.9 Schéma illustratif de l'algorithme RPG.

3- lorsque la propagation vers l'avant est terminée, on obtient à la sortie le résultat Y.

4- on calcule alors l'erreur entre la sortie donnée par le réseau « Y » et la sortie désirée Z pour cet échantillon, pour chaque neurone i dans la couche de sortie, on calcule :

$$E = \frac{1}{2} \times \sum_i (Y_i - Z_i)^2 \qquad (2.5)$$

5- Si l'erreur n'est pas acceptable, on calcule :

$$e_i^{(sortie)} = g'^{(sortie)}(h_i^{(sortie)}) \times (Z_i - Y_i) \qquad (2.6)$$

Puis on propage l'erreur vers l'arrière par la formule suivante :

$$e_j^{(n-1)} = g'^{(n-1)}(h_j^{(n-1)}) \times (\sum_i w_{ij}^{(n)} \times e_i^{(n)}) \qquad (2.7)$$

6- On met à jour les poids synaptiques :

$$\Delta w_{ij}^{(n)} = \delta \times e_i^{(n)} \times x_j^{(n-1)} \qquad (2.8)$$

Où δ : représente le taux d'apprentissage du réseau (compris entre 0 et 1).

6. Les réseaux de Kohonen

Teuvo Kohonen a proposé dès 1982 un algorithme [11] qui produit une carte d'organisation topologique. Le processus ne dépend que des entrées et ne nécessite pas l'intervention d'un superviseur (mode d'apprentissage non supervisé), on parle dans ce cas d'auto-organisation. D'ailleurs on désigne souvent le réseau de kohonen par le terme anglais *self organizing map* ou SOM (carte auto organisatrice ou auto adaptative).

Sa fonction principale est de faire correspondre les éléments de l'espace d'entrée (généralement de grande dimension) avec des unités (neurones) ordonnées sur une carte – qui est une représentation graphique où chaque unité est entourée de ses voisines, les voisinages ayant été définis a priori. Le résultat est une fonction de l'espace des entrées vers l'ensemble des unités.

L'application la plus courante des cartes SOM est la classification de l'espace d'entrée, où l'on définit une notion de voisinage entre les classes qui n'est pas prise en compte par les méthodes de classification classiques.

La figure II.10 illustre un tel réseau dans sa configuration la plus courante, le cas bidimensionnel. L'entrée du réseau, unique et commune à tous les neurones, est le vecteur de caractéristiques X. A chaque neurone sont associés des coordonnées (x,y) indiquant sa position sur la carte, ainsi qu'un vecteur de poids synaptiques W, appelé ainsi par analogie avec les synapses rencontrées dans le cerveau.

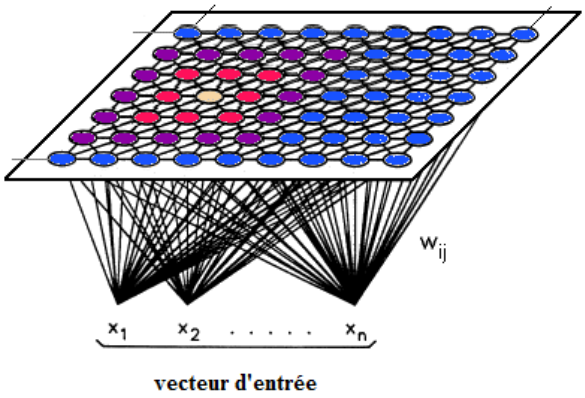


Figure II.10. Carte auto-organisatrice

6.1. Idée de base

Les réseaux de Kohonen sont inspirés de l'organisation des neurones à l'intérieur du cerveau humain, dans lequel les neurones sensibles à des stimuli proches, se trouvent topologiquement dans le même endroit. L'idée de Kohonen a donc été d'essayer de projeter des stimuli voisins, situés dans un espace multidimensionnel, dans un espace géométrique beaucoup plus simple, mono ou plus fréquemment bidimensionnel, en conservant au mieux la topologie.

Techniquement, la carte réalise une quantification vectorielle de l'espace d'entrée, cela signifie diviser l'espace en zones, dites zones de Voronoi, et assigner à chaque zone un point significatif appelé vecteur référent.

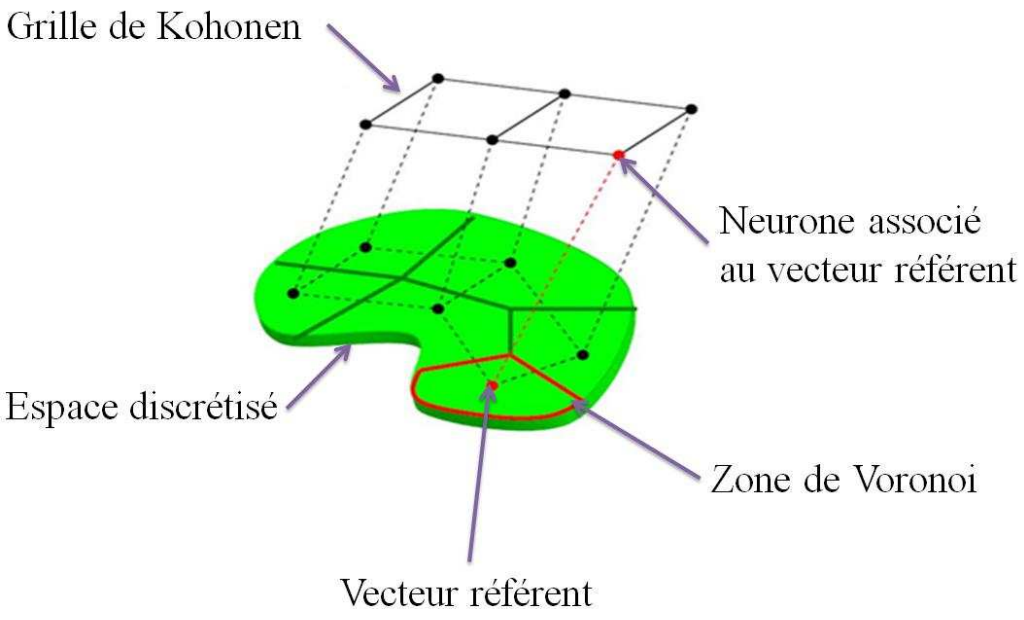


Figure II.11 Discrétisation de l'espace d'entrée par la carte SOM

6.2. La notion de voisinage :

L'algorithme de Kohonen est un processus d'auto-organisation très puissant. Il contient une notion de voisinage entre les J classes que l'on peut symboliser par des liens entre J unités disposées en réseau. Dans la plupart des cas, cette structure est de dimension deux, on parle alors de grille. Ce type de représentation est illustré par la figure II.12 dans laquelle les unités proches correspondent à des classes voisines définies naturellement.

Pour repérer une unité sur une grille on numérote les unités de 1 à J et on affecte à l'unité *j* ses coordonnées cartésiennes *(x , y)* sur la carte.

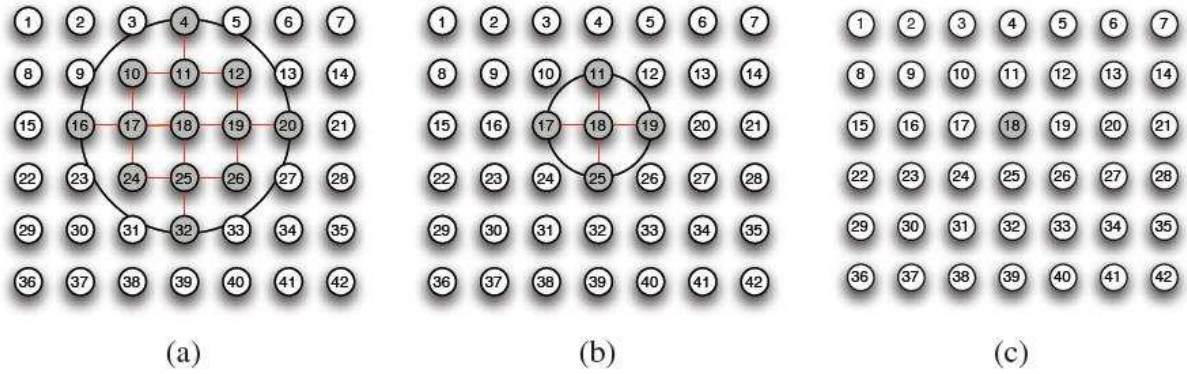


Figure II.12 Topologie de voisinage pour une carte à deux dimensions : $(a) r_{18} = 2$, $(b) r_{18} = 1$, $(c) r_{18} = 0$.

On définit le voisinage de rayon r d'une unité j_0, noté $V_r(j_0)$, comme l'ensemble des unités *j* situées sur le réseau à une distance inférieure ou égale à r.

6.3. Algorithme de Kohonen :

6.3.1. Principe :

Après une initialisation aléatoire des valeurs de chaque neurone on soumet une à une les données à la carte auto organisatrice. Selon les valeurs des neurones, il y en a un qui répondra le mieux au stimulus, celui dont la valeur sera la plus proche de la donnée présentée. Alors ce neurone sera gratifié d'un changement de valeur pour qu'il réponde encore mieux à un autre stimulus de même nature que le précédent. On gratifie aussi un peu, les neurones voisins du gagnant avec un facteur multiplicatif du gain inférieur à un. Ainsi, c'est toute la région de la carte autour du neurone gagnant qui se modifie.

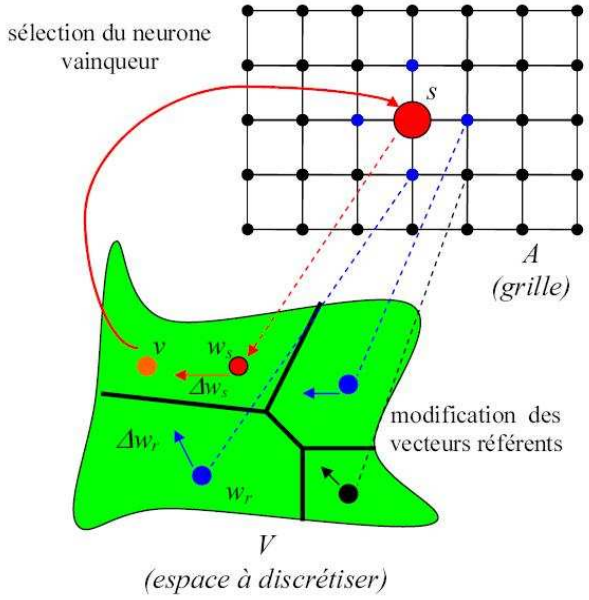


Figure II.13 Représentation de l'algorithme de Kohonen

La Figure II.13 illustre le principe de l'algorithme d'auto-organisation, chaque neurone a un vecteur référent qui le représente dans l'espace d'entrée. Un vecteur d'entrée *v* est présenté. *v* sélectionne le neurone vainqueur *s*, le plus proche dans l'espace d'entrée. Le vecteur référent du vainqueur w_s est rapproché de *v*. Les vecteurs référents des autres neurones, voisins du neurone gagnant, sont aussi déplacés vers *v*, mais avec une amplitude moins importante.

6.3.2. Algorithme :

Comme on l'a mentionné précédemment, le réseau de kohonen est composé de deux couches, la première constitue l'entrée du réseau et la seconde est la sortie. Le réseau réalise une liaison entre une entrée à I composantes et un ensemble J de sorties par l'intermédiaire de $I \times J$ poids.

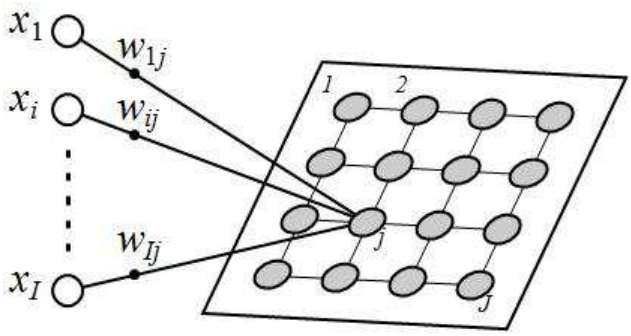


Figure II.14 Carte de Kohonen

L'espace d'entrée E est de dimension I, une entrée étant un vecteur $X = (x_1, x_2, \dots, x_I)$. Chaque composante x_i est reliée aux J neurones de sortie, par J coefficients w_{ji}. Chaque sortie j peut être donc considérée comme porteuse d'un vecteur image $w_j = (w_{j1}, w_{j2}, \dots, w_{jI})$.

L'algorithme d'auto-organisation de Kohonen se présente ainsi :

- Initialisation des poids synaptiques w_{ji} , par des petites valeurs aléatoires.
- Initialisation du rayon de voisinage r.
- Tant que la condition d'arrêt (à définir) n'est pas vérifiée (c'est-à-dire tant que $n<n_{max}$) à chaque itération n :
  - On présente à l'entrée du réseau, l'observation x = x (n) extraite au hasard de la base de données.
  - On détermine le neurone gagnant J* le plus prés de x au sens de la distance choisie, c'est-à-dire, celui qui réalise :

$$\|x - w_{j*}\| = \min\|x - w_j\| \qquad (2.9)$$

  - On détermine les voisins de J* suivant la règle de voisinage à l'itération n.
  - La mise à jour : on modifie les poids synaptiques w_{ji} par les transformations suivantes :

$$W_{ji}(n+1) = W_{ji}(n) + \alpha(n) \times [\, x_i(n) - w_{ji}(n) \,] \qquad (2.10)$$

$$\text{avec } \alpha(n) = \begin{cases} \eta(n) & \text{si } j = j* \\ \eta(n) \times h(j*, n) & \text{si } j \in V(j*, r(n)) \\ 0 & \text{sinon} \end{cases} \qquad (2.11)$$

  - n=n+1.

Où $\eta(t)$ correspond au taux d'apprentissage, c'est une fonction décroissante dans le temps.

$V(j*, r(n))$ Correspond au voisinage autour de l'unité gagnante $j*$.

La fonction $h(j*, n)$ décrit comment les neurones dans la proximité du vainqueur $j*$ sont entraînés dans le mouvement de correction. On utilise en général :

$$h(j*, n) = \exp\left(-\frac{(u_j - u_{j*})^2}{r(n)^2}\right) \qquad (2.12)$$

u_j : est un vecteur qui décrit la position du j-ième neurone sur la carte auto-organisatrice.

u_{j*}:est un vecteur qui décrit la position du neurone gagnant sur la carte auto-organisatrice.

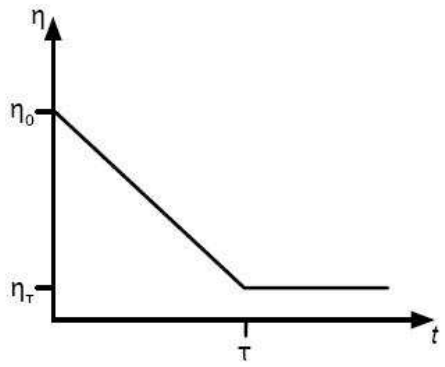


Figure II.15 Exemple d'une fonction taux d'apprentissage

Pendant la phase d'entraînement, un vecteur $X = (x_1, x_2, \dots, x_I)$ est présenté aux J noeuds de la mémoire contenant les vecteurs $W_j = (w_{j1}, w_{j2}, \dots, w_{jI})$ initialisés à des valeurs aléatoires. Les vecteurs des poids de la mémoire sont tous mis à jours selon la relation (2.10). La mise à jour des voisins dépend de la fonction $h(j*, n)$. Le résultat obtenu après la phase d'entraînement est une mémoire contenant un ensemble de vecteurs de poids $W_j = (w_{j1}, w_{j2}, \dots, w_{jI})$ affectés aux nœuds j, $j \in [1, J]$.

Pendant la phase de classification, on présente un vecteur inconnu $X = (x_1, x_2, \dots, x_I)$ et on détermine le noeud le plus proche en termes de distance. Le vecteur X portera l'étiquette de la classe du neurone gagnant.

6.4. Particularités de l'algorithme de Kohonen :

Cet algorithme présente des opérations simples, il est donc très léger en termes de coût de calculs.

L'algorithme de kohonen a la particularité d'être robuste au sens où le résultat ne peut être grandement modifié par l'ajout d'un nouvel élément à la base de données, bien sûr si celui-ci n'est pas trop extravagant.

L'un des points forts de la SOM par rapport à l'apprentissage supervisé, c'est qu'elle s'auto-organise, donc est autonome et dynamique.

Les ancêtres des cartes auto-organisatrices, les algorithmes comme "k-moyennes", réalisent la discrétisation de l'espace d'entrée en ne modifiant à chaque cycle d'adaptation qu'un seul vecteur référent. Leur processus d'apprentissage est donc très long. L'algorithme de Kohonen profite des relations de voisinage dans la grille pour réaliser une discrétisation dans un temps très court. Malheureusement, le voisinage dans les cartes auto adaptatives est fixe, et une liaison entre neurones ne peut être cassée même pour mieux représenter des données discontinues. Comme solution, nous avons proposé de réduire le rayon de voisinage après un certain nombre d'itérations.

7. Conclusion

Les réseaux de neurones artificiels constituent un moyen de modélisation et de résolution de problèmes dans plusieurs domaines notamment la reconnaissance de formes (images ou signaux), le diagnostic, la traduction automatique, la compréhension du langage etc.….

De ce fait, la véritable exploitation de ces réseaux serait – à notre avis – dans le cadre d'automatisation de longues tâches, éventuellement itératives, nécessitant un certain niveau d'intelligence forcément peu avancé.

Reconnaissant leur limite, mais connaissant leur avantage, le directeur du Centre de Théorie de la Cornell University, le physicien Malvin H. Kalos dit : "Des humains avec les outils, seront beaucoup plus performants que des humains sans les outils".

Pour notre système, nous avons opté pour un réseau de Kohonen car il semble bien adapté au problème de classification, C'est un bon classificateur, il est en effet, très utilisé dans les domaines de la médecine et de l'industrie.

CHAPITRE III :

EXTRACTION DES PARAMÈTRES CHROMOSOMIQUES

1. Introduction

Notre objectif, rappelons-le, est de concevoir un système capable d'établir un caryotype à partir d'une image de mitose. Pour ce faire, nous devons au préalable commencer par extraire, à partir de l'image de mitose les trois paramètres chromosomiques : longueur, indice centromérique et densités de profil, que nous allons utiliser par la suite pour la classification.

Pour faciliter l'extraction de l'information, il est nécessaire de supprimer les facteurs de bruit, pour une meilleure lisibilité. Nous examinerons donc au cours de ce chapitre les différentes transformations applicables à de telles images, mais avant cela, des notions élémentaires du traitement d'image sont incontournable.

Nous décrirons par la suite les principales étapes d'extraction des paramètres chromosomiques.

2. Notions fondamentales

2.1. L'image numérique

L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle [12].

2.2. Notion de pixel

Une image est constituée d'un ensemble de points appelés pixels (le mot pixel est l'abréviation de PICture ELement). Si le bit est la plus petite unité d'information que peut traiter un ordinateur, le pixel est le plus petit élément que peuvent manipuler les matériels d'affichage ou d'impression.

Etant donné que l'écran effectue un balayage de gauche à droite et de haut en bas, on désigne par les coordonnées [0,0] le pixel situé en haut à gauche de l'image, cela signifie que les axes de l'image sont orientés de la façon suivante : l'axe X est orienté de gauche à droite, et l'axe Y est orienté de haut en bas.

2.3. L'intensité

L'intensité ou luminance est l'intensité lumineuse par unité de surface perpendiculaire à la direction d'émission.

2.4. Image à niveau de gris

Le niveau de gris est le codage correspondant à la quantification de l'intensité des pixels en une valeur numérique. Chaque pixel, codé sur huit bits, peut prendre des valeurs allant du noir (la valeur 0) au blanc (la valeur 255), en passant par un nombre fini de niveaux de gris.

2.5. Le contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image [13].

Si L1, L2 sont les degrés de luminosité respectivement des deux zones voisines A1, A2 d'une image, alors le contraste C est défini comme suit :

$$C=\frac{|L1-L2|}{L1+L2} \qquad (3.1)$$

2.6. L'Histogramme

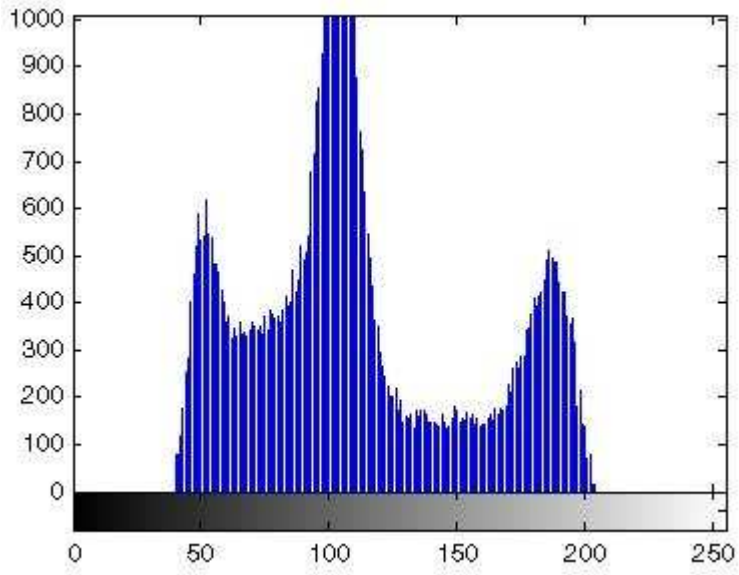


Figure III.1 Histogramme d'une image à niveaux de gris

L'histogramme est une représentation graphique qui illustre la fréquence d'apparition de chaque niveau de gris dans une image. Il nous permet d'extraire beaucoup d'information

de l'image, par exemple une image ayant une courbe d'histogramme qui n'occuperait que la partie centrale ne serait pas (ou peu) contrastée.

2.7 Le Bruit dans une image

Le bruit dans une image est caractérisé par une variation brusque de l'intensité d'un pixel par rapport à ses voisins, il provient généralement de l'éclairage des dispositifs optiques et électroniques du capteur.

3. La Binarisation

Binariser une image à niveau de gris consiste à attribuer à chaque pixel de luminosité L la valeur 0 (noir) ou 1 (blanc), la dynamique de l'image est alors réduite a 2 niveaux. La mise en œuvre la plus simple est de fixer un seuil S, typiquement 127, d'attribuer à chaque pixel de luminosité L la valeur L' tel que:

L' = 0 si L dans [0 , S[

L' = 255 si L dans [S ,255].

On peut recourir à une binarisation, qui est considérée comme une opération élémentaire, pour la facilité de manipulation d'images binaires et le faible espace mémoire requis pour les stocker.

4. Filtrage spatial

Par opposition au traçage de l'histogramme ou la binarisation, qui sont des opérations ponctuelles sur tous les pixels de l'image, les filtrages spatiaux sont des produits de convolution qui mettent en jeu le voisinage de chaque pixel.

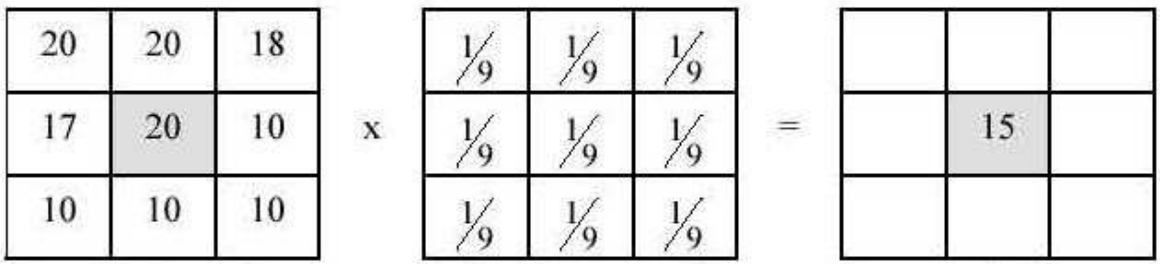


Figure III.2 Produit de convolution : exemple d'un filtrage spatial

L'application du filtre consiste à déplacer la fenêtre de convolution (généralement 3×3 ou 5×5) dans l'image ligne par ligne et colonne par colonne, la valeur du pixel central sera remplacée par le résultat du produit de convolution. Une nouvelle image est ainsi générée.

4.1. Filtre passe bas

Les filtres passe bas sont destinés à éliminer le bruit. L'inconvénient de ce type de filtrage est qu'il a tendance à effacer les limites des objets. La fenêtre de convolution illustrée dans la figure III.2 représente un exemple de filtre passe bas appeler filtre moyen.

4.2. Filtre passe haut

Contrairement au filtre passe bas, le filtre passe haut n'élimine pas le bruit, il augmente le contraste de l'image et permet ainsi la mise en évidence des contours.

5. La segmentation

La segmentation est une opération de traitement d'image qui vise à séparer une image en régions ayant des caractéristiques semblables selon deux approches :

- La première approche sépare les objets par détection de contours (détection de discontinuités locales).
- La seconde détecte les zones de l'image dont les pixels présentent des caractéristiques d'homogénéité.

6. La squelettisation

La squelettisation est une étape essentielle de la reconnaissance de forme, elle a pour but de décrire chaque objet par des lignes infiniment fines centrées dans la forme d'origine.

Le squelette est généralement défini comme étant l'ensemble des lignes médianes c'est-à-dire l'ensemble des points équidistants de deux points de la frontière. Il faut noter que la squelettisation ne s'applique que sur des images binarisées.

La figure III.3 montre le squelette de quelques objets que nous avons obtenu par Matlab.

Figure III.3 Squelette obtenu par Matlab

7. L'extraction des paramètres chromosomiques

Tout d'abord, on commence par l'amélioration de contraste et l'élimination du bruit contenu dans l'image de mitose, puis on fait une segmentation permettant l'étiquetage des chromosomes. Enfin on effectue une squelettisation de tous les objets étiquetés dans la phase de segmentation. Les images présentées ci-après illustrent ces opérations :

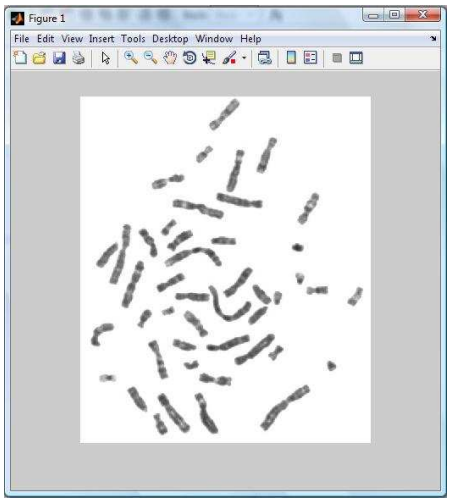


Figure III.4 Image de mitose d'une cellule d'homme normale.

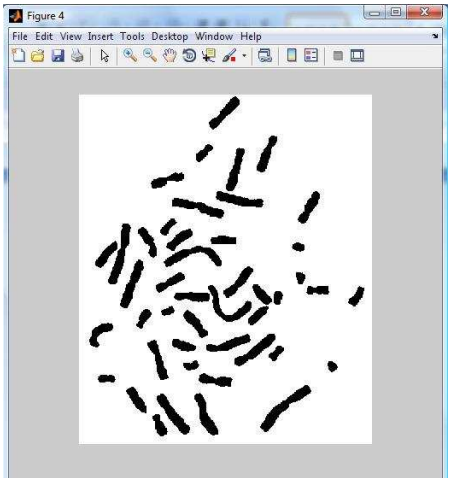


Figure III.5 Image de mitose binarisée.

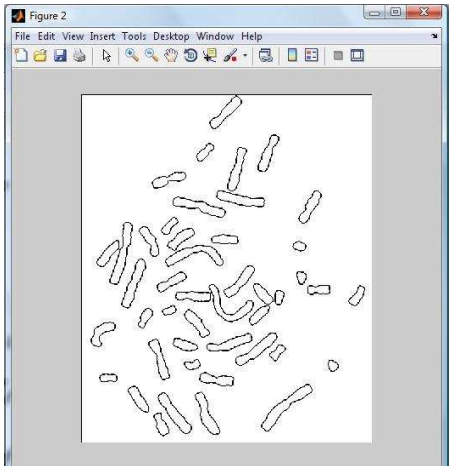


Figure III.6 Image de mitose segmentée.

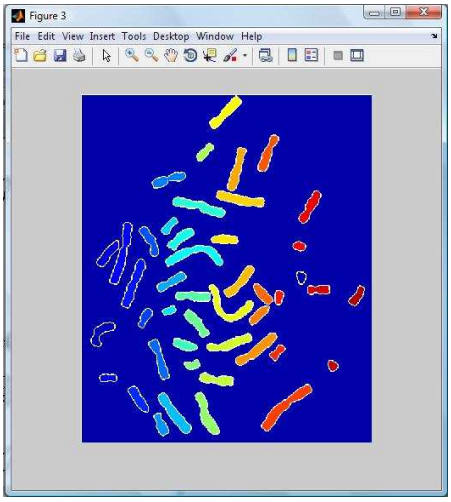


Figure III.7 Etiquetage des différents chromosomes.

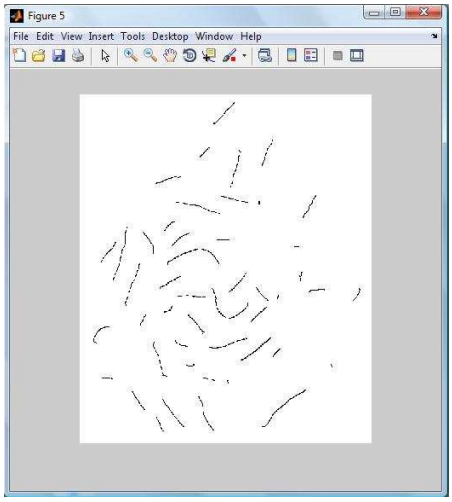


Figure III.8 Image de mitose squelettisée.

Après la squelettisation des chromosomes et un traitement de mise en forme, nous pouvons passer à l'extraction des paramètres de ceux-ci.

- Calcul de la longueur de chaque chromosome : pour calculer la longueur d'un chromosome donné on compte le nombre de pixels de son squelette.
- Calcul de l'indice centromérique : donné par la formule suivante :

$$Ic = \frac{p}{p+q} \qquad (3.2)$$

Avec : p =longueur du bras court et q = longueur du bras long.

Pour calculer ces deux longueurs, on doit d'abord détecter le centromère. Celui-ci se présente sur le chromosome comme une zone d'étranglement, cette zone aura le plus petit diamètre sur le chromosome. Le squelette nous permet de parcourir l'objet en cherchant cette zone comme illustré sur la figure III.9 :

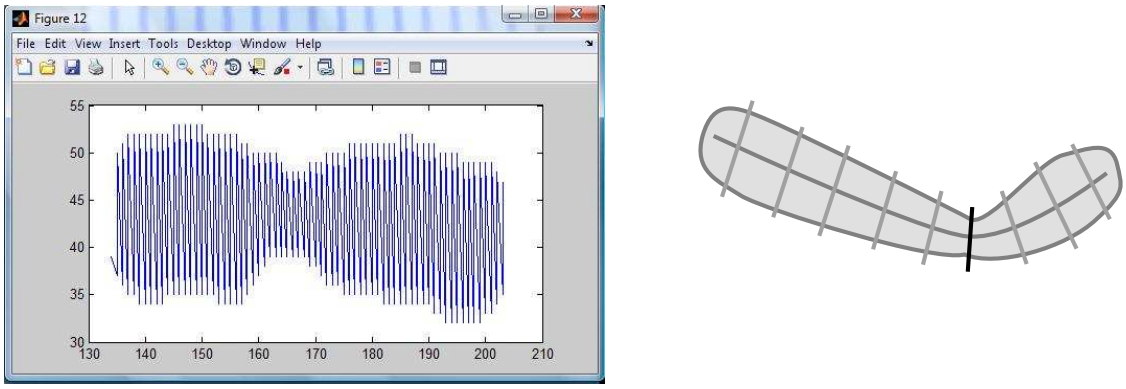


Figure III.9 Traçage de segments fictifs tout au long du squelette.

- Extraction des densités de profil de chaque chromosome : Nous avons fixé le nombre de densités de profil d'un chromosome au nombre de pixels de son squelette. En parcourant le squelette de bout en bout, la valeur de chaque densité est une moyenne des intensités des pixels appartenant à un segment de droite, passant par le pixel courant, et perpendiculaire au squelette en ce point. La figure III.10 représente une courbe de densité de profil tirée du chromosome qui porte l'étiquette 37.

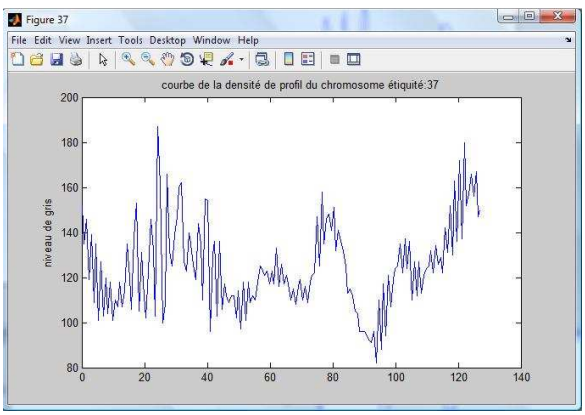


Figure III.10 Courbe de densité de profil.

8. Conclusion

Le traitement d'images est un domaine si vaste, qu'au cours de ce chapitre, nous n'avons évoqué qu'une infime partie de ses concepts de base qui constitue cependant, une partie forcément primordiale au projet.

Les connaissances acquises à travers la présentation des notions fondamentales en traitement d'images numériques, permettent déjà de concevoir un module de traitement pour les images de mitoses, par l'utilisation des techniques classiques.

CHAPITRE IV :

PRÉSENTATION DU SYSTÈME RÉALISÉ

1. Introduction

Dans ce chapitre nous allons expliquer l'architecture proposée pour l'implémentation hardware de l'algorithme de Kohonen sur le circuit FPGA, utilisé pour l'établissement du caryotype humain.

Le système que nous avons réalisé permet d'établir le caryotype et par conséquent de déterminer d'éventuelles anomalies de nombre.

Les entrées (ou les données à traiter) du système sont, dans un premier temps, les longueurs et les indices centromériques des chromosomes, supposés déjà extraits à partir d'une image de chromosomes en phase de mitose prise par un microscope photonique, analysée et prétraitée.

En sortie, le système doit retourner des indices correspondant chacun à une paire de chromosomes du caryotype selon la classification de l'ISCN. Les indices sont donc compris entre 1 et 24. Les chromosomes dits sexuels X et Y sont repérés respectivement par les indices 23 et 24.

Chaque indice va donc correspondre à une classe et donc à une unité (neurone) dans le réseau de Kohonen.

Pour la première partie de notre travail, seules les caractéristiques dites morphologiques (taille et indice centromérique) sont prises en compte pour l'établissement du caryotype.

Au cours de ce chapitre, nous présenterons les différentes phases de classification, on peut ainsi distinguer les deux phases suivantes :

- Phase d'apprentissage.
- Phase d'utilisation ou de classification.

qui seront détaillées dans la section suivante.

2. Description de l'architecture

2.1. Phase d'apprentissage :

Cette phase, élémentaire pour tout type de réseau de neurones, est constituée de 3 blocs principaux :

- Bloc de mémorisation des paramètres chromosomique ;
- bloc de calcul de distance ;
- bloc de mise à jour.

Le schéma de la figure IV.1 explique brièvement en quoi consiste cette phase.

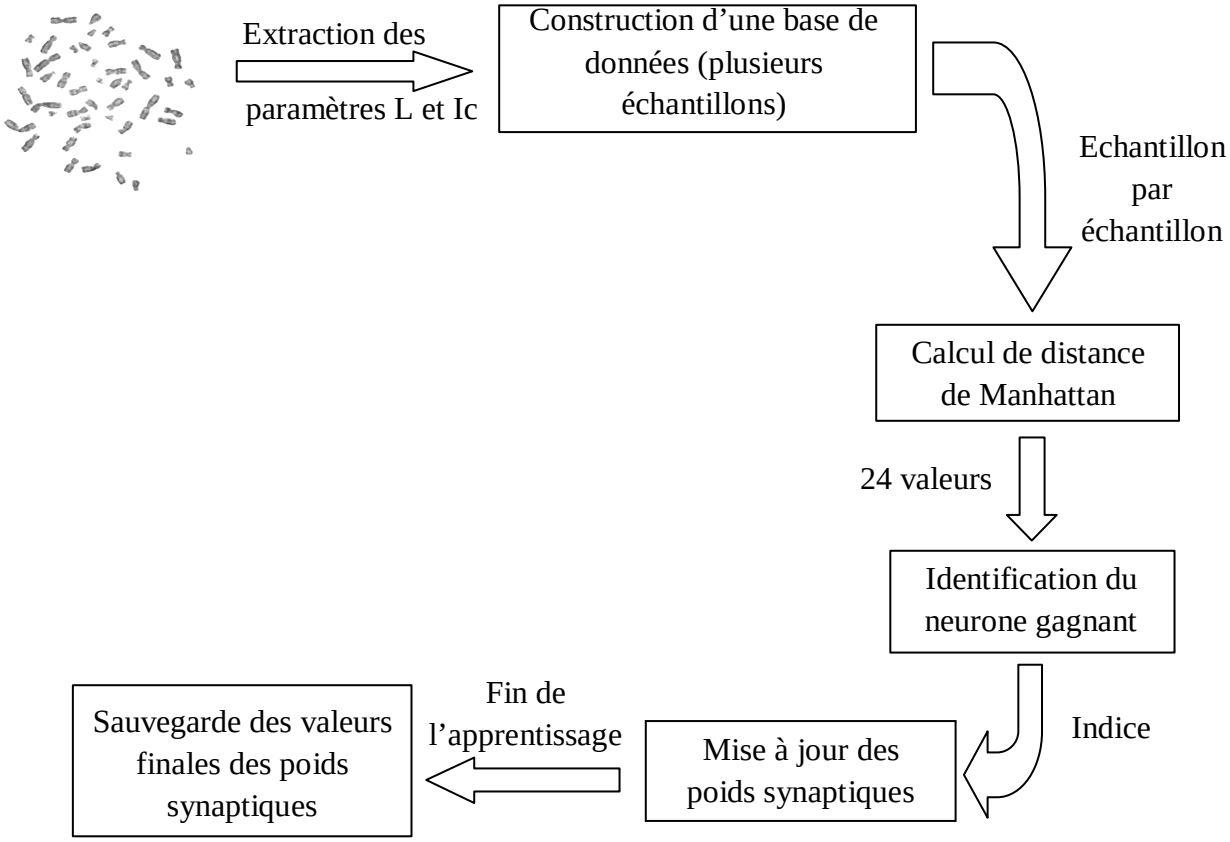


Figure IV.1 Différentes étapes de phase d'apprentissage.

2.1.1. Bloc de mémorisation

Le bloc de mémorisation des paramètres chromosomiques est constitué principalement de deux Rom : Rom_L et Rom_Ic qui contiennent un volume important de données (longueurs et indices centromériques respectivement extraits auparavant à partir d'un module de traitement d'image, détaillé dans le chapitre précédent) en plus d'un générateur d'adresse pseudo-aléatoire (figure IV.2).

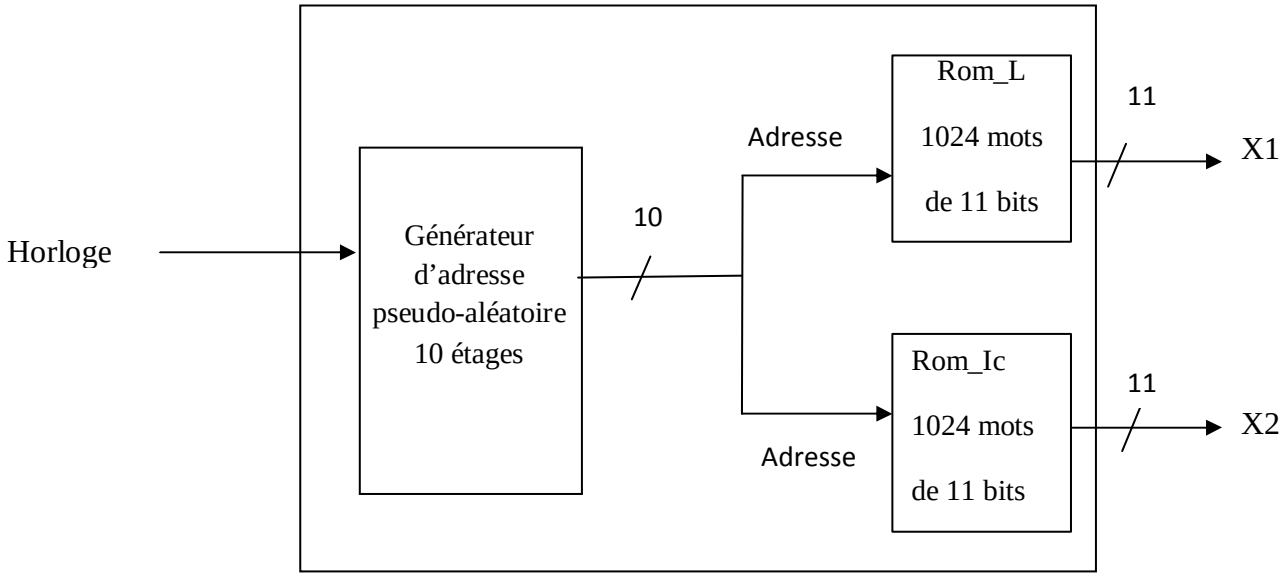


Figure IV.2 Architecture du bloc de mémorisation des paramètres chromosomiques

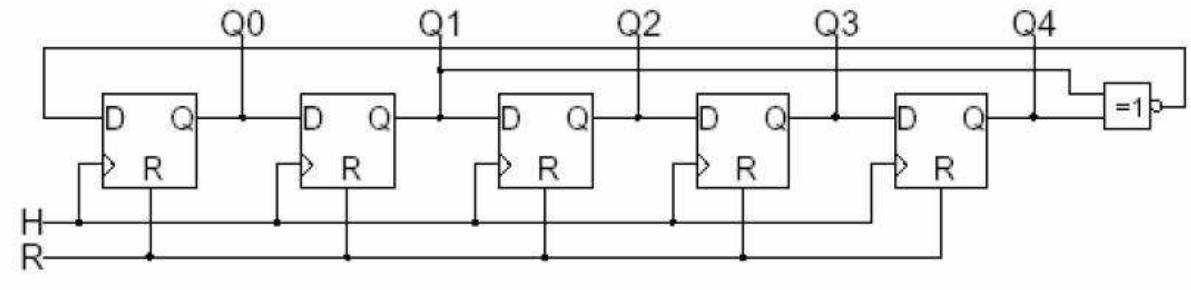
Pour la représentation des données en mémoire, nous avons utilisé un codage en virgule fixe, les données sont ainsi représentées sur 11 bits. Un simple script Matlab qui nous a permis de coder la base de données et la rendre compatible avec le type de variable que manipule l'outil ISE est fourni en annexe A.

Le bloc de mémorisation des paramètres chromosomiques a pour rôle de présenter en sortie, à chaque itération un vecteur X de dimension 2, les deux composantes de ce vecteur sont tirées respectivement des Rom_L et Rom_Ic. Dans la suite de notre exposé on notera x1 la donnée extraite de la rom Rom_L, et x2 la donnée extraite de la rom Rom_Ic.

Selon l'algorithme de Kohonen on doit présenter à chaque itération une nouvelle entrée X(n) choisie aléatoirement à partir de la base de données. La

génération d'adresse d'une manière aléatoire est pratiquement impossible, on a eu donc recours à un générateur d'adresse pseudo-aléatoire.

Dans notre projet, nous avons décrit un générateur de séquence binaire pseudo aléatoire à 10 étages car les deux rom sont composées de 1024 valeurs, nous avons donc besoin d'un bus d'adresse de 10 lignes.



Figue III.3 Générateur de séquence binaire pseudo aléatoire à 5 étages.

Nous avons utilisé pour le réaliser, dix bascules D et une porte Not Xor. La longueur maximale de la séquence d'un tel générateur est de 2^N-1 (N étant le nombre d'étages). Pour notre cas, N=10 donc : $2^{10} - 1 = 1023$: ce qui fait 1023 séquences binaires aléatoires.

Sa description est attachée dans l'annexe A, le résultat de la simulation de ce générateur d'adresse est représenté dans la figure IV.4. On voit qu'à chaque cycle d'horloge, il génère une adresse aléatoire comprise entre 0 et 1023.

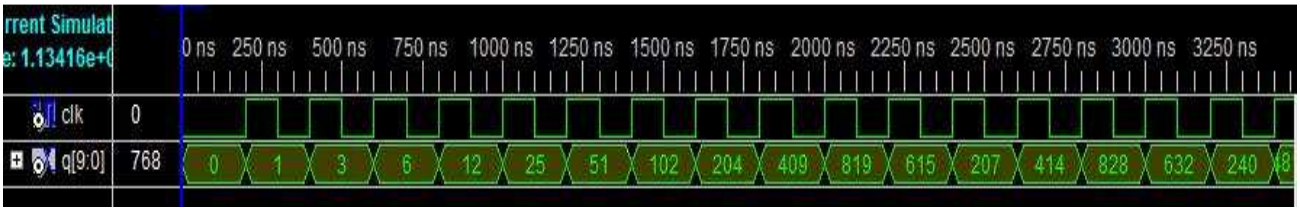


Figure IV.4 résultat de simulation du générateur d'adresse pseudo-aléatoire à 10 étages.

Et dans la figure IV.5 le résultat de simulation de tout le bloc, ayant comme entrée l'horloge CLK et comme sorties X1 et X2.

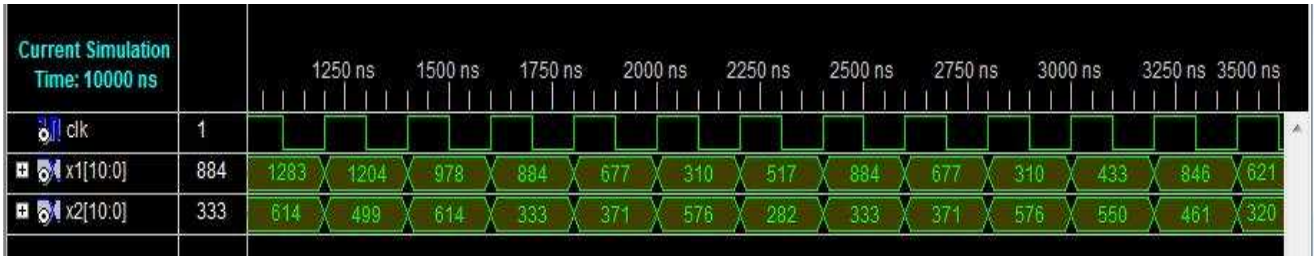


Figure IV.5 Résultat de simulation du bloc de mémorisation des paramètres chromosomiques.

2.1.2. Bloc de calcul de distance

Au cours de notre étude théorique de l'algorithme de Kohonen, nous avons trouvé que la plupart des ouvrages qui le traitent, utilisent la distance euclidienne. Contrairement à ce qui a été vu en théorie, et exploitant nos connaissances antérieures, nous avons opté pour l'utilisation de la distance de Manhattan, qui elle, se calcule en un temps réduit et donc occupe moins d'espace sur le circuit FPGA sur lequel nous allons implémenter notre application.

Un petit script Matlab (Annexe A) comparant la distance euclidienne avec celle de Manhattan permet de constater que les deux formules donnent exactement le même résultat en termes d'indice du neurone gagnant.

La distance de Manhattan se calcule comme suit :

Dans un espace vectoriel normé $(E, \|.\|)$, on peut toujours définir de manière canonique une distance *d* à partir de la norme. Il suffit de poser :

$$\forall(x,y) \in E \times E, d(x,y) = \|y - x\|$$

En particulier, dans $\mathbb{R}^n$, soit deux points de E, $(x_1, x_2, \dots, x_n)$ et $(y_1, y_2, \dots, y_n)$, on exprime les différentes distances ainsi :

La distance euclidienne : $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

La distance de Manhattan : $\sum_{i=1}^{n}|x_i - y_i|$

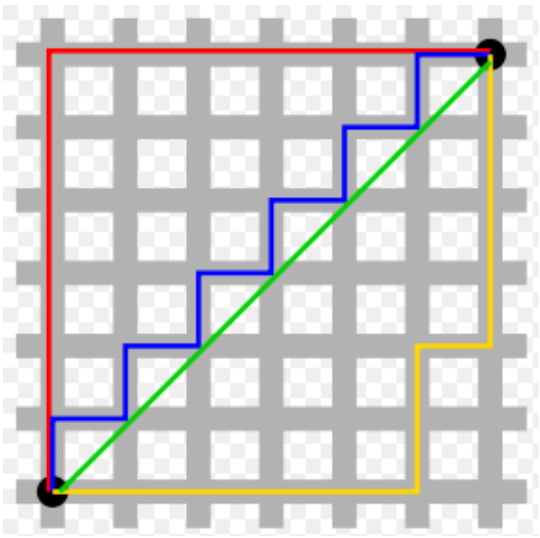


Figure IV.6 Distance de Manhattan (chemin rouge, jaune et bleu) et distance euclidienne en vert

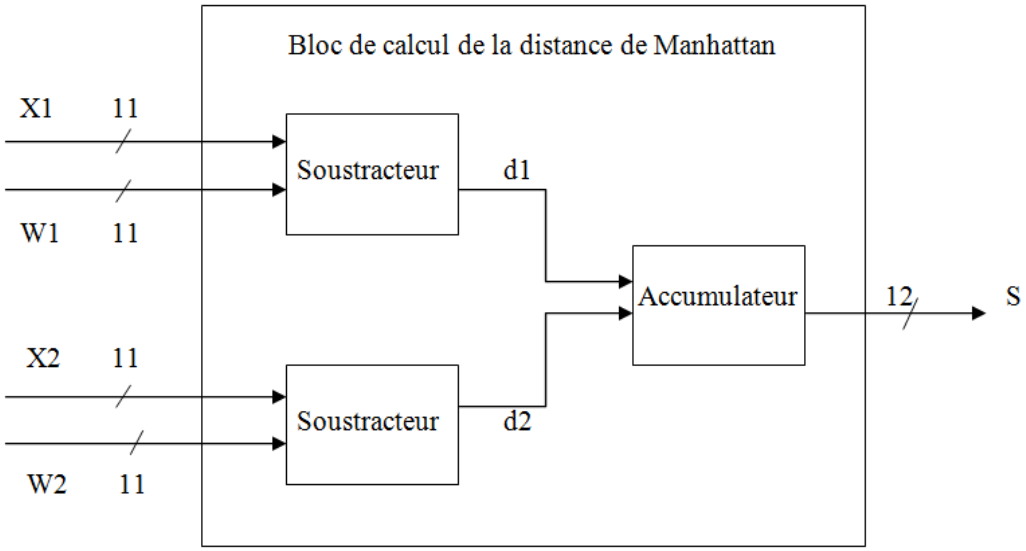


Figure IV.7 Architecture du bloc de calcul de distance de Manhattan.

Les soustracteurs calculent la valeur absolue de la différence entre les composantes des vecteurs d'entrées X_i et W_i.

X_1 et X_2 étant les données concernant la longueur, et l'indice centromérique extraits du bloc de mémorisation à l'itération n.

W_1 et W_2 étant les poids synaptiques mis à jour à l'itération (n-1) et délivrés par le bloc de mise a jour des poids synaptiques dont l'architecture sera décrite par la suite.

On verra par la suite que X_1 *et* X_2 sont communs pour toutes les unités (ou neurones), alors que W_1 *et* W_2 sont spécifiques pour chaque unité de la carte auto-organisatrice.

Ci-dessous, la figure du résultat de simulation de ce bloc.

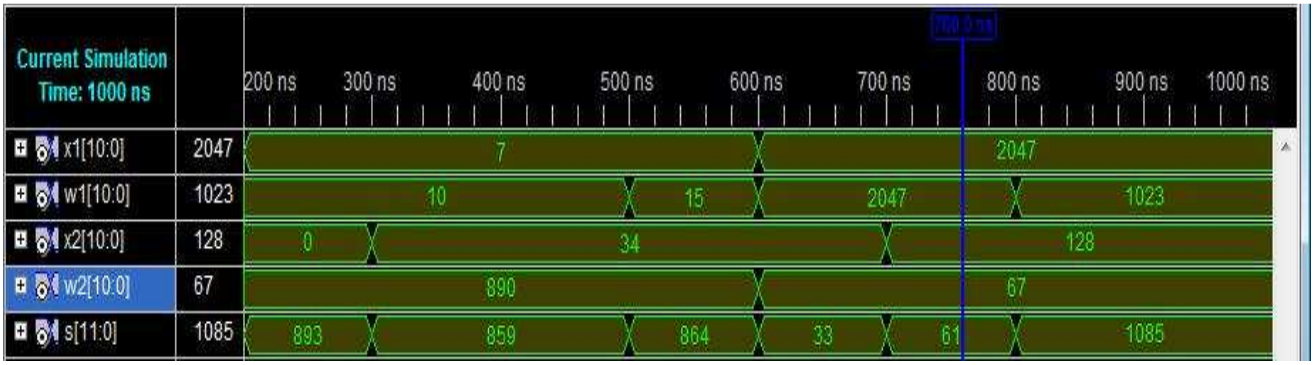


Figure IV.8 Résultat de simulation du bloc de calcul de la distance de Manhattan.

Ce bloc sera exploité 24 fois pour le calcul des 24 distances. Pour un système de classification dynamique et plus robuste, nous avons ajouté une entrée supplémentaire au comparateur (figure IV.9) qui permet de déterminer le neurone gagnant, pour que celle-ci soit exploitée si l'on veut par exemple utiliser ce système pour une classification à 25 classes (par exemple cas où le classificateur détecte une aberration chromosomique structurale telle que le 9 inversé).

Le comparateur, appelé « find_min_25 » retourne l'indice du neurone gagnant.

On note :

W_{j1} : la première composante du vecteur de mise à jour du poids synaptique du jème neurone

W_{j2} : la deuxième composante du vecteur de mise à jour du poids synaptique du jème neurone

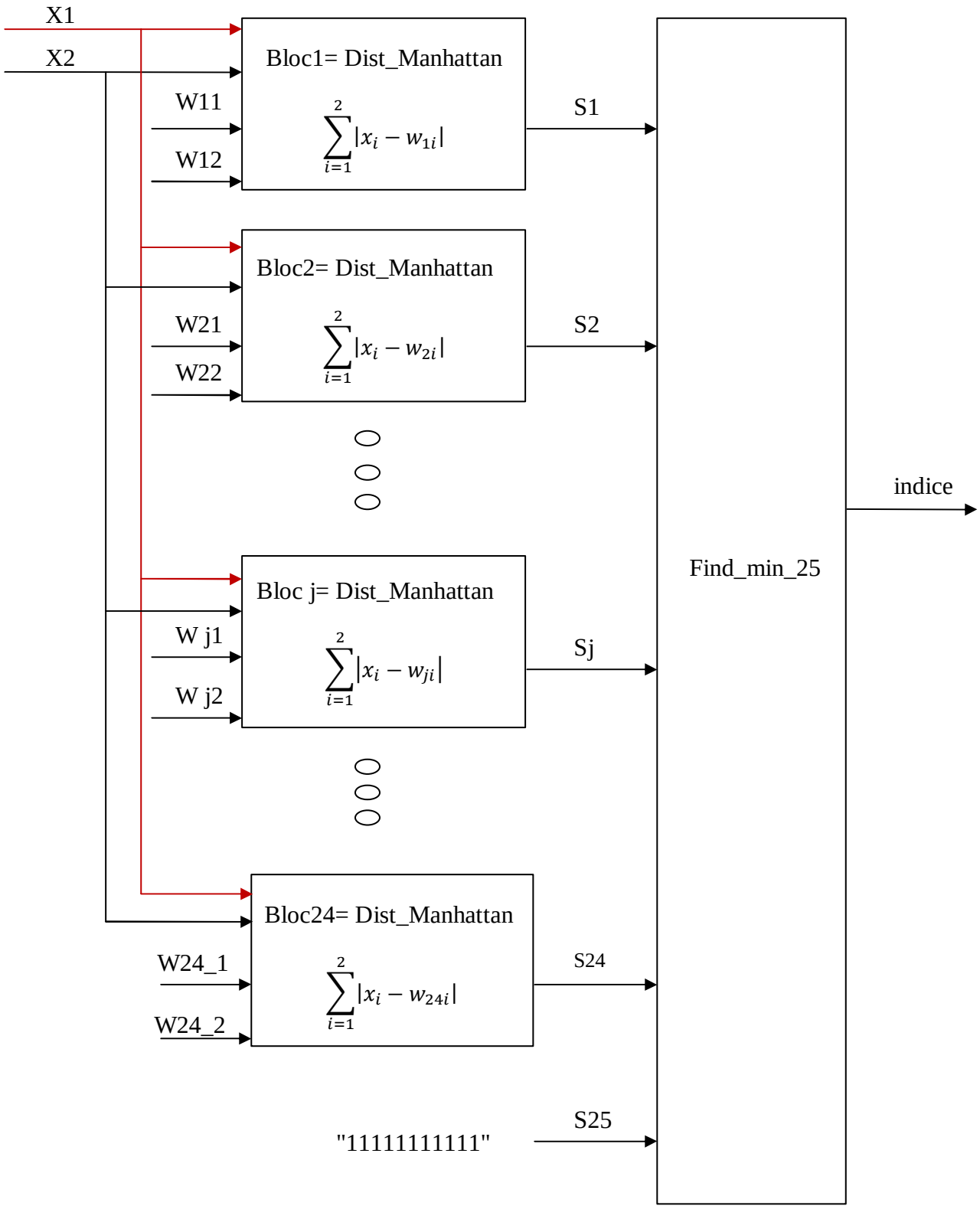


Figure IV.9 Architecture du bloc de détermination du neurone gagnant.

La figure IV.10 montre le résultat de simulation de ce bloc.

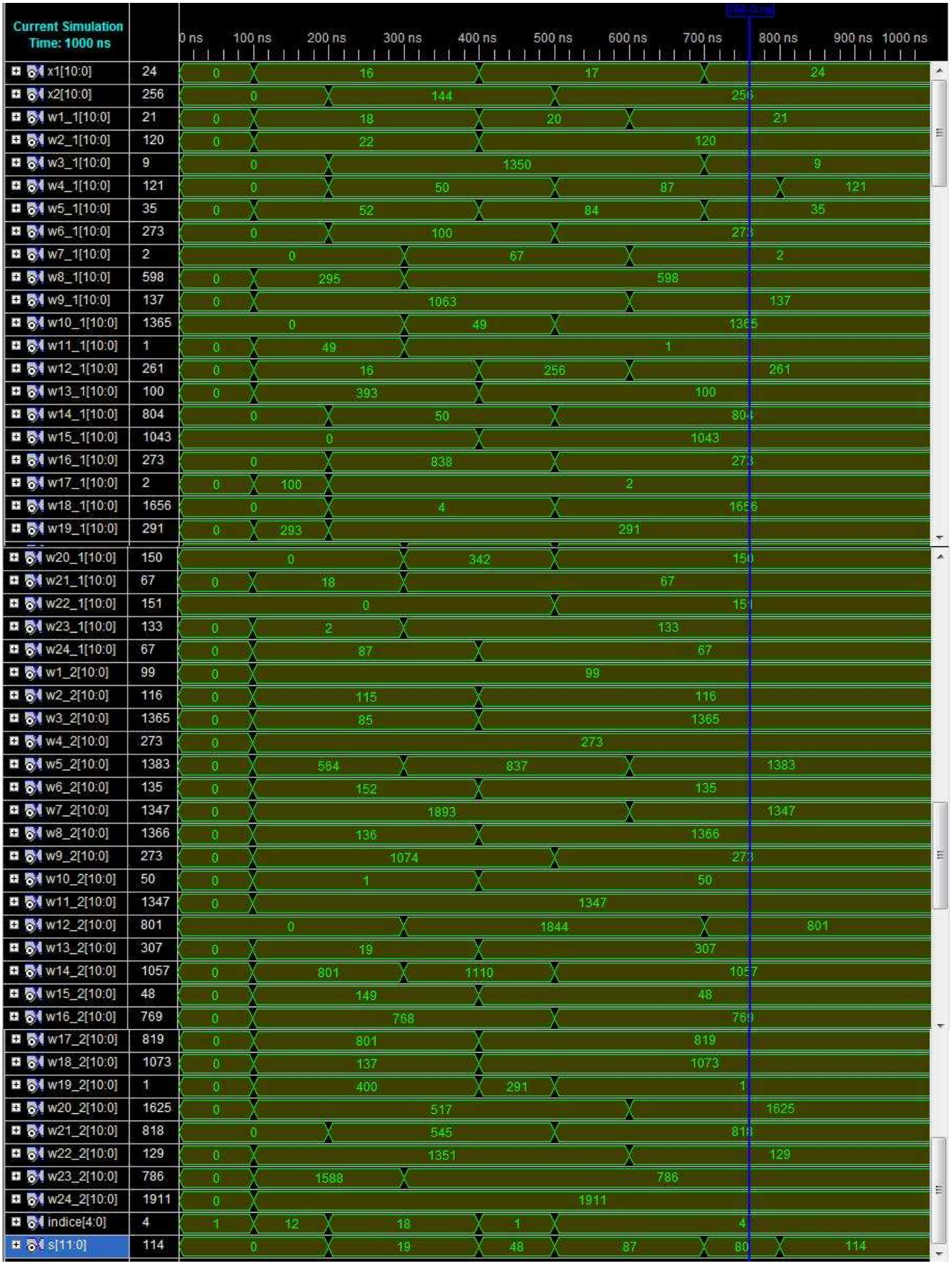


Figure IV.10 Résultat de simulation du bloc de détermination du neurone gagnant.

2.1.3. Bloc de mise à jour des poids synaptiques :

C'est le bloc qui met en œuvre l'équation (2.10) de mise à jour de l'algorithme de Kohonen. La figure IV.11 décrit globalement son architecture interne.

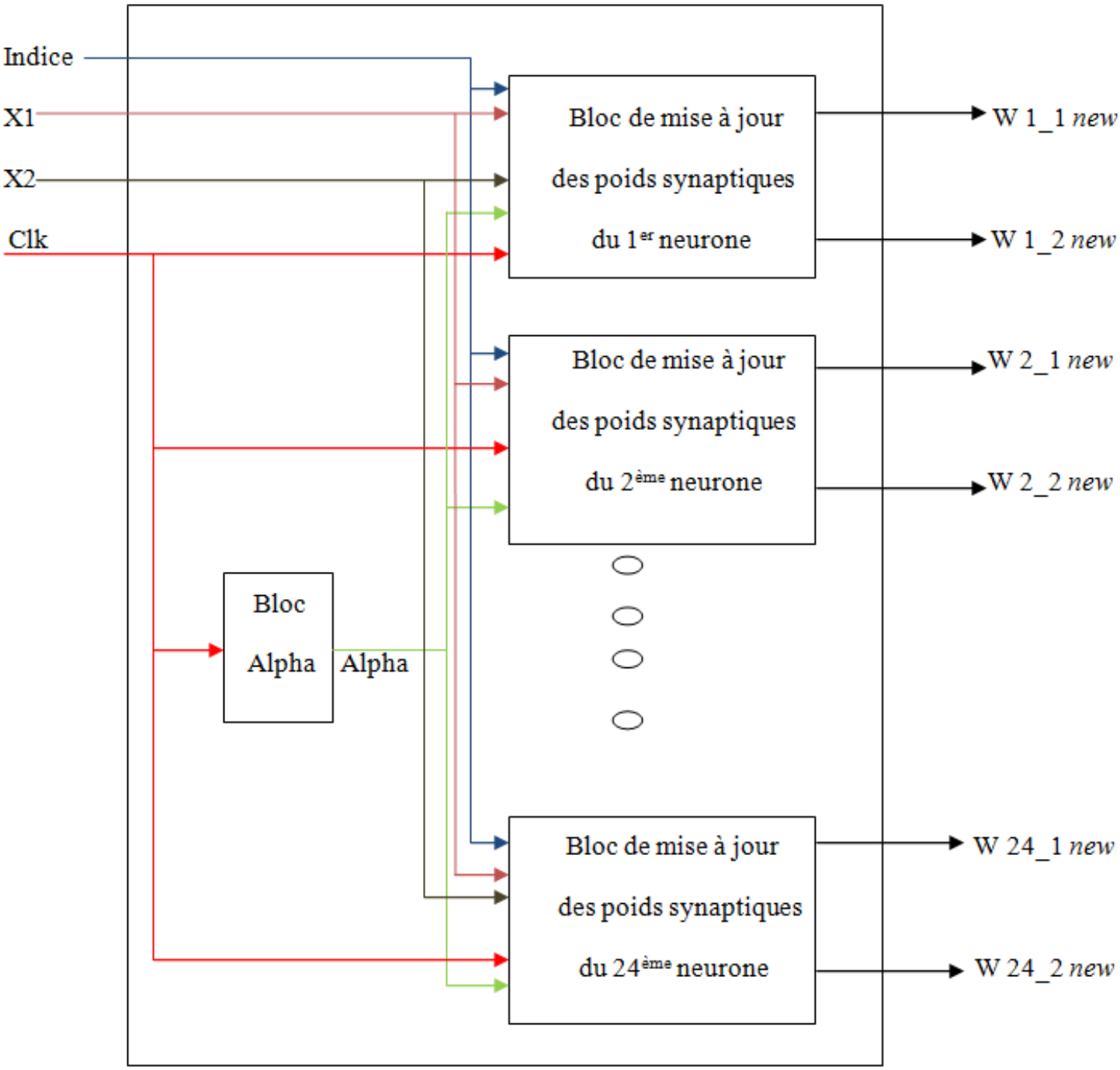


Figure IV.11 Architecture du bloc de mise à jour des poids synaptiques.

Une fois le neurone gagnant déterminé, les poids W_{ji} doivent être mis à jour. Le neurone gagnant et ses voisins seront gratifiés selon les équations (2.10) et (2.11) décrites au chapitre II.

Le bloc Alpha : est celui d'une fonction décroissante donnant en sortie la variable Alpha qui correspond au coefficient d'apprentissage. Nous avons proposé un décompteur qui se décrémente à chaque itération. Lorsqu'il atteint la valeur zéro (0) il génère un signal d'arrêt d'apprentissage.

Le bloc de mise à jour du j-ième neurone est décrit par la figure IV.12.

Contrairement à ce que l'on peut croire, les 24 blocs de mise à jours des neurones sont différents les uns des autres. Ceci est dû d'une part à la fonction de voisinage qui diffère d'une classe à une autre (et donc d'un neurone à un autre) et d'autre part aux indices locaux attribués à chaque neurone.

L'unité de calcul est constituée essentiellement, comme le montre la figure IV.13, d'additionneurs, de soustracteurs, de multiplieurs et de diviseurs.

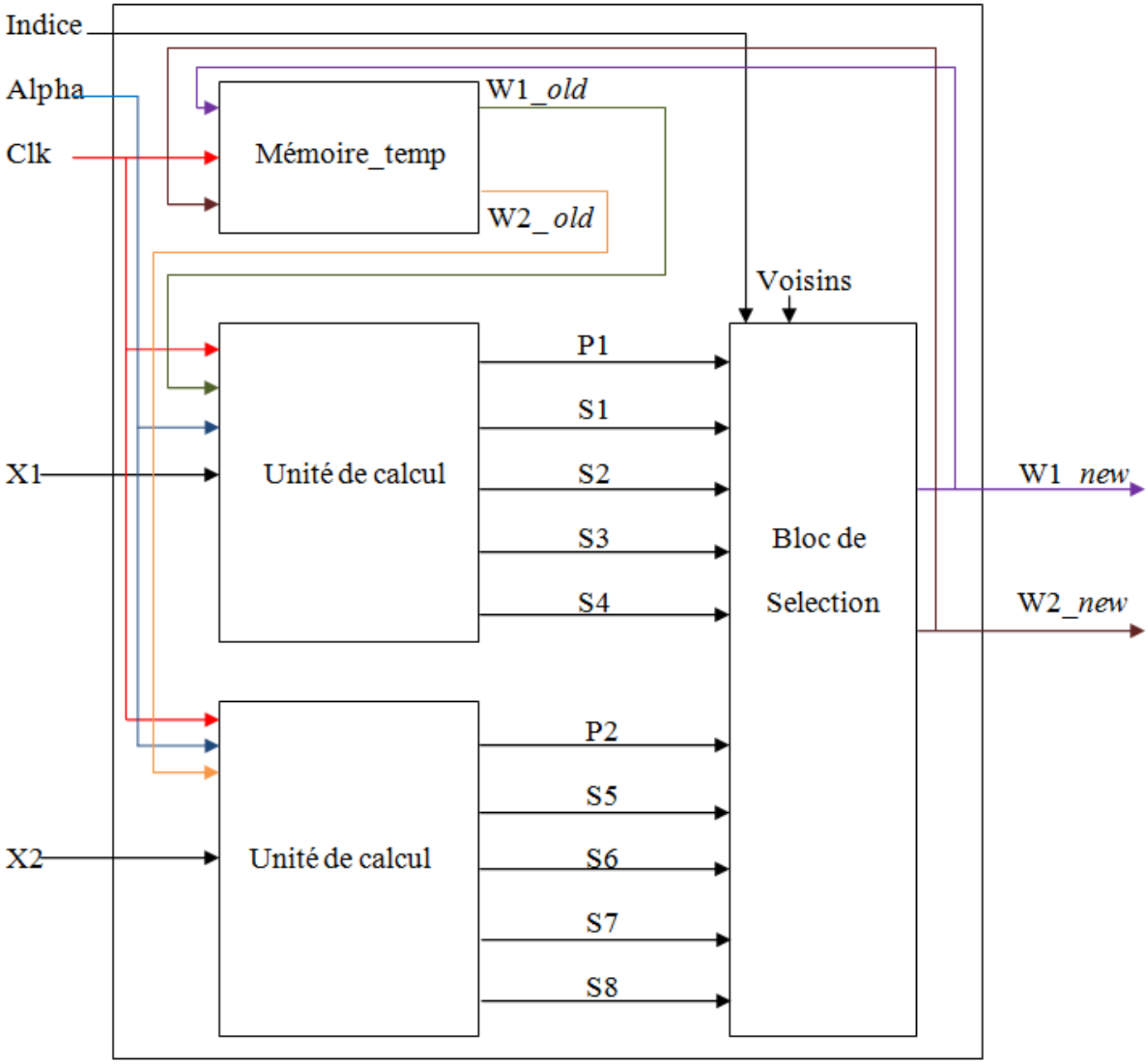


Figure IV.12 Description du bloc de mise à jour du j-ième neurone.

Les diviseurs utilisés réalisent la division par 128 (2^8). Il est relativement simple à réaliser. En effet, diviser un nombre par une puissance de 2 (c à d 2^n) revient à faire le complément à 2 de ce nombre puis un décalage vers la droite de *n* bits. Ceci rend la représentation des données dans la mémoire possible, voire même la rend facile. Prenons l'exemple suivant :

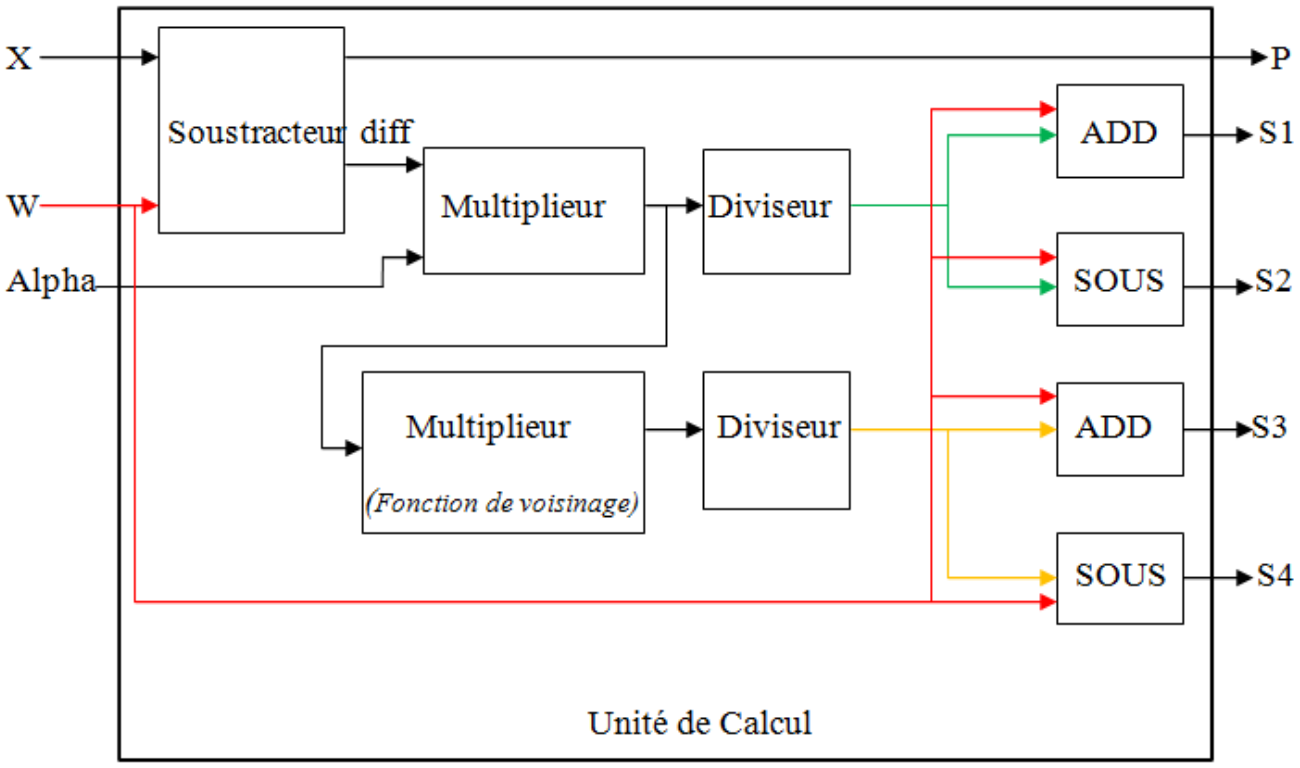


Figure IV.13 Description schématique d'une unité de calcul

Soit un chromosome caractérisé par un vecteur $X\begin{pmatrix}10{,}02\\4{,}8\end{pmatrix}$, la première composante étant la longueur de ce chromosome, la deuxième composante de ce vecteur étant l'indice centromérique multiplié par dix (pour gagner en précision).

La représentation de ce vecteur en mémoire s'obtient comme suit :

$10{,}02 \times 128 = 1282.56 \cong 1283$, représenté en mémoire après conversion par : "10100000011".

$4{,}80 \times 128 = 614.4 \cong 614$, représenté en mémoire après conversion par : "01001100110".

Supposons que les poids synaptiques du neurone gagnant sont donnés par le vecteur $W\begin{pmatrix}9.8\\4.5\end{pmatrix}$

Ces poids sont représentés en mémoire par :

$9.8 \times 128 = 1254.4 \cong 1254$, après conversion : "10011100110"

$4.5 \times 128 = 576 \cong 576,$ après conversion:"01001000000"

Selon l'équation (2.9) de l'algorithme de Kohonen, la mise à jour de ces poids se fait comme suit :

$$W_{ji}(n+1) = W_{ji}(n) + \alpha(n) \times \left[x_i(n) - w_{ji}(n) \right]$$

Comme vu précédemment, le bloc Alpha se décrémente de 127 à 0 , et donc en base décimale de 0.99 à 0. Dans cet exemple on prendra $\alpha = 100$ (en base 128), c.à.d. 0.781 en base décimale.

Calcul dans la base décimale :

La mise à jour de la première composante du vecteur *W* :

$W_1 = 9.8 + 0.781 \times (10.02 - 9.8) = 9.97$

La mise à jour de la deuxième composante du vecteur *W* :

$W_2 = 4.5 + 0.781 \times (4.8 - 4.5) = 4.73$

Calcul dans la base 128 :

La mise à jour de la première composante du vecteur W :

$$W_1 = 1254 + \frac{[100 \times (1283 - 1254)]}{128} = 1277$$

La mise à jour de la deuxième composante du vecteur W :

$$W_2 = 576 + \frac{100 \times (614 - 576)}{128} = 606$$

Vérification :

$$W1_{\text{dans la base décimale}} = \frac{W1_{\text{dans la base 128}}}{128} = \frac{1277}{128} = 9.97$$

$$W2_{\text{dans la base décimale}} = \frac{W2_{\text{dans la base 128}}}{128} = \frac{606}{128} = 4.73$$

Le bloc de sélection décide quelles valeurs doivent prendre les poids synaptiques à l'itération *i*. La sélection se fait selon l'organigramme suivant :

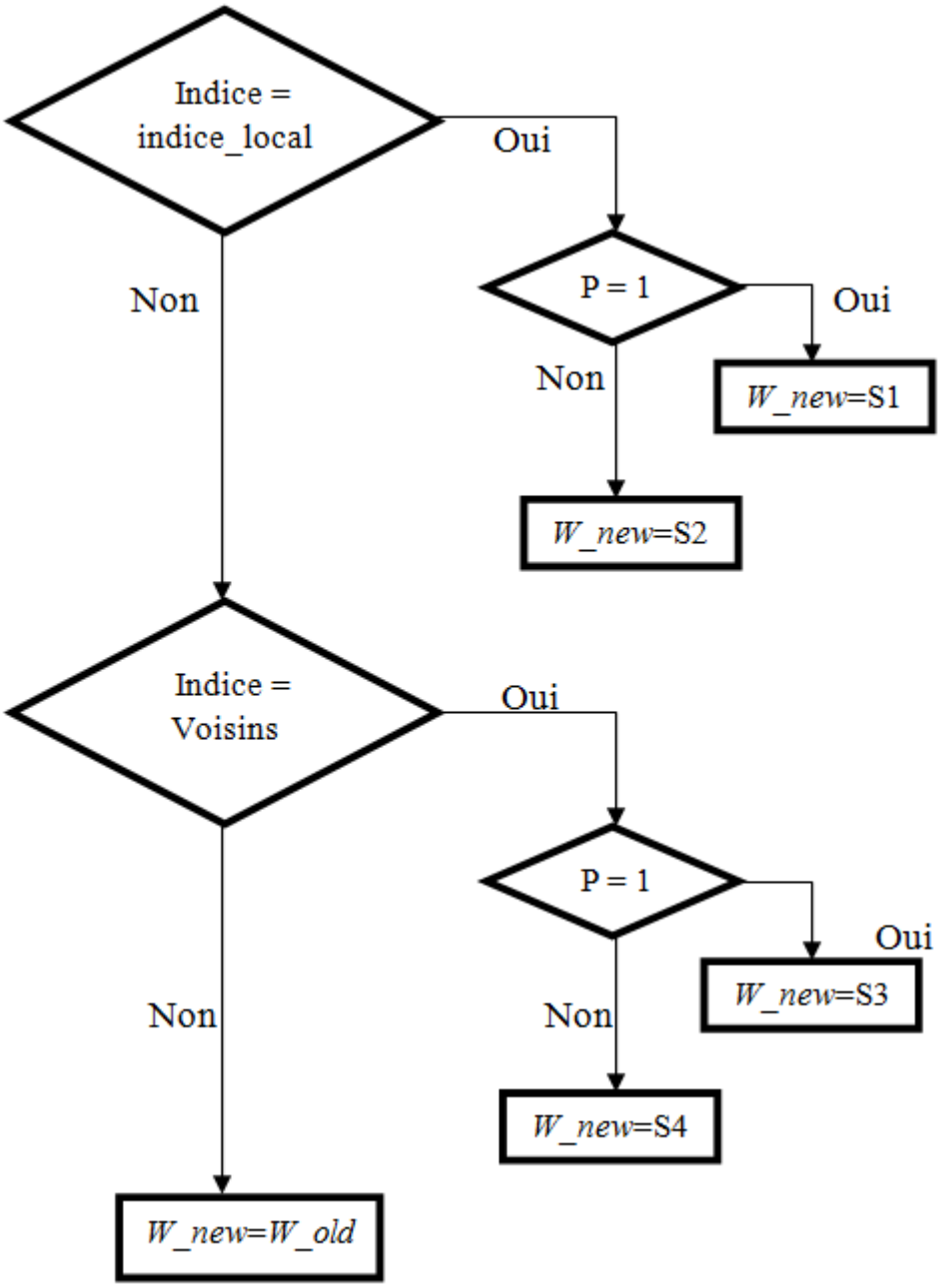


Figure IV.14 Organigramme illustrant le fonctionnement du bloc de sélection.

Le résultat de simulation du bloc de mise à jour des poids synaptique est montré à la figure IV.15.

Ce bloc est utilisé 24 fois, donnant ainsi les nouveaux poids synaptiques correspondant à chaque neurone.

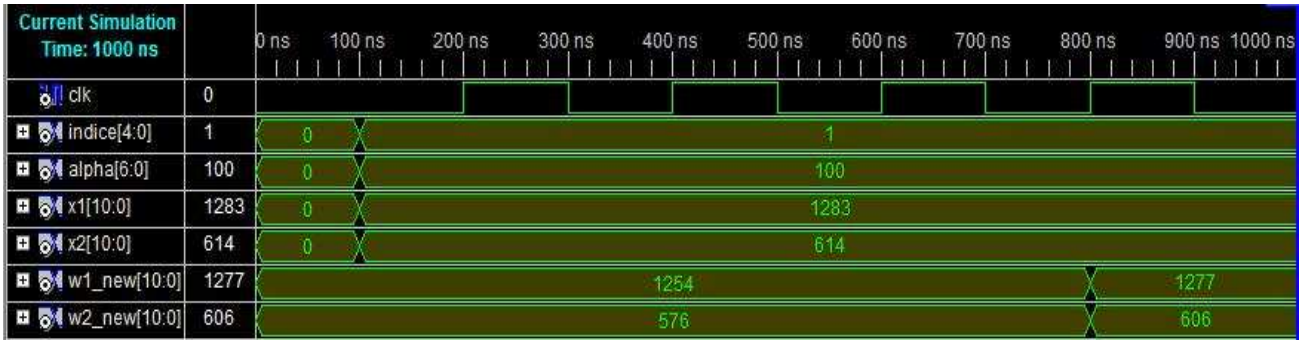


Figure IV.15 Résultat de simulation de bloc de mise à jour.

2.2. Phase de classification :

Dans cette phase, on présente à l'entrée du système les valeurs qui correspondent aux longueurs et indices centromériques des chromosomes, extraites à partir d'une image en phase de mitose. Ces valeurs sont stockées dans une mémoire appelée Ram_test.

La figure IV.16 explique les différentes étapes à suivre lors de la classification.

Comme on peut le constater, dans cette partie on exploite les différents blocs réalisés dans la phase d'apprentissage.

Dans la figure IV.17 est illustré un schéma bloc du classificateur.

Extraction des paramètres

Une Rom contenant les paramètres des chromosomes dont on veut réaliser le caryotype

Les poids synaptiques finaux déjà calculés dans la phase d'apprentissage

Calcul de la distance de Manhatten

24 valeurs

Identification du neurone gagnant

Indices

Fin de la classification

Enregistrement des résultats de la classification

Ce fichier .txt contient

Le résultat de la classification

Figure IV.16 Différentes étapes de la phase de classification.

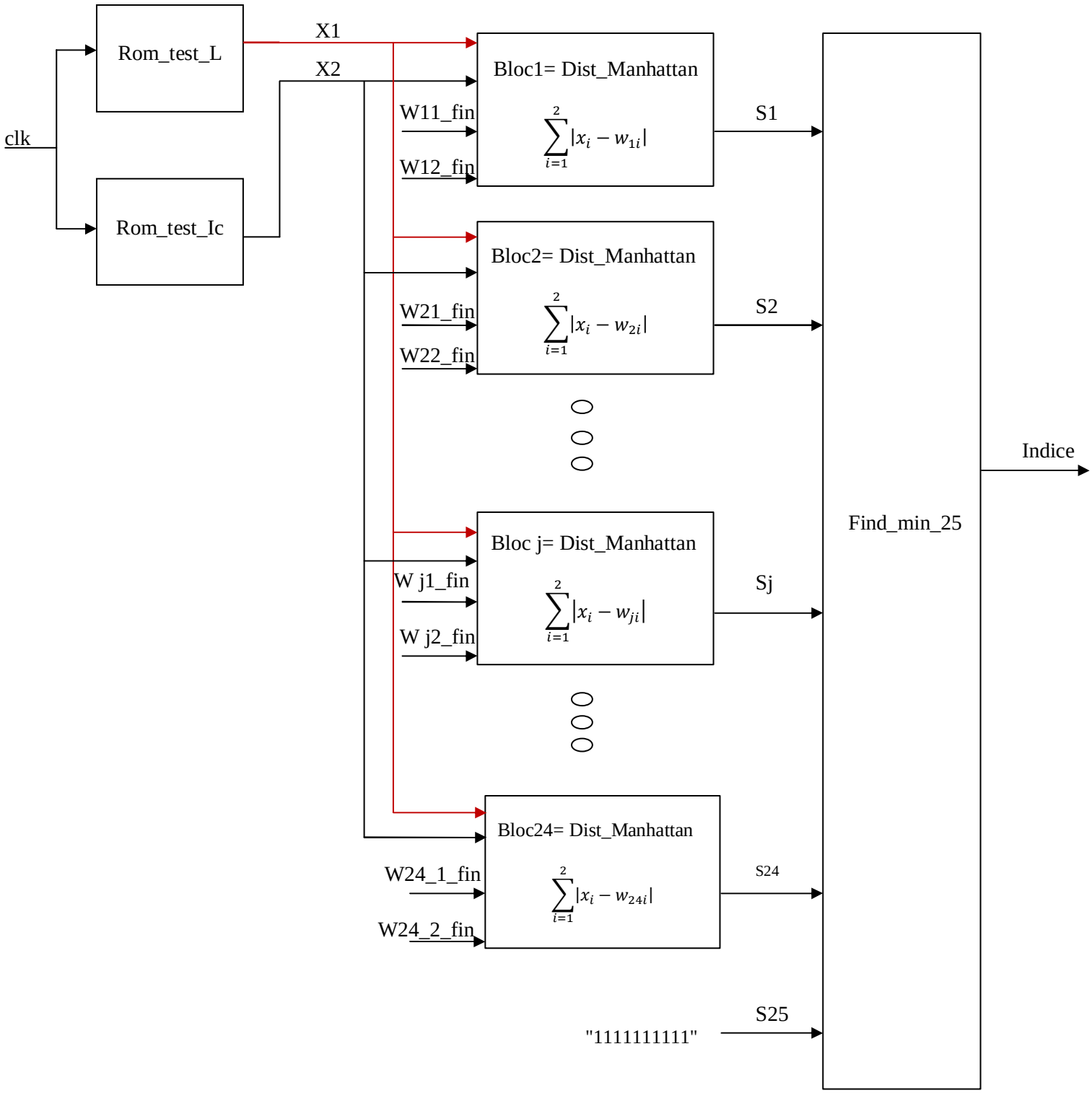


Figure IV.17 Architecture du bloc de classification.

Les Wji_fin correspondent aux poids synaptiques finaux calculés dans la phase d'apprentissage. Et la variable de sortie « indice » correspond à la classe du chromosome dont les caractéristiques (L et Ic) sont présentées à l'entré du système (X1 et X2) qui proviennent de Rom_test_L et Rom_test_Ic respectivement.

Dans un premier temps, les indices qui correspondent à la classe de chaque paire (L , Ic) seront affichés dans un fichier texte dans le même ordre que dans les ram_test. Toutefois, avec les commandes d'affichage dans un ficher texte on ne peut pas implémenter la description c'est-à-dire qu'il n'est pas synthétisable, nous avons donc exploité l'afficheur LCD de la Spartan 3E.

Notons que notre système permet de détecter les aberrations numériques telles que la trisomie 21.

3. Le classificateur

Vue par un esprit scientifique, l'automatisation de la classification des chromosomes ne devrait pas s'éloigner de la logique classique établie et exploitée par les cytogénéticiens. En d'autres termes et d'une manière plus précise, le système de neurones classificateur devrait « penser » de la même manière – ou presque – qu'un cytogénéticien.

Graduellement, nous avons développé ce raisonnement, dont le déroulement aura conduit à opter pour un système multi-réseau, composé de deux modèles de Kohonen :

- Le premier (déjà réalisé) classifie les chromosomes par leurs morphologies (longueur et indice centromérique).
- Le second reconnaît les échantillons selon leurs motifs (densités de profil).

Pour ce faire, nous avons construit le deuxième réseau de neurones tout en suivant les mêmes démarches que précédemment, la seule différence réside dans le nombre d'entrée (128 vecteurs codés sur 11 bits chacun) contrairement au premier réseau qui lui, avait deux entrées (la longeur des chromosomes L et l' indices centromériques Ic, codé sur 11 bits). Ainsi, les blocs construits précédemment seront exploités.

Une combinaison banale (addition ordinaire ou avec coefficient) des vecteurs de sortie respectifs de ces deux réseaux constituera l'ultime résultat de notre classification.

La Figure IV.18 représente une illustration du système de neurones adopté. Le même système est représenté en schéma bloc dans la figure IV.19.

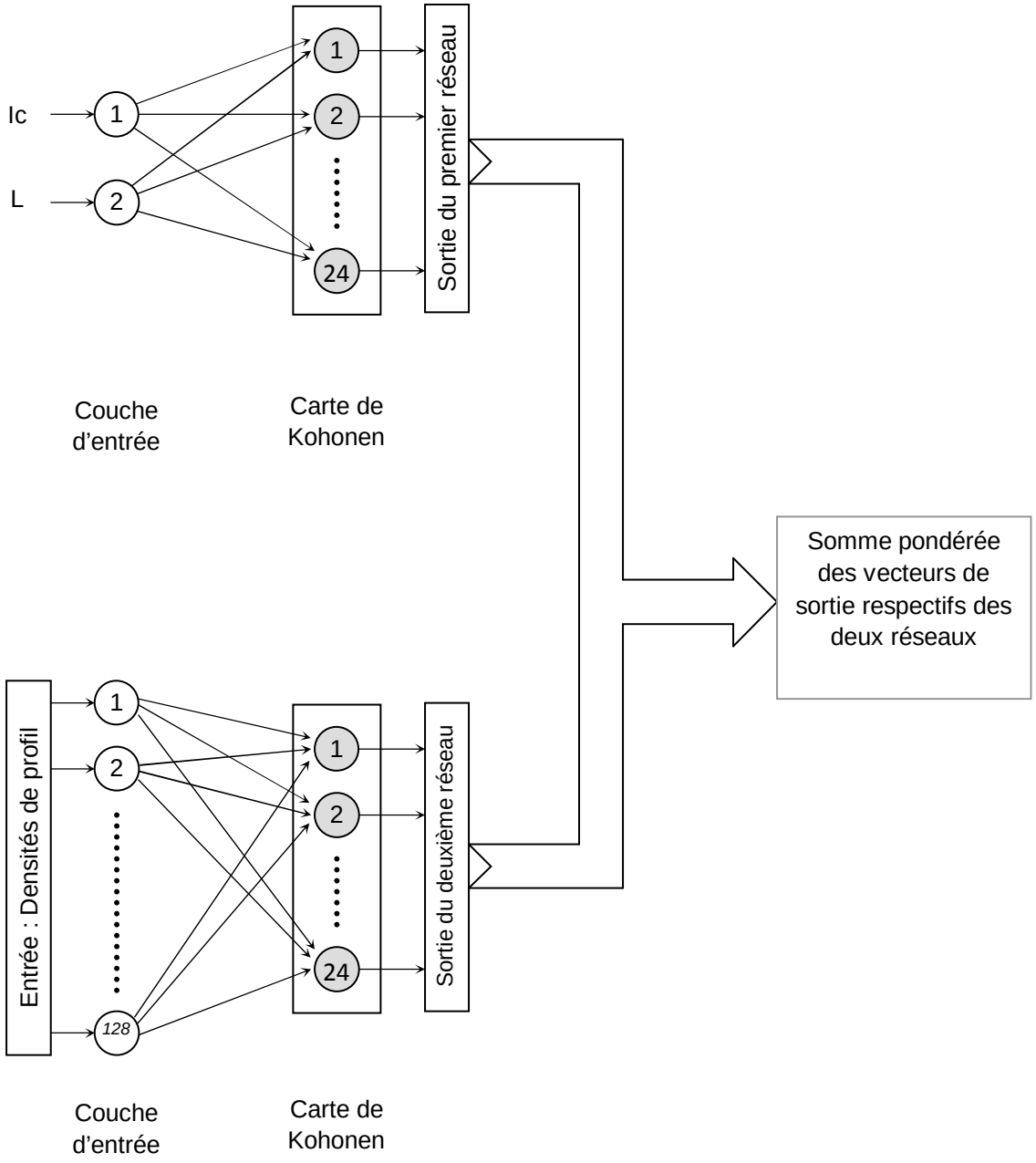


Figure IV.18 Illustration du système de neurones adopté.

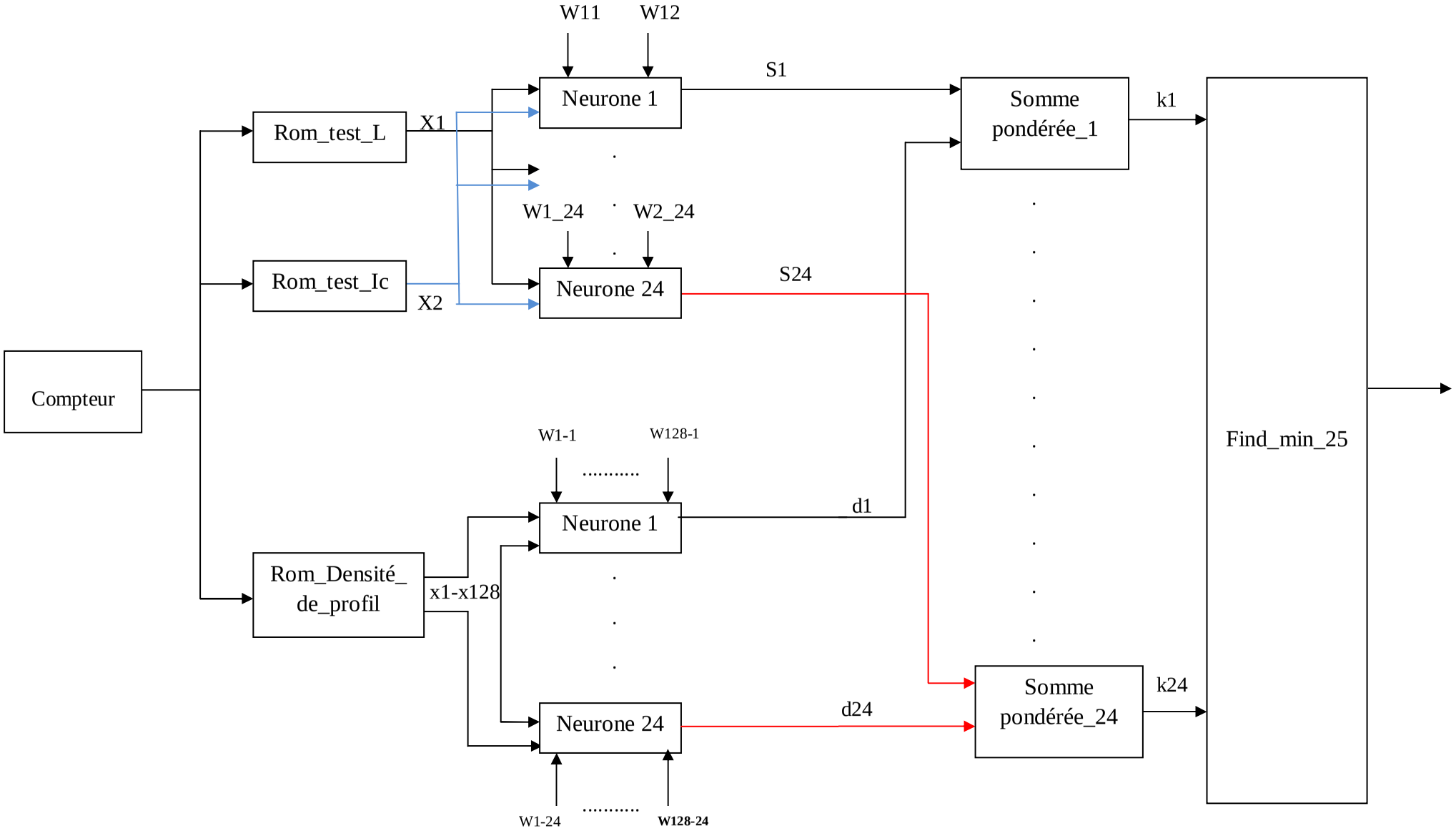


Figure IV.9 Schéma bloc du système de neurones adopté

4. Conclusion

A travers ce chapitre, nous avons exposé les différents modules qui composent notre classificateur. Nous constatons que chaque bloc, le plus petit qu'il soit, a un rôle primordial dans l'architecture que nous avons proposé.

L'idée de l'architecture multi réseau permet de pondérer la sortie finale en favorisant l'une ou l'autre des propriétés selon la qualité de l'image et par conséquent d'orienter la classification selon les paramètres les mieux perçus.

Dans le chapitre qui suit, les différentes étapes de la conception et de la réalisation du système sont exposées.

CHAPITRE V :

CONCEPTION ET RÉALISATION

1. Introduction

Dans ce chapitre, nous allons présenter les différents outils utilisés pour la conception et la réalisation de notre projet, ainsi que les différentes étapes suivies.

2. L'outil ISE de Xilinx :

ISE Foundation 10.1 est un environnement intégré de développement de systèmes numériques ayant pour but une implémentation matérielle sur FPGA de la compagnie Xilinx. Les designs peuvent être décrits sous trois formes principales :

- Schémas.
- langage de description matérielle (HDL) comme VHDL et Verilog.
- diagrammes d'états.

Figure V.1 L'outil ISE 10.1 de Xilinx

ISE intègre donc différents outils permettant de passer à travers tout le flot de conception d'un système numérique :

- un éditeur de textes, de schémas et de diagrammes d'état.
- un compilateur VHDL/Verilog.
- un outil de simulation.
- des outils pour la gestion des contraintes temporelles.
- des outils pour la synthèse.
- des outils pour la vérification.
- des outils pour l'implémentation sur FPGA.

3. Langage VHDL :

La complexité croissante des systèmes a imposé rapidement de recourir à des niveaux d'abstraction de plus en plus importants pour pouvoir appréhender les systèmes à concevoir dans leur globalité et également pour pouvoir en assurer le test et la maintenance de manière efficace. Ce sont des besoins qui ont conduit à la définition du langage VHDL qui a pour objectif de couvrir toutes les étapes de cycle de vie des systèmes électroniques, de la spécification à la réalisation.

VHDL pour VHSIC Hardware Description Langage. (VHSIC : Very High Speed Integrated Circuit) n'est pas un langage de programmation mais un langage de description matériele. Il a été défini et introduit pour apporter de la méthode et de la rigueur dans le cycle de développement des systèmes matériels, il permet de décrire les fonctionnalités d'un système matériel, les relations qu'elles ont entre elles et avec l'extérieur.

A ce jour, on utilise le langage VHDL pour :

- Concevoir des ASIC.
- Programmer des composants programmables du type PLD, CPLD et FPGA.
- Concevoir des modèles de simulations numériques ou des bancs de tests.

4. Les circuits FPGA :

4.1.Architecture :

Les FPGA (Field Programmable Gate Arrays ou "réseaux logiques programmables") sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser dans des algorithmes différents en un temps très court.

Les circuits FPGA sont essentiellement constitués d'une matrice de blocs logiques programmables entourés de blocs d'entrée/sortie programmables également. L'ensemble est relié par un réseau d'interconnexions programmable.

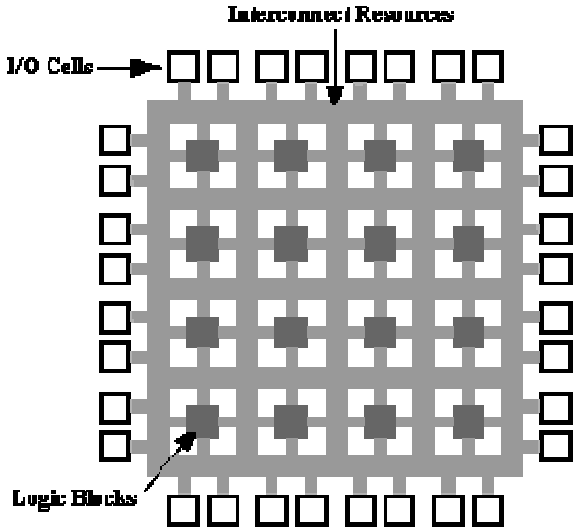


Figure V.2 Structure interne d'un FPGA

4.1.1. Les CLB :

CLB pour Configurable Logic Bloc, Les blocs logiques configurables sont des éléments déterminants des performances du FPGA. La structure du bloc logique change selon le fabricant, la famille,…etc. pour la spartan 3-E de Xilinx, chaque CLB est constitué de 4 Slices. De même, Chaque slice est constitué principalement d'une look-up Table (RAM) pour implémenter une fonction combinatoire, en plus d'une bascule D.

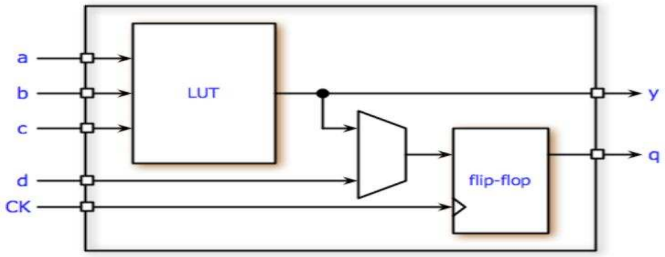


Figure V.3 Structure d'un slice a 4 entrées

La fonction de la LUT est de stocker la table de vérité de la fonction combinatoire à implémenter dans la cellule. Comme l'indique la figure suivante :

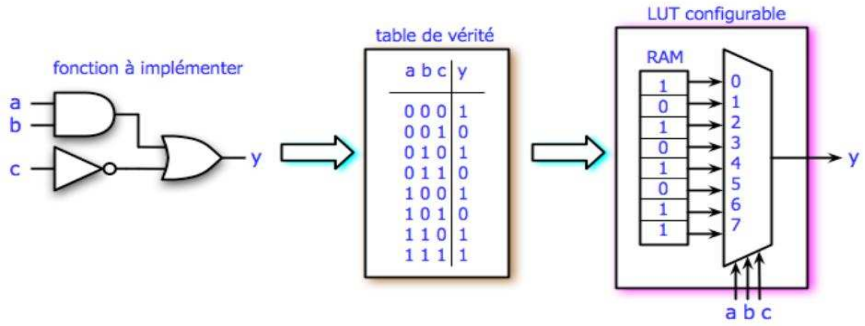


Figure V.4 Structure comportant une LUT de 3 bits

4.1.2. Les IOB :

IOB pour Input Output Bloc, ces blocs d'entrée/sortie permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Ces blocs IOB contrôlent les broches du composant et ils peuvent être définis en entrée, en sortie, en signaux bidirectionnels ou être inutilisés (haute impédance).

4.1.3. Les interconnexions :

Il ne suffit pas de disposer de blocs logiques ou d'entrées/sorties performants, encore il faut pouvoir les interconnecter avec un maximum d'efficacité afin que leur taux d'utilisation dans un circuit donné soit aussi élevé que possible. On distingue 3 types d'interconnexions :

- les interconnexions à usage général.
- les interconnexions à grande vitesse.
- Les longues lignes : permettent de transmettre avec un minimum de retard, les signaux entre les éléments éloignés du réseau.

Interconnecter les différents blocs du circuit FPGA (IOB et CLB), consiste à appliquer un potentiel bien défini sur la grille de certains transistors CMOS, régulièrement répartis le long du composant FPGA. Afin de réaliser la fonctionnalité souhaitéé, ces potentiels sont mémorisés dans une mémoire SRAM (Static RAM).

Il existe actuellement plusieurs fabricants de circuits FPGA dont Xilinx , Altera et Alcatel sont les plus connus.

4.2. La plate-forme de développement SPARTAN 3E

La plate forme que nous avons utilisée pour l'implémentation de notre application, disponible au CDTA (Centre de Développement des Technologies Avancées) à Baba Hassen où nous avons effectué notre stage, dispose des composants suivants :

- Un FPGA Spartan-3E (XC3S500E).
- Une PROM(XCF04S).
- Un CPLD (XS2C64A).
- Un oscillateur quartz délivrant un signal d'horloge a 50Mhz.
- Un Port VGA
- Une interface série RS232.
- Une interface JTAG
- Port Sourie/clavier SP/2
- Une LED ON : carte sous tension
- Une LED DONE : FPGA programmé.
- Un bouton poussoir PROG : initialise le FPGA
- Ecran LCD, LEDs, boutons poussoirs
- Port Ethernet 10/100

La carte est alimentée par une tension continue 5V. Des régulateurs fournissent des tensions comprises entre 1.1V et 3.5V nécessaires au fonctionnement des différents composants du kit de développement SPARTAN 3-E. La température d'utilisation est comprise entre 0°C et 85°C.

Pour l'affichage des résultats de notre travail, nous avons exploité l'écran LCD que dispose le kit de développement Spartan 3E. Tout ce qui concerne ces types d'afficheurs et leur pilotage par FPGA, en particulier celui dont on dispose est détaillé dans un document attaché dans l'annexe B.

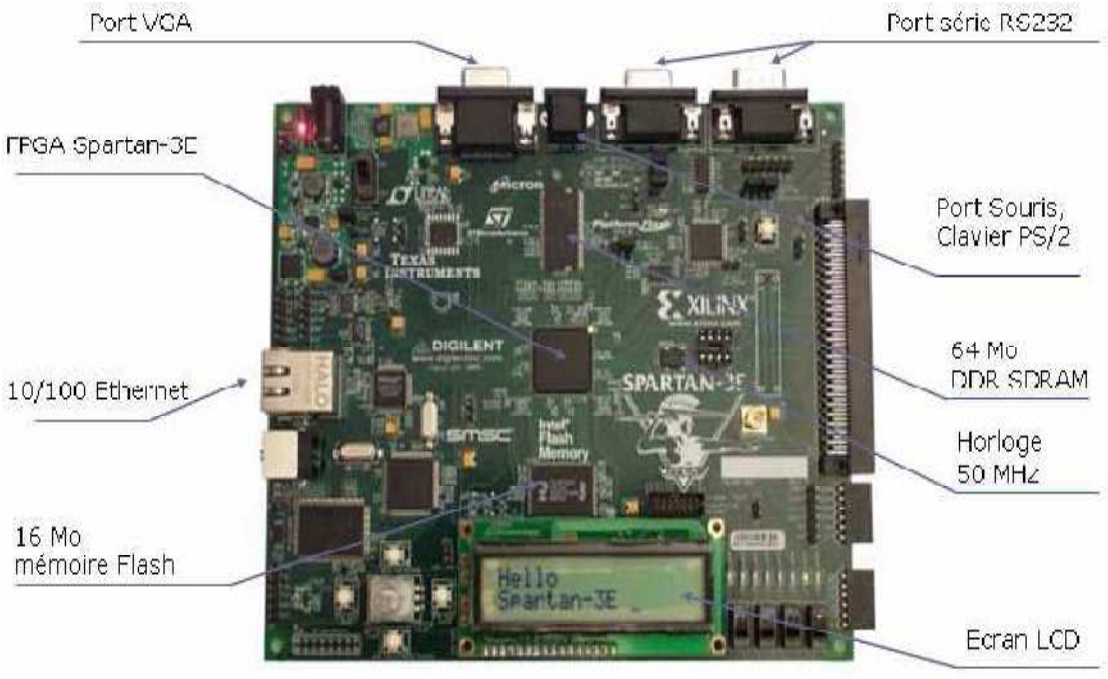


Figure V.5 Vue d'ensemble de la plate forme Spartan 3E

La nomenclature de la puce FPGA est : *XC3S500E-4fg320* tel que :

- *XC*: compagnie Xilinx
- *3SE* : Spartan 3 série E.
- *500* : c'est-à-dire le FPGA contient 500.000 portes élémentaires
- *-4* : la vitesse de fonctionnement du composant,
- *fg* : le type de boitier FPGA.
- *320* : le nombre de pins.

5. Les différentes étapes d'implémentation

Une fois le programme écrit, sa syntaxe vérifiée, l'implantation physique de ce programme sur un circuit FPGA doit passer par les étapes suivantes :

- La synthèse logique : La synthèse permet à partir d'une spécification VHDL, la génération d'une architecture au niveau transfert de registre RTL (register transfert level) qui permet l'ordonnancement et l'allocation de ressources sans une représentation physique, compilable par un outil de synthèse logique. Cette étape permet de générer la Netlist qui est une représentation en fichier texte du circuit qui souligne les connexions entre les différents éléments du circuit.

- La simulation fonctionnelle : cette étape permet de visualiser la forme des signaux (test bench waveform). Elle permet donc de vérifier uniquement la validité du circuit par rapport au cahier des charges d'un point de vue fonctionnel et non d'un point de vue temporel.

- L'optimisation et le placement / routage (Implement Design) : Avant l'utilisation du fichier netlist, celui-ci est optimisé. Dans cette étape les signaux inutilisés sont retirés, les expressions booléennes sont simplifiées, les signaux équivalents sont détectés, ceci se fait en trois sous étapes :

  - La première (Translate) est la transformation du circuit numérique en éléments logiques élémentaires.
  - La deuxième (Map) est la prise en compte des différents retards temporels dûs au câblage des différents blocs logiques programmables.
  - La troisième (Placement / routage) : cette étape consiste à attribuer les cellules (CLB) du circuit à chaque équation de la netlist et à définir les connexions. Des directives jointes à la netlist permettent une bonne répartition des cellules.

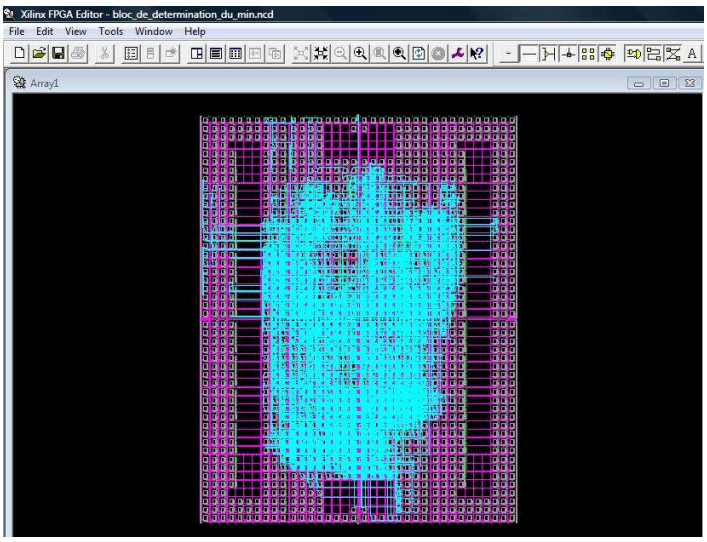


Figure V.6 Placement et routage de l'architecture sur circuit FPGA

- la génération de fichier de configuration : La dernière étape consiste à générer un fichier de configuration appelé Bitstream d'extension **'.bit'**. Ce fichier contient les informations fournies au composant FPGA Xilinx afin qu'il prenne en charge la configuration souhaitée.

Le logiciel de simulation offre une étape de spécification (User Constraints) permettant entre autre de spécifier les connexions des pattes de la puce FPGA avec les différents périphériques de la plate-forme (ecran LCD, LED, port VGA…), générant ainsi un fichier dit User File Constraint.

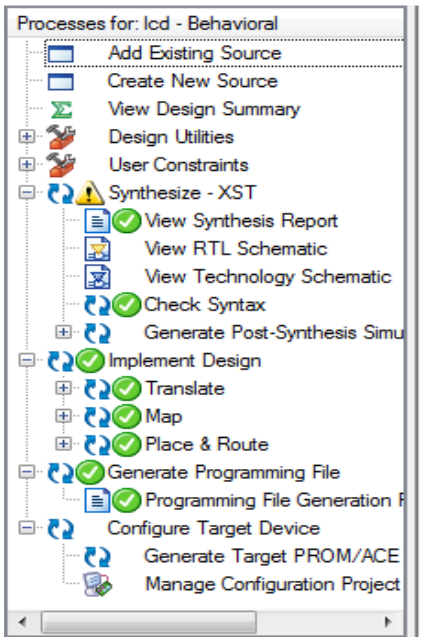


Figure V.7 Les différentes étapes d'implémentation sur circuit FPGA

Ainsi, à la mise sous tension du circuit, et en envoyant le bitstream, notre circuit est configuré et est prêt à l'utilisation.

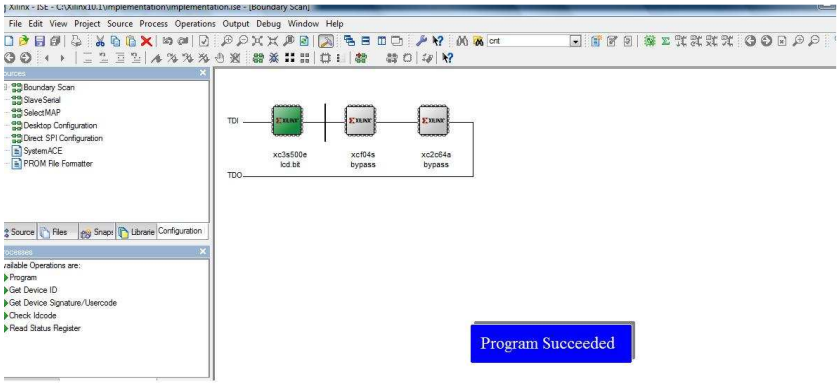


Figure V.8 L'envoi du *bitstream* et la configuration du FPGA

Des photos ont été prises avec un appareil photo numérique, illustrant l'application de notre projet.

Figure V.9 Photos illustrant le résultat final.

6. Conclusion

La réalisation de notre projet s'est introduite par la sélection d'un ensemble d'outils et de technologies. Nous avons pris soin à ce qu'ils soient les plus professionnels, les plus récents et les plus performants. Un bon résultat nécessite, néanmoins, une utilisation de ces outils, mariée à une conception longuement étudiée. Les différentes sections de notre plan se sont basées, pour la plupart, sur des concepts intuitifs, mais efficaces, réaffirmant que les meilleures idées sont souvent les plus simples.

CHAPITRE VI :

RÉSULTATS ET CONCLUSION

1. Introduction

Nous présenterons au cours de ce chapitre les résultats de notre projet, les différentes conclusions tirées et les perspectives.

2. Espace occupé dans la Spartan3E

Nous avons réussi à implémenter les deux réseaux de kohonen nécessaires pour notre application, sur la carte FPGA Spartan 3-E, bien que celle-là ne dispose que d'un nombre limité de ressources comparé aux nouvelles générations des cartes FPGA (la famille Virtex). Cela revient principalement à l'utilisation de la distance de Manhattan au lieu de la distance euclidienne. Ce qui nous a fait gagner beaucoup de ressources sur la carte, en plus d'une rapidité de calcul.

L'algorithme de Kohonen, bien que puissant, n'occupe pas beaucoup d'espace dans la carte, il utilise 62% seulement des ressources (figure VI.1), néanmoins, le projet au complet a occupé environs 86% de la carte. Ceci est dû à tout ce qui a été attaché à l'algorithme : afficheur, détection d'aberration, ...etc.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	1,831	9,312	19%
Number of 4 input LUTs	5,316	9,312	57%
Logic Distribution			
Number of occupied Slices	3,212	4,656	68%
Number of Slices containing only related logic	3,212	3,212	100%
Number of Slices containing unrelated logic	0	3,212	0%
Total Number of 4 input LUTs	5,860	9,312	62%
Number used as logic	5,316		
Number used as a route-thru	544		
Number of bonded IOBs	9	232	3%
Number of BUFGMUXs	1	24	4%

Figure VI.1 Espace occupé dans la carte par l'algorithme de Kohonen

Concernant le temps de convergence qui constitue un paramètre très important pour la détermination de la performance d'un réseau de neurones, nous avons constaté que les résultats obtenus ne varient plus après 409,6 µs, (figure VI.2).

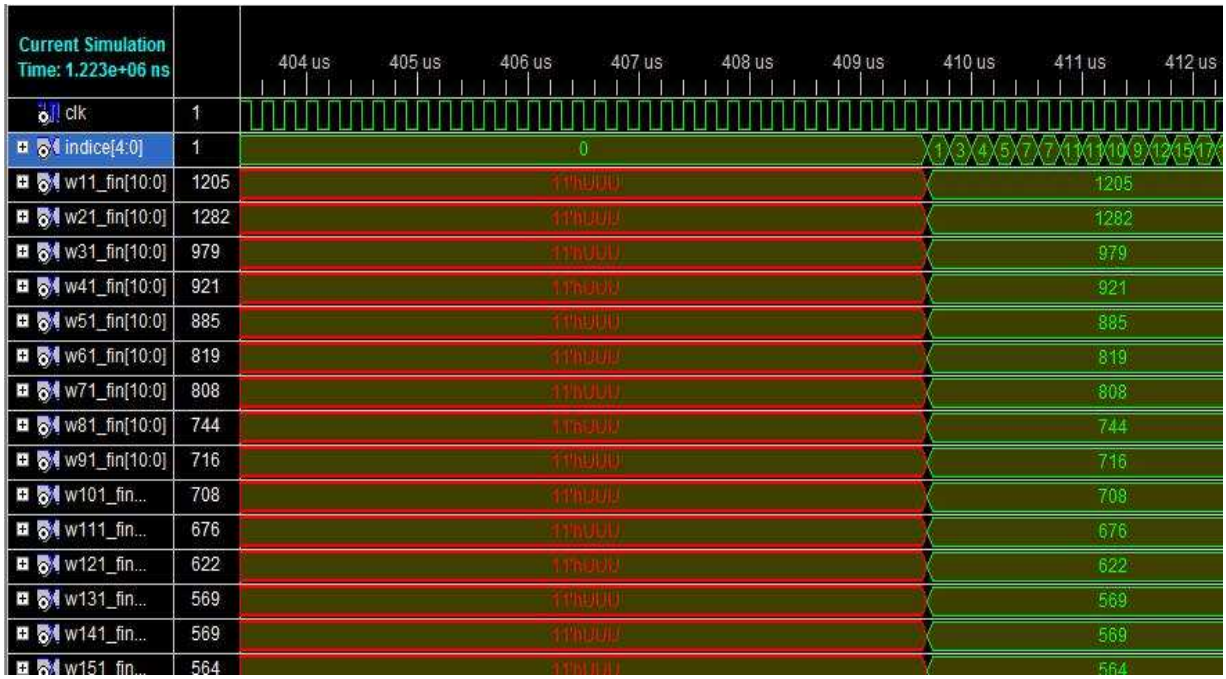


Figure VI.2 Temps de convergence du réseau

3. Taux de réussite :

Le taux de réussite de notre projet a été calculé en faisant varier les coefficients de pondération.

Le tableau suivant donne le taux de réussite en fonction des coefficients :

Premier réseau (morphologie) en %	Deuxième réseau (motif) en %	Taux de réussite en %
0	100	14.32
5	95	33.64
10	90	37.34
15	85	38.61
20	80	45.14
25	75	46.87
30	70	55.40
35	65	65.50
40	60	63.30
45	55	61.00
50	50	54.10
55	45	54.35
60	40	53.38
65	35	53.38
70	30	52.7
75	25	52.7
80	20	54.35

85	15	54.35
90	10	54.35
95	5	50.00
100	0	49.30

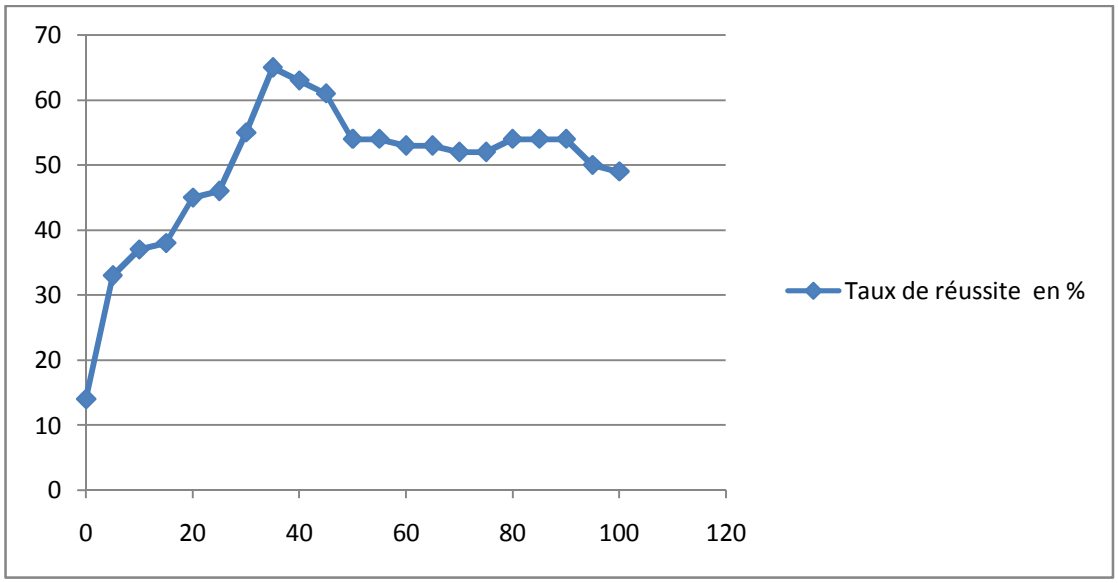


Figure VI.3 Courbe du taux de réussite en fonction du coefficient de pondération du réseau morphologique.

Cette courbe montre que les caractéristiques morphologiques donnent une meilleure classification par rapport à la densité de profil, mais il existe une combinaison des deux réseaux qui donne la valeur maximale du taux de réussite du classificateur. Ceci s'explique par la qualité des images dont nous disposons et des différentes manipulations pour l'extraction des caractéristiques.

4. Conclusion et perspectives

Comme perspectives, notre système peut être amélioré par l'ajout d'un ensemble de modules de telle manière à ce que le tout constitue une chaine de traitement allant d'un échantillon de quelques millimètres de sang prélevé à un patient jusqu'à l'image du caryotype avec le diagnostic approprié (figure VI.4)

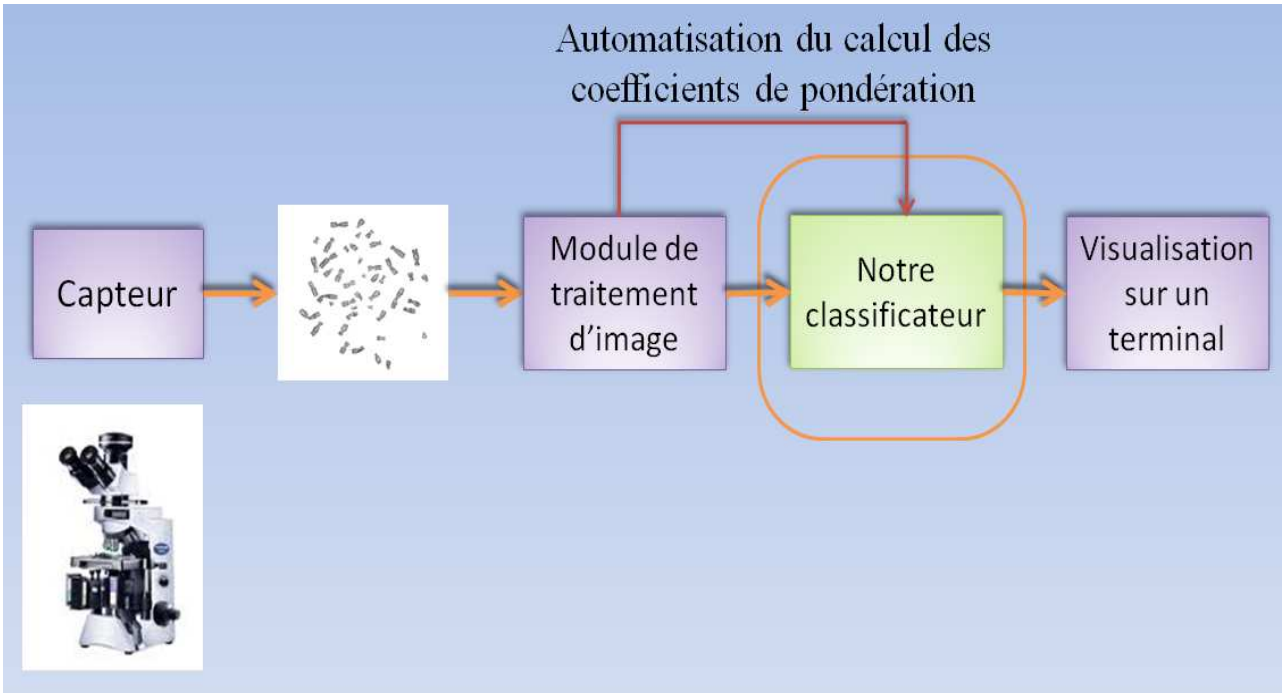


Figure VI.5 Illustration des perspectives.

On propose donc un système complet d'acquisition d'images tel un microscope électronique doté d'une caméra relié à une puce dédiés au traitement d'image à partir desquels les données nécessaires à notre classificateur sont extraites. Et au lieu d'exploiter un afficheur LCD d'une quelconque plate forme de développement, on peut visualiser les résultats sur un terminal (imprimante, écran...) qui permet de faire correspondre l'indice de chaque chromosome (c'est-à-dire la sortie de notre système) à son image pour enfin afficher le caryotype complet.

Aussi, les coefficients de pondération peuvent être calculés automatiquement en fonction des caractéristiques de l'image à traiter, à savoir le contraste, la luminosité,...etc.

CONCLUSION GÉNÉRALE

Conclusion générale

L'établissement du caryotype reste le moyen de base pour la détection primaire d'anomalies chromosomiques afin de lutter contre les maladies. Ceci a engendré pour les cytogénéticiens un besoin pressant pour un système d'aide à la classification des chromosomes en raison de la demande grandissante pour ce type d'examen.

Les réseaux de neurones et en particuliers le modèle de Kohonen semble constituer un bon classificateur. L'architecture multi réseaux adoptée ne peut que mieux conditionner le résultat final.

En effet, le traitement de manière séparée des caractéristiques morphologiques (longueur et indice centromérique) et texturales (densités de profil) et la possibilité offerte par l'architecture de pondérer l'apport de l'une et de l'autre pour la sortie finale, participe à rattraper une mauvaise présentation de l'une des caractéristiques.

Les expériences menées ont montré que les caractéristiques géométriques donnaient une meilleure classification par rapport aux densités de profil.

Ceci peut s'expliquer par la qualité des images dont la texture a été détériorée par plusieurs manipulations dans le cadre de leur numérisation.

Notre système a donné un taux de reconnaissances de 65,5 %. Ce taux même s'il n'est pas très élevé reste pour nous très encourageant car il est en grande partie tributaire de la qualité des images dont nous disposions pour les tests.

Nous espérons que notre technique puisse présenter assez d'intérêt pour avoir une continuité et être améliorée éventuellement, car nous pensons que, comme toute nouvelle conception, elle est perfectible.

Nous insistons principalement sur l'importance de la distance de Manhattan qui permet de réduire le temps de calcul, et sur le choix du support d'intégration qui doit être fait avec soin.

La connaissance, l'expérience et la patience que nous avons développées, constituent notre acquisition la plus importante au cours de la réalisation de ce projet par lequel nous espérons avoir contribué quelque peu à la mise en place d'une méthodologie pour la conception et la réalisation d'un système de classification automatique des chromosomes.

BIBLIOGRAPHIE

Bibliographie

[1] Larousse Médical, Librairie Larousse, (p 225-226), 1981

[2]A. Berkaloff, J. Bourguet, P. & N. Favard, J.-C. Lacroix, 'Biologie et physiologie cellulaire', Hermann, 1981.

[3] Anne-Marie Capodano, 'Le caryotype humain', Cours de deuxième année du premier cycle des études médicales (PCEM2), Faculté de Médecine de Marseille, Novembre 2001.

[4] A. RAHMANIA, 'Système de Détection d'Aberrations Chromosomiques Structurales', PFE, USTHB, Déc.2002. (Dirigé par Pr. L. HAMAMI, ENP)

[5] Michel Weinfeld, ''Les réseaux de neurones'', Techniques de l'ingénieur. Doc H1-990, 1999

[6] L'encyclopédie libre Wikipedia. URL : http://www.wikipedia.org/

[7] Eric Davalo, Patrick Naim, ''Des réseaux de neurones'', Edition Eyrolles, 2nde édition, 1993.

[8] Les réseaux de neurones, URL : http://www.sylbarth.com/nn.php.

[9] Saljoqi Mansour, ''Réseaux de Neurones, une introduction'', Université des Sciences de Montpellier (UM2), Master 2 Informatique Unifié Pro Recherche,2001.

[10] Dalila SALHI, 'Implémentation d'un classificateur neuronales basé sur l'algorithme RPG des rythmes cérébraux sur FPGA', Mémoire de Magister, ENP, Mars 2009.

[11] Réseaux de Kohonen, URL : http://www.ensta.fr/~bmonsuez/Cours/doku.php?id=in104:projets:filliat.

[12] R.C. Gonzales, P. Wintz, 'Digital Image Processing', Addison Wessley, 1977.

[13] BRIKI Yacine, SMATI Djamel, 'Contribution à la conception d'un outil d'aide pour le choix d'une technique de traitement d'image'. Mémoire de fin d'étude, INI, Juin 2003.

[14] G. Dreyfus, J-M. Martiez, M. Samuelides, ''Réseaux de neurones : methodologie et application'', Edition Eyrolles 2nde édition, 2002.

[15] Ahmed Naassani, Eric Monmasson, Mohamed Wissem Naour, ''Commandes numeriques à base de composant FPGA'', Techniques de l'ingénieur. D2902, Novembre 2008.

[16] Khelifaoui Mohamed, Guergeb Mohamed, ‘‘Utilisation des cartes auto-organisatrices de Kohonen dans la recherche documentaire‘‘, Decembre 2003.

[17] Teuvo Kohonen, ‘‘The Self Organisation Map‘‘, (p1464-1480), 1999.

[18] Patrick Rousset, ‘‘Applicatin des algorithmes d’auto-organisation à la classifaction et à la prévision‘‘, Thèse de Doctorat de l’université Paris I, France 1999.

[19] Lionel Dricot, Pierre Lison, ‘Conception d’un classifieur homme/femme pour la reconnaissance de visages’, Université Catholique de Louvain, Faculté des Sciences Appliquées, Département d’Ingénierie Informatique, 23 décembre 2005.

[20] Basma M. K. Younis, Basil Sh. Mahmood, Fakhraldeen H. Ali, ‘Reconfigurable Self-Organizing Neural Network Design and it's FPGA Implementation’, 1998.

[21] Vincent Lemaire, ‘Cartes auto-organisatrices pour l’analyse de données’, 2005.

[22] Marie Cottrell, Patrick Letremy, ‘Algorithme de Kohonen : classification et analyse exploratoire des données’, Université Paris 1- Sorbonne, 2004.

[23] Spartan-3E FPGA Family, Data Sheet, DS312 (v3.8) August 26, 2009, www.xilinx.com.

[24] David Fritz, ‘Spartan 3 FPGA Tutorial’, Oklahoma State University, June 2005.

[25] Houria AIT-ABDESSLAM, ‘Réalisation d’un système d’aide à la détection des aberrations chromosomiques’, Mémoire de Magister, ENP, Février 2005.

[26] M. Turhan Taner, ‘Kohonen’s Self Organizing Map’, Novembre 1997.

[27] Peter J. Ashenden, ‘VHDL Quick start’, The University of Adelaide, 1999.

[28] Philippe Lecardonnel, Philippe Letenneur, ‘Introduction à la synthèse logique VHDL’, Lycée Julliot de la Morandière, Granville, 2001.

ANNEXES

Code source Matlab du codage de la base de données :

```
%Script Matlab : codage de la base de données
N=46 ;
For  j=1:N
l(j)=input(' ');                %pour introduire les données
end
for j=1:N
b=l(j)/0.0078125;               %codage en virgule fixe
h=round(b);
t=dec2bin (h,11);               %faire la conversion décimale - binaire
fprintf('rom(%d)<="%s";\n',j,t)     %représentation en mémoire des données
                                %       adapté au langage VHDL
 end
```

Description VHDL du générateur pseudo aléatoire à 10 étages :

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

 entity generateur_pseudo_aleatoire is
 port (clk :in std_logic;
 Q :out std_logic_vector(9 downto 0));
 end generateur_pseudo_aleatoire;

 architecture behavioral of generateur_pseudo_aleatoire is

 signal X :std_logic_vector(9 downto 0):=(others=>'0');
 signal Y :std_logic:='1';

 begin
 process(clk)
 begin
 if (clk'event and clk='1') then
 X <= X(8 downto 0) & Y;
 end if;
 end process;

 Y <= not (X(9) xor X(1));
 Q <= X;

 end behavioral;
```

Description VHDL d'une rom

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

 entity rom is
```

```
 port
 (
 data_out: out std_logic_vector(10 downto 0);
 address: in std_logic_vector(9 downto 0)
 );

 end rom;

 architecture behavioral of rom_L is
 type memoire is array(0 to 1023) of std_logic_vector(10 downto 0);
 signal rom: memoire;

 begin
 read: process(address)
 variable index: integer range 0 to 1023 := 0;
      rom(0)<="10100000011";
      rom(1)<="10000000001";
      rom(2)<="11100011111";
      rom(3)<="10111001011";
      rom(4)<="01000110100";
      rom(5)<="01000000101";









      rom(1022)<="00100000101";
      rom(1023)<="11000100101";
index := conv_integer( address(9 downto 0));
data_out <= rom(index);
end process;
end behavioral;
```

Description VHDL du bloc de distance de Manhattan :

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

entity bloc_de_calcul_de_la_distance_de_manhattan is

Port ( w1 : in STD_LOGIC_VECTOR (10 downto 0);
       w2 : in STD_LOGIC_VECTOR (10 downto 0);
       x1 : in STD_LOGIC_VECTOR (10 downto 0);
       x2 : in STD_LOGIC_VECTOR (10 downto 0);
        s : out STD_LOGIC_VECTOR (11 downto 0));
 end bloc_de_calcul_de_la_distance_de_manhattan;
```

```
architecture Behavioral of bloc_de_calcul_de_la_distance_de_manhattan
is

signal s1,s2 : std_logic_vector(10 downto 0);

component soustracteur is
Port (signe : out STD_LOGIC;
a : in STD_LOGIC_VECTOR (10 downto 0);
b : in STD_LOGIC_VECTOR (10 downto 0);
d : out STD_LOGIC_VECTOR (10 downto 0));
```

```
end component soustracteur;

component accumulateur_12bits is
Port ( a : in STD_LOGIC_VECTOR (10 downto 0);
b : in STD_LOGIC_VECTOR (10 downto 0);
s : out STD_LOGIC_VECTOR (11 downto 0));
end component accumulateur_12bits;

begin
bit0: soustracteur port map ( a => x1, b=> w1 , d => s1);
bit1: soustracteur port map ( a => x2, b=> w2 , d => s2);
bit2: accumulateur_12bits port map ( a => s1, b=> s2 , s => s);
end Behavioral;
```

Description VHDL du bloc « find_min_25 » :

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

 entity find_min_of_25 is

Port (
e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12,e13, e14, e15,e16,
e17, e18,e19, e20, e21, e22,e23, e24,e25 : in STD_LOGIC_VECTOR(11
downto 0):=(others=>'0');
s : out STD_LOGIC_VECTOR (11 downto 0);
 indice : out STD_LOGIC_VECTOR (4 downto 0):=(others=>'0'));
 end find_min_of_25;

 architecture Behavioral of find_min_of_25 is

 component find_min_of_8

 port ( e1, e2, e3, e4,e5, e6, e7, e8 : in STD_LOGIC_VECTOR (11
downto 0) ;
 s : out STD_LOGIC_VECTOR (11 downto 0);
 indice : out STD_LOGIC_VECTOR(2 downto 0) :=(others=>'0'));
 end component find_min_of_8;

 component find_min_of_4 is

 Port ( e1,e2,e3,e4 : in STD_LOGIC_VECTOR (11 downto 0);
 s : out STD_LOGIC_VECTOR (11 downto 0);
 indice : out STD_LOGIC_VECTOR(1 downto 0):=(others=>'0'));
 end component find_min_of_4;

 signal s1, s2, s3 : STD_LOGIC_VECTOR (11 downto 0);
 signal i1,i2,i3 : STD_LOGIC_VECTOR(2 downto 0);
 signal i4 : STD_LOGIC_VECTOR(1 downto 0);

 begin

 bloc1 : find_min_of_8 port map (e1=>e1 , e2=>e2 , e3=>e3 , e4=>e4 ,
e5=>e5 ,e6=>e6 , e7=>e7 ,e8=>e8 , s=>s1 , indice=>i1);

 bloc2 : find_min_of_8 port map (e1=>e9 , e2=>e10 ,e3=>e11 , e4=>e12
, e5=>e13 ,e6=>e14 ,e7=>e15 , e8=>e16 , s=>s2 , indice =>i2);

 bloc3 : find_min_of_8 port map (e1=>e17 ,e2=>e18 ,e3=>e19 , e4=>e20
,e5=>e21 ,e6=>e22 ,e7=>e23 , e8=>e24 , s=>s3 , indice => i3);

 bloc4 : find_min_of_4 port map (e1=>s1 ,e2=>s2 ,e3=>s3 ,e4=>e25 ,
s=>s , indice=>i4);
```

```
 process(i1,i2,i3,i4)
 begin
 if    i4="00" then indice<=("00" & i1) + "00001";
 elsif i4="01" then indice<=("00" & i2) + "01001";
 elsif i4="10" then indice<=("00" & i3) + "10001";
 else  indice<="11001"; end if;

 end process m;

 end Behavioral;
```

La description VHDL du multiplicateur de l'unité de calcul :

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

entity bloc_multiplication is

port(
      diff :in std_logic_vector(10 downto 0);
      alpha:in std_logic_vector(6 downto 0);
       mul:out std_logic_vector(17 downto 0));

end bloc_multiplication;


architecture Behavioral of bloc_multiplication is

begin

process(diff)
variable i:std_logic_vector(17 downto 0):=(others=>'0');
variable ai:std_logic_vector(17 downto 0):=(others=>'0');
begin
  for j in 0 to 6 loop
   if alpha(j)='1' then
    ai(j+10 downto j):=diff;
    i:=i+ai;
    ai:=(others=>'0');
   end if;
  end loop;
 mul<=i;
 i:=(others=>'0');
end process;

end Behavioral;
```

Description VHDL du diviseur par 128:

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

 entity diviseur is
 Port (
 vecteur : in std_logic_vector (17 downto 0);
 resultat: out std_logic_vector (17 downto 0));
 end diviseur;
```

```
 architecture Behavioral of diviseur is

 signal c,c0: integer range 0 to 2048;
 signal b0: integer range 0 to 63;
 signal a,b,n: integer ;

 begin

 b<=128;
 b0<=63;
 a <= CONV_INTEGER(vecteur);
 n<=a mod b;
 c0<=a/b;
 c <= c0 when n<= b0 --arrondir le résultat
 else c0+1;
 resultat<= CONV_STD_LOGIC_VECTOR (c ,18);

end Behavioral;
```

La description VHDL du bloc de mise à jour des poids synaptique du J ème neurone :

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

 entity bloc_mise_a_jour_Jeme_neurone is
 generic(
 w10:std_logic_vector(10 downto 0):="10011100110";--initiation des
poids
 w20:std_logic_vector(10 downto 0):="01001000000";--synaptiques
 indice_local:integer range 0 to 24:=1; --l'indice du neurone
 voisin1:integer range 0 to 24:=2; --le voisinage du J eme neurone
 voisin2:integer range 0 to 24:=3);

 port( indice:in std_logic_vector(4 downto 0);
          clk:in std_logic;
           x1:in std_logic_vector(10 downto 0);
           x2:in std_logic_vector(10 downto 0);
        alpha:in std_logic_vector(6 downto 0);
       w1_new:out std_logic_vector(10 downto 0);
       w2_new:out std_logic_vector(10 downto 0) );
 end bloc_mise_a_jour_Jeme_neurone;

 architecture Behavioral of bloc_mise_a_jour_Jeme_neurone is
 component ram is
 Port ( r1 : in STD_LOGIC_vector(10 downto 0):=w10;
        r2 : in STD_LOGIC_vector(10 downto 0):=w20;
        w1 : out STD_LOGIC_vector(10 downto 0);
        w2 : out STD_LOGIC_vector(10 downto 0);
       clk : in STD_LOGIC );
 end component ram;

 component unite_de_calcul is
 Port ( w,x : in STD_LOGIC_VECTOR (10 downto 0);
      alpha : in STD_LOGIC_VECTOR (6 downto 0);
        clk : in STD_LOGIC;
          p : out STD_LOGIC;
 s1,s2,s3,s4: out STD_LOGIC_VECTOR (10 downto 0));
 end component unite_de_calcul;

 signal set:integer range 0 to 2;
 signal n:integer range 0 to 24;
 signal p1,p2 : STD_LOGIC;
```

```
 signal s1,s2,s3,s4,s5,s6,s7,s8:STD_LOGIC_VECTOR (10 downto 0);
 signal w1: std_logic_vector(10 downto 0):=w10;
 signal w2: std_logic_vector(10 downto 0):=w20;
 signal h1,h2: std_logic_vector(10 downto 0);

 begin
 bit0: ram port map (clk=>clk,r1=>w1,r2=>w2,w1=>h1,w2=>h2);
 bit1:unite_de_calcul port map
(w=>h1,x=>x1,alpha=>alpha,p=>p1,s1=>s1,s2=>s2,s3=>s3,s4=>s4,clk=>clk)
;
 bit2:unite_de_calcul port map
(w=>h2,x=>x2,alpha=>alpha,p=>p2,s1=>s5,s2=>s6,s3=>s7,s4=>s8,clk=>clk)
;
 process(clk)

 begin

 n <= CONV_INTEGER(indice);

 if   (n=indice_local)then set<=0;
 elsif(voisin1=n or voisin2=n)then set<=1;
 else  set<=2; end if;

 if   (p1='1' and set=0) then w1<=s1;
 elsif(p1='0' and set=0) then w1<=s2;
 elsif(p1='1' and set=1) then w1<=s3;
 elsif(p1='0' and set=1) then w1<=s4;
 else w1<=w1; end if;

 if   (p2='1' and set=0) then w2<=s5;
 elsif(p2='0' and set=0) then w2<=s6;
 elsif(p2='1' and set=1) then w2<=s7;
 elsif(p2='0' and set=1) then w2<=s8;
 else w2<=w2; end if;

 w1_new<=w1;
 w2_new<=w2;
 end process;

 end Behavioral;
```

Code source Matlab comparant la distance de Manhattan avec la distance Euclidienne :

```
fprintf('génération aleatoire des poids synaptiques:')

w= rand(2,10)
for n=1:10

 fprintf('---------------le cas numéro:%d--------------\n',n)
 x=rand(2,1)
 d1=zeros(1,8);
 d2=zeros(1,8);

 for j=1:8
  for i=1:2
  d1(j)=d1(j)+[x(i)-w(i,j)]*[x(i)-w(i,j)];
  d2(j)=d2(j)+abs (x(i)-w(i,j));
  end
 end

 fprintf('les distances euclidiennes:')
 d1
```

```
 fprintf('les distances de manhatten:')
 d2

 ref=d1(1);
 ind=1;
 for i=2:8
  if (ref < d1(i))
  ref=ref;
  else ref=d1(i);
  ind=i;
  end
 end

 fprintf('indice du min de la distance euclidienne:')
 ind
 ref
 ref2=d2(1);
 ind2=1;
 for i=2:8
  if (ref2 < d2(i))
  ref2=ref2;
  else ref2=d2(i);
  ind2=i;
  end
 end
 fprintf('indice du min de la distance de manhatten:')
 ind2
 ref2
end
```

Résultat d'exécution du code source Matlab comparant la distance de Manhattan avec la distance Euclidienne

```
Génération aléatoire des poids synaptiques:
w =
 0.9501  0.6068  0.8913  0.4565  0.8214  0.6154  0.9218  0.1763  0.9355
0.4103
 0.2311  0.4860  0.7621  0.0185  0.4447  0.7919  0.7382  0.4057  0.9169
0.8936

----------------------------le cas numéro:1----------------------------
x =
 0.0579
 0.3529
les distances euclidiennes:
d1 =
 0.8109  0.3191  0.8620  0.2707  0.5914  0.5036  0.8948  0.0168
les distances de Manhattan:
d2 =
 1.0140  0.6821  1.2426  0.7329  0.8554  0.9966  1.2493  0.1712
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.0168
indice du min de la distance de Manhattan:
ind2 =
 8
ref2 =
 0.1712
----------------------------------le cas numéro:2----------------------
x =
 0.8132
 0.0099
```

```
les distances euclidiennes:
d1 =
 0.0677  0.2693  0.5720  0.1273  0.1892  0.6507  0.5423  0.5623
les distances de Manhattan:
d2 =
 0.3582  0.6824  0.8304  0.3653  0.4431  0.9798  0.8370  1.0327
indice du min de la distance euclidienne:
ind =
 1
ref =
 0.0677
indice du min de la distance de Manhattan:
ind2 =
 1
ref2 =
 0.3582
-----------------------------------le cas numéro:3----------------------
x =
 0.1389
 0.2028
les distances euclidiennes:
d1 =
 0.6589  0.2992  0.8790  0.1348  0.5244  0.5742  0.8997  0.0426
les distances de Manhattan:
d2 =
 0.8396  0.7512  1.3117  0.5018  0.9245  1.0657  1.3184  0.2403
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.0426
indice du min de la distance de Manhattan:
ind2 =
 8
ref2 =
 0.2403
-----------------------------------le cas numéro:4----------------------
x =
 0.1987
 0.6038
les distances euclidiennes:
d1 =
 0.7035  0.1804  0.5047  0.4090  0.4130  0.2090  0.5409  0.0397
les distances de Manhattan:
d2 =
 1.1241  0.5259  0.8509  0.8430  0.7818  0.6049  0.8575  0.2205
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.0397
indice du min de la distance de Manhattan:
ind2 =
 8
ref2 =
 0.2205
-----------------------------------le cas numéro:5----------------------
x =
 0.2722
 0.1988
les distances euclidiennes:
d1 =
 0.4606  0.1945  0.7006  0.0665  0.3621  0.4696  0.7130  0.0520
les distances de Manhattan:
d2 =
 0.7103  0.6218  1.1824  0.3646  0.7951  0.9364  1.1890  0.3028
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.0520
indice du min de la distance de Manhattan:
ind2 =
```

```
 8
ref2 =
 0.3028
----------------------------------le cas numéro:6---------------------
x =
 0.0153
 0.7468
les distances euclidiennes:
d1 =
 1.1398  0.4180  0.7677  0.7250  0.7411  0.3622  0.8219  0.1423
les distances de Manhattan:
d2 =
 1.4505  0.8524  0.8913  1.1695  1.1082  0.6453  0.9151  0.5021
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.1423
indice du min de la distance de Manhattan:
ind2 =
 8
ref2 =
 0.5021

----------------------------------le cas numéro:7---------------------
x =
 0.4451
 0.9318

les distances euclidiennes:
d1 =
 0.7460  0.2249  0.2279  0.8343  0.3789  0.0486  0.2647  0.3491
les distances de Manhattan:
d2 =
 1.2057  0.6076  0.6159  0.9247  0.8634  0.3102  0.6703  0.7949
indice du min de la distance euclidienne:
ind =
 6
ref =
 0.0486
indice du min de la distance de Manhattan:
ind2 =
 6
ref2 =
 0.3102
----------------------------------le cas numéro:8---------------------
x =
 0.4660
 0.4186
les distances euclidiennes:
d1 =
 0.2695  0.0244  0.2988  0.1602  0.1270  0.1617  0.3099  0.0841
les distances de Manhattan:
d2 =
 0.6716  0.2082  0.7688  0.4097  0.3815  0.5227  0.7754  0.3027
indice du min de la distance euclidienne:
ind =
 2
ref =
 0.0244
indice du min de la distance de Manhattan:
ind2 =
 2
ref2 =
 0.2082
----------------------------------le cas numéro:9---------------------
x =
 0.8462
 0.5252
les distances euclidiennes:
d1 =
 0.0972  0.0588  0.0582  0.4086  0.0071  0.1244  0.0511  0.4631
les distances de Manhattan:
```

```
d2 =
 0.3979  0.2785  0.2820  0.8964  0.1053  0.4976  0.2886  0.7894
indice du min de la distance euclidienne:
ind =
 5
ref =
 0.0071
indice du min de la distance de Manhattan:
ind2 =
 5
ref2 =
 0.1053


-----------------------------------le cas numéro:10--------------------
x =
 0.2026
 0.6721
les distances euclidiennes:
d1 =
 0.7532  0.1980  0.4823  0.4917  0.4346  0.1847  0.5216  0.0717


les distances de Manhattan:
d2 =
 1.1885  0.5904  0.7786  0.9075  0.8462  0.5326  0.7852  0.2928
indice du min de la distance euclidienne:
ind =
 8
ref =
 0.0717
indice du min de la distance de Manhattan:
ind2 =
 8
ref2 =
 0.2928
>>
```

Les afficheurs à cristaux liquides :

Les afficheurs à cristaux liquides, autrement appelés afficheurs LCD (Liquid Crystal Display), sont des modules compacts et nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu (de 1 à 5 mA), ils sont bons marchés et s'utilisent avec beaucoup de facilité.

Principe de fonctionnement :

L'afficheur est constitué de deux lames de verre, distantes de 20 µm environ, sur lesquelles sont dessinées les mantisses formant les caractères. L'espace entre elles est rempli de cristal liquide L'application entre les deux faces d'une tension alternative basse fréquence de quelques volts (3 à 5 V) rend le cristal liquide absorbant. Les caractères apparaissent sombres sur fond clair.

Brochage :

Broche	Nom	Fonction
1	Vss	Masse
2	Vdd	Alimentation positive + 5V
3	V0	Cette tension permet, en la faisant varier entre 0 et +5V, le réglage du contraste de l'afficheur.
4	RS	Sélection du registre (Register Select) Grâce à cette broche, l'afficheur est capable de faire la différence entre une commande et une donnée. un niveau bas indique une commande et un niveau haut indique une donnée.
5	R/W	Lecture ou écriture (Read/Write) L : Écriture H : Lecture
6	E	Entrée de validation (Enable) active sur front descendant. Le niveau haut doit être maintenu pendant au moins 230 ns à l'état haut.
7	D0	Bus de données bidirectionnel 3 états (haute impédance lorsque E=0)
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	Anode rétro éclairage (+5V)
16	K	Cathode rétro éclairage (masse)

Les connexions à réaliser sont donc simples puisque l'afficheur LCD dispose de peu de broches. Il faut seulement, l'alimenter, le connecter à un bus de données (4 ou 8 bits), et connecter les broches **E**, **R/W** et **RS**.

Les mémoires :

L'afficheur possède deux type de mémoire, la DD RAM et la CG RAM. La DD RAM est la mémoire d'affichage et la CG RAM est la mémoire du générateur de caractères.

La mémoire d'affichage (DD RAM) :
La DD RAM est la mémoire qui stocke les caractères actuellement affichés à l'écran. Pour un afficheur de 2 lignes de 16 caractères, les adresses sont définies de la façon suivante :

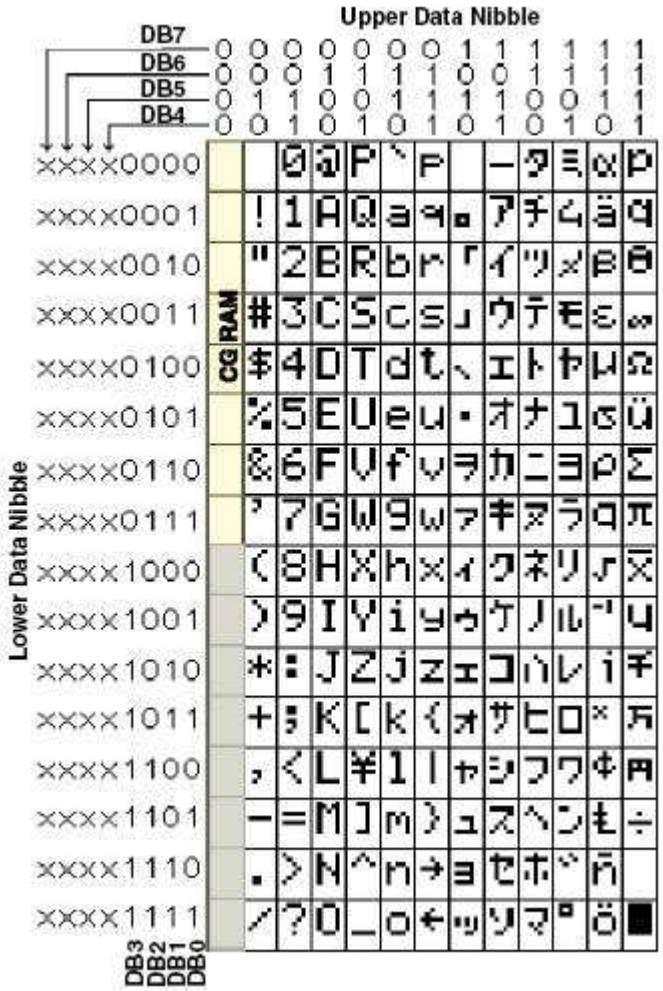
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

La mémoire du générateur de caractères (CG RAM) :

La mémoire du générateur de caractères (CG Ram) permet d'afficher l'ensemble des caractères ASCII, ainsi que quelques symboles, on peut tout de même afficher tous les caractères désirés grâce à des caractères définissables par l'utilisateur. Ceux-ci sont au nombre de huit et correspondent aux caractères de 0 à 7.

L'afficheur est en mesure d'afficher 200 caractères :

- de 00H à 07H : 8 caractères définissables par l'utilisateur.
- de 20H à 7FH : 96 caractères ASCII .
- de A0H à DFH : 64 caractères japonais (alphabet kana).
- de E0H à FFH : 32 caractères spéciaux (accent, lettres grecques, ...).



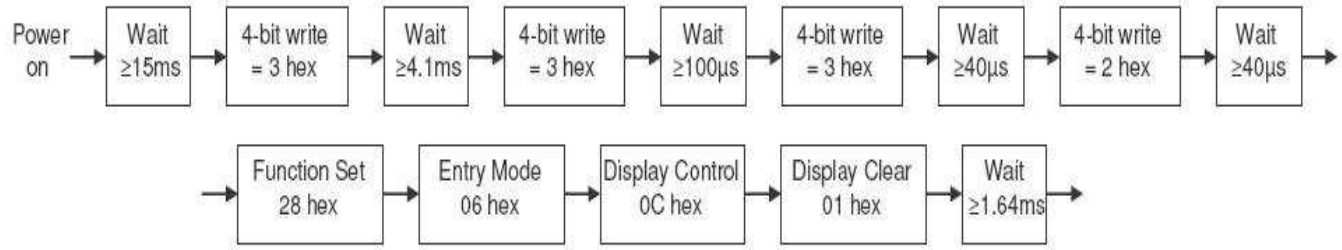
Commande d'un afficheur LCD

Avant tout il faut savoir que l'afficheur LCD a deux modes de fonctionnement : le mode 4 bits et le mode 8 bits, modes que l'on choisira à l'initialisation de l'afficheur (voir plus bas).

Dans le mode 8 bits, les données sont envoyées à l'afficheur sur les broches **D0** à **D7.** Mais il se peut que dans certains cas, il s'avère nécessaire de diminuer le nombre de fils utilisés pour commander l'afficheur, comme c'est le cas pour le kit de développement de la carte FPGA Spartan 3-E de Xilinx, qui ne dispose que de 4 broches pour lire les données. Dans ce cas on utilise le mode 4 bits de l'afficheur LCD. Dans ce mode , seuls les 4 bits de poids fort (D4 a D7) sont utilisés pour transmettre les données.

Initialisation :

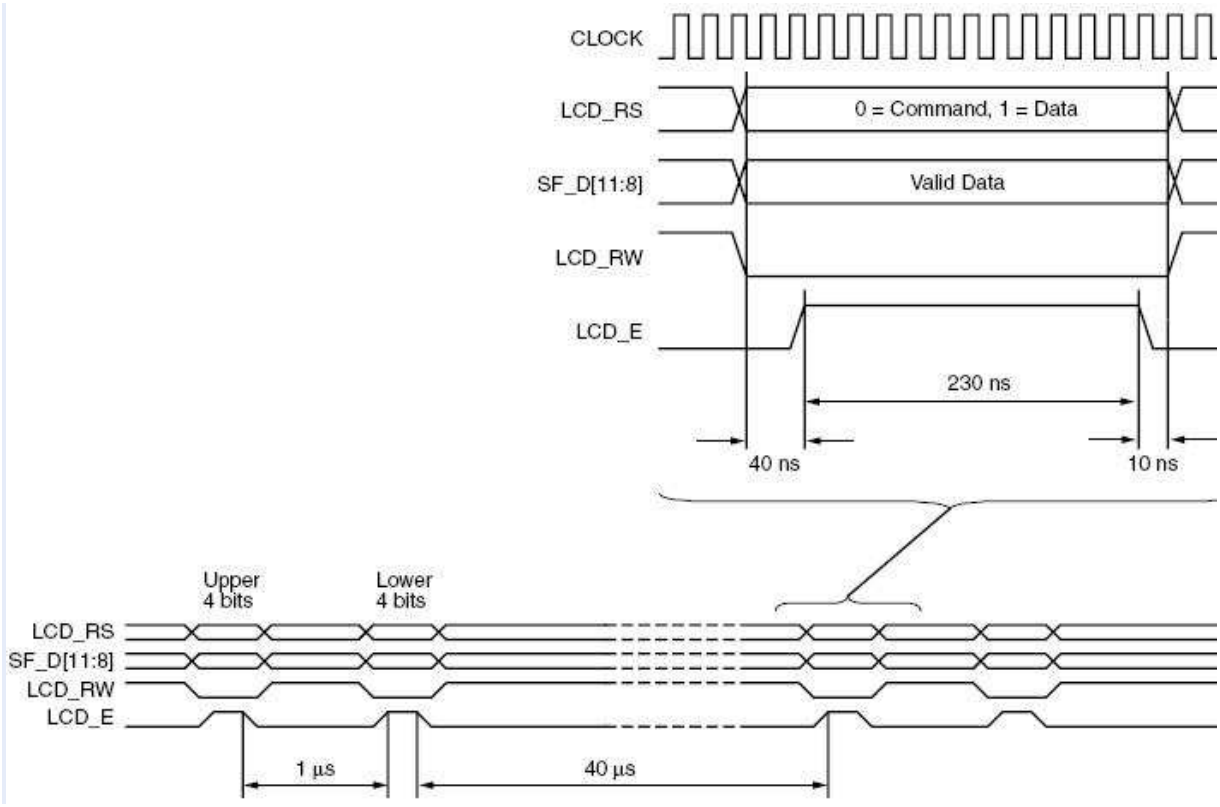
Avant de pouvoir utiliser l'afficheur, il faut tout d'abord l'initialiser. En résumé, voici sur quatre bits, les commandes (RS=0) à envoyer à l'afficheur LCD pour l'initialiser. Entre chaque valeur, il faut envoyer une impulsion positive sur la ligne E et maintenir R/W à niveau bas(en écriture) :



- **0H, 1H, 0H, 0H, 1H :** on commence par effacer l'afficheur.
- **3H, 3H, 3H :** on force le LCD en mode 8 bits.
- **2H :** on passe en mode 4 bits .ne pas envoyer cette commande en mode 8 bits.
- **2H, 8H :** mode 4 bits, 2 lignes, caractères 5x7 . **3H, 8H :** pour le mode 8 bits.
- **0H, CH :** affichage sans curseur.
- **0H, 6H :** le curseur se déplace vers la gauche.
- **0H, 1H :** on efface l'affichage.

Chargement des données :

La figure illustre une opération d'écriture sur l'écran LCD. En indiquant le temps nécessaire pour la validation des données présentées a l'entrée des broches D4 à D7.



Exemple d'affichage :

Voila les étapes qu'on a suivies pour l'affichage des mots « Ecole Nationale Polytechnique » sur l'afficheur LCD du Kit de développement FPGA spartan 3-E :

La première étape : on doit commencer par l'initialisation de l'afficheur.

Une fois l'initialisation faite le curseur se situe automatiquement à l'adresse 00H,

pour écrire la lettre E à cette adresse, on doit présenter aux broches D4 à D7 les bits correspondant à la lettre E dans le tableau du CG RAM .

(Le Kit Spartan 3-E n'est relié à l'afficheur LCD que par 4 broches de données donc, on va travailler obligatoirement en mode 4 bits).

Les 4 bits supérieurs de la lettre E (ou le Upper Nibble) sont :0100.

Les 4 bits inferieurs de la lettre E (ou le lower Nibble) sont :0101.

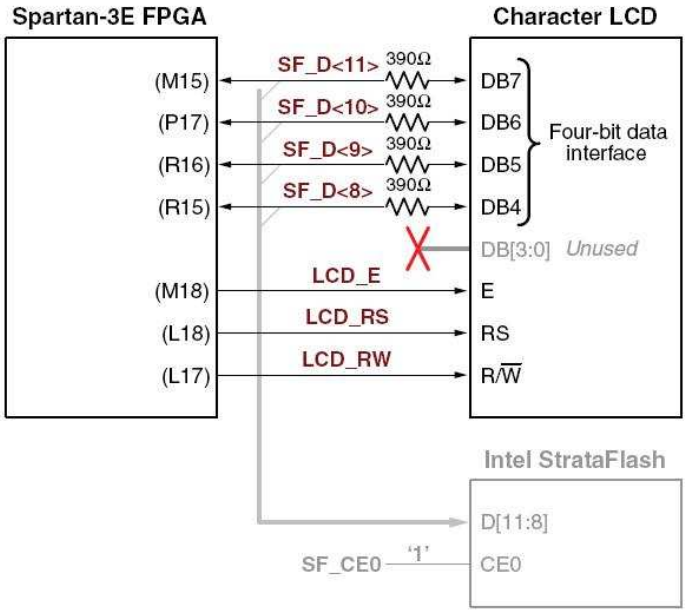
Donc pour afficher un E, on envoie 0100 sur les broches data(D4 à D7). Comme on est en mode écriture la broche R /W doit être mise à 0, RS =1 puisque on envoie une donnée, mais la broche E qui était initialement au niveau bas doit être maintenue au niveau haut pendant 230 ns c.à.d. 12 cycles d'horloge (la Spartan 3-E dispose d'un oscillateur quartz qui délivre une

fréquence de 50 Mhz). Lorsque l'entrée Enable retourne à son état initial la donnée sera validée. Puis il faudrait attendre 1µs (50 cycles d'horloge) avant d'envoyer le lower Nibble de la lettre « E ».

Une fois cette donnée validée, il faut encore attendre 40 µs (2000 cycles d'horloge) avant d'envoyer le upper Nibble de la lettre suivante c.à.d. la lettre « c ».

Lorsqu'on arrive à l'adresse 0F H le curseur ne retourne pas automatiquement a la deuxième ligne, il faut envoyer une commande (RS=0) de retour a la ligne, et data<=C0 H , le curseur se situe à l'adresse 40H. Idem lorsqu'on arrive à l'adresse 4F, et qu'on veut afficher une nouvelle page on envoie la commande data<=80 H qui ramène le curseur à l'adresse 00H.

Branchement avec la carte FPGA spartan 3-E :



Les I /O de la carte FPGA destinées à être reliées à l'afficheur LCD sont indiquées dans la figure ci-dessus, la mémoire Flash : Intel StrataFlash, permet la sauvegarde temporaire des données.

Pour piloter l'afficheur LCD par la carte FPGA il reste seulement la génération du fichier de contrainte (.ucf). La figure suivante fournit les affectations des broches d'I/O contenues dans le fichier de contrainte.

```
NET "LCD_E"   LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS"  LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW"  LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

# The LCD four-bit data interface is shared with the StrataFlash.
NET "SF_D<8>"  LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>"  LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```