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

ECOLE NATIONALE POLYTECHNIQUE
Département d'Electronique

MEMOIRE
Présenté par
Mr BELAIFA SALAH SALIM
Ingénieur d'état en électronique(ENP)

Pour l'obtention du

MAGISTER EN ELECTRONIQUE
Option: Système de traitement de l'information

Thème

PERFORMANCES DES TECHNIQUES SEQUENTIELLE ET SYSTOLIQUE D'IMPLEMENTATION HARDWARE DES RESEAUX DE NEURONES

Soutenue le 10 / 2 /2001

Devant le jury composé de :

Mr	**F.Boudjema**	Professeur (E.N.P)	Président
Mr	**M. Mehenni**	Maître de conférences (E.N.P)	Rapporteur
Mr	**C. Larbes**	PHD (E.N.P)	Examinateur
Mr	**A. Belouchrani**	Docteur d'état (E.N.P)	Examinateur
Mlle	**N.Izeboudjen**	Chercheur (C.D.T.A)	Invitée

Dédicace

A la mémoire de mon très cher et brave père .

A ma très chère mère.

A mes sœurs et frères.

A tous les amis et ceux qui me sont chers

salim

Remerciements

Je tiens en premier lieu à remercier Mr M. Mehenni maître de conférences et chercheur au laboratoire des techniques digitales et systèmes de l'ENP, pour son aide, son suivi et ses conseils qui m'ont permis de mener cette thèse à son aboutissement.

Je remercie ensuite Mr A. Farah professeur à l'ENP d'avoir proposé ce sujet.

De plus je tiens à remercier vivement Mr F. Boudjema professeur à L'ENP d'avoir accepté de présider ce jury.

Mes remerciements s'adressent aussi à : Mr A. Bellouchrani Docteur d'état et chercheur au laboratoire de signal et communication de L'ENP, Mr C. Larbes PHD et chercheur au laboratoire d'Automatique de L'ENP et Mlle N. Izeboudjen chercheur au laboratoire de microelectronique du Centre de Développement des Technologies Avancées (CDTA) d'avoir acceptés d'être dans mon jury.

SOMMAIRE

SOMMAIRE

Chapitre IV : Les architectures hardware des RNAs

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Historiquement, l'intérêt porté aux réseaux de neurones provient du désir de saisir les principes menant à la compréhension des fonctions du cerveau, d'une part, et de construire des taches complexes d'autres part .

La neurobiologie à découvert que la puissance de traitement dans les couches du cerveau humain est due au grand nombre d'unités de traitement identiques (neurones), liés entre eux par des longueurs variables en terme de réseau de poids synaptiques.

Cette puissance de traitement neuronal est la cause de l'apparition d'un large espace de recherche dans le domaine des réseaux de neurones artificiels (*RNAs*) incluant la modélisation neural, algorithmes, architectures, implémentations et applications .

Aujourd'hui , les réseaux de neurones trouvent leur application dans divers domaines tels : L'identification, la classification, le reconnaissance des formes....etc.

L'implémentation hardware nécessite des méthodes et des techniques modernes pour contourner les problèmes de conception et de réalisation traditionnelles, surtout en ce qui concerne la justesse, la fiabilité de la conception, ce qui a poussé les développeurs du hardware à réfléchir aux nouvelles méthodes et techniques flexibles et efficaces permettant de développer des circuits à très large échelle d'intégration.

L'évolution rapide de la technologie et spécialement les outils de CAO ont apporté de nouvelles techniques de développement du matériel tout en utilisant la modélisation et la simulation logicielle avant de passer à son implémentation matérielle .

La partie software à pour tâche l'apprentissage, celui-ci étant basé un algorithme d'apprentissage , on choisi dans notre cas l'algorithme de rétrpropagation du gradient (*RPG*) Afin de calculer les synapses pour par la suite les stocker dans des mémoires pour la tâche de généralisation (partie hardware).

La partie hardware consiste à implémenter la phase de généralisation de l'algorithme .

Dans notre cadre de cette thèse de magister, nous nous sommes intéressés à l'implémentation digitale d'un *RNAs* pour les architectures séquentielle et systolique, afin de déduire leur performances en ce qui concerne

- la complexité architecturale (nombre de blocs).
- la rapidité.
- le parallélisme.

Le travail de ce mémoire se compose des chapitres suivants :

- Le premier chapitre est consacré aux fondements théoriques des réseaux de neurones, leurs différents modèles ainsi que les algorithmes d'apprentissage associés à chaque modèle,

- Dans le deuxième chapitre, nous présentons l'algorithme de rétropropagation du gradient Comme un algorithme d'apprentissage pour un exemple d'application,

- Le troisième chapitre présentera les fondements de base de la technologie *FPGA,* utilisés pour la conception des circuits digitaux,

- Dans le quatrième chapitre nous proposons les architectures hardware séquentielle et systolique, pour implémenter un réseau de neurones,

- Le cinquième chapitre est consacré aux inconvénients et avantages des deux implémentations.

- finalement nous terminons par une conclusion générale.

CHAPITRE I
RESEAUX DE NEURONES

I.1.Introduction :

Les modèles connexionistes neuronaux sont des systèmes qui tentent de simuler les mécanismes de traitement de l'information ayant lieu dans le cerveau humain.**[5]**

Le premier problème de la formalisation du fonctionnement du cerveau est celui de l'apprentissage, c'est à dire la capacité d'acquérir de nouveaux comportements ou d'en modifier par expérience.

L'apprentissage va permettre au réseau de modifier sa structure interne (poids synaptiques) pour s'adapter à son environnement. En effet, nous n'avons pas besoin d'une formulation rigoureuse d'un problème donné pour le résoudre, tout ce dont nous avons besoin est une collection d'exemples représentative de la fonction désirée.

Le réseau s'adapte ensuite pour reproduire les sorties souhaitées quand un exemple d'entrée est présenté.

L'enjeu, dont est actuellement l'objet de l'intelligence artificielle a renouvelé l'intérêt pour l'apprentissage.

En effet, à mesure que les techniques de représentation des connaissances se font de plus en plus nombreuses et efficaces, le principal goulot d'étranglement dans la réalisation d'un système basé sur la connaissance est précisément l'acquisition de cette connaissance. La construction d'une base de connaissance nécessite la mise en œuvre d'importants moyens humains et constitue la part la plus importante des efforts de conception d'un système expert.

Il existe aujourd'hui une panoplie de réseaux qui se distinguent par leur architecture et/ou par la loi d'apprentissage qui régit le réseau.

Dans ce chapitre nous présentons une synthèse des aspects descriptifs, algorithmes et des modèles neuronaux les plus connus. **[8]**

I.2.Génèse:

C'est en 1943 que, dans un article resté fameux, Mc Culloch et Pitt 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.

En 1949,D.O. Hebb, dans un livre resté lui aussi fameux, introduit la notion de « plasticité synaptique», c'est à dire le mécanisme de modification progressive des couplages entre neurones responsables de changements permanent de leurs propriétés collectives, ce qui l'on peut appeler «apprentissage». Son hypothèse consiste à la limitation de l'augmentation du coefficient de couplage entre deux neurones réels qui sont excités simultanément de plus cette hypothèse a été étendue aux neurones artificiels comme une règle par laquelle les couplages se modifient proportionnellement aux corrélations entre neurones, que ces corrélations soient positives (activation) ou négatives(inhibition).**[8]**

La première machine adaptative ayant connu un certain succès est le «perceptron» de Rosenblatt. Parallèlement, d'autres méthodes d'apprentissage étaient proposées, en particulier la célèbre procédure de Widrow et Hoff ou Adaline 1960, basée sur la minimisation itérative d'un critère quadratique.**[2]**

F. Rosenblatt a publié en 1958 (dans une revue de psychologie) un article sur un modèle probabiliste du stockage de l'information et de l'organisation dans le cerveau, intitulé «perceptron». Il a tenté de montrer qu'une machine (une rétine à réponse binaire, connectée à une couche constituée de neurones formels, elle-même connectée à une couche analogue dite «d'association») était capable de distinguer entre plusieurs stimulus, et même capable de grouper des stimulus ayant entre eux des relations de proximité, en les séparant d'autres groupes de stimulus. **[8]**

En ce sens, on parle des travaux mathématiques sur la classification de données, mais dans le cadre du connexionnisme et de l'auto-organisation.

Le renouveau actuel des réseaux de neurones est du à des contributions originales, comme celles d'Hopfield en 1982 qui a étudié la capacité de stockage et de restauration des informations «mémoire associative». Il est intéressant de noter que ce modèle ne levait pas les limitations du perceptron et de ses variantes. Malgré cela, le perceptron et les raisons de son échec semblaient oubliées.**[2]**

I.3. Modèle biologique du neurone:

L'élément fonctionnel de base du système nerveux est le neurone. D'une espèce animale à une autre, les neurones présentent des différences notoires; à l'intérieur d'une même espèce, on dénombre de nombreux types de neurones. Les différences portent tant sur des aspects anatomiques que fonctionnels. Cependant des point communs sont à la base de l'archétype de la cellule nerveuse que nous présentons ici. La figure *I.1(a)* représente l'anatomie d'un neurone.**[2]**

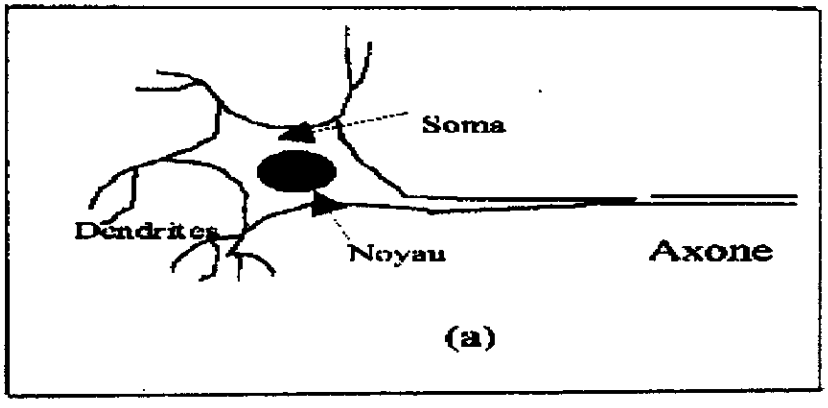
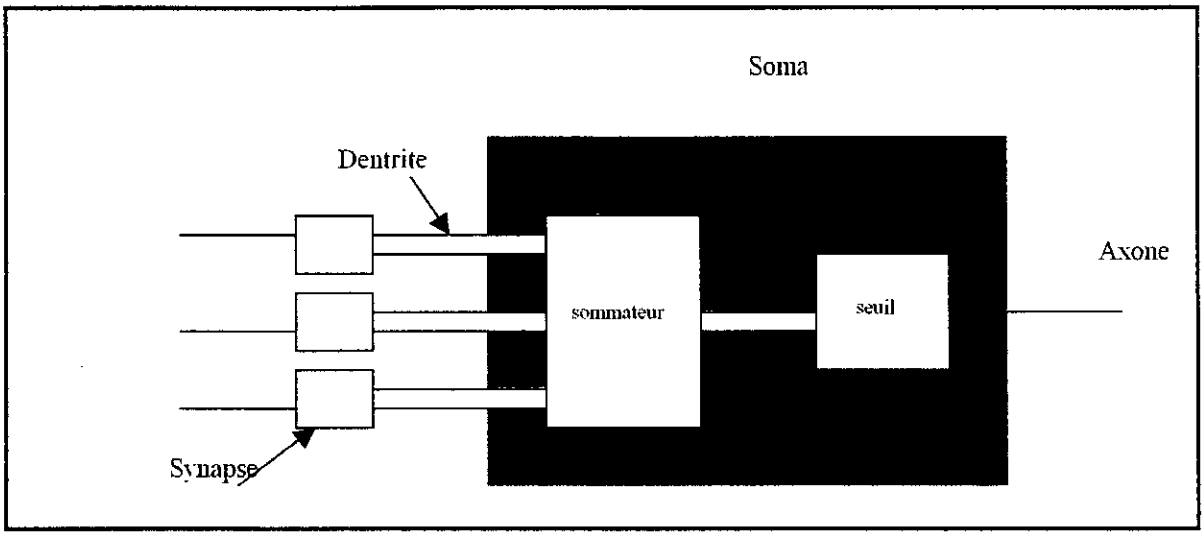


Figure (I.1(a)) : Anatomie d'un neurone



Figure(I.1(b)): Schéma simplifié du fonctionnement du neurone[2]

Un neurone est une cellule. Autour du noyau, on trouve le corps cellulaire (soma) celui-ci se prolonge par un axone unique et comporte de nombreuses dendrites qui constituent son organe «d'entrée». L'influx nerveux est assimilable à un signal électrique, se propageant dans les neurones de la manière suivante :

- les dendrites reçoivent l'influx nerveux d'autres neurones,

- le neurone évalue alors l'ensemble de la stimulation qu'il reçoit c'est à dire sa dépolarisation par rapport à l'extérieur ,

- En fonction de cette stimulation (si la dépolarisation est suffisante>-50mV), Le neurone transmet un signal de type tout ou rien le long de son axone. On dira alors que le neurone est excité ou inhibé,

- l'excitation du neurone est propagée le long de l'axone jusqu'aux autres ou fibres musculaires qui y sont connectés via les synapses. **[2]**

I.4. Modèle mathématique du neurone :

La première modélisation d'un neurone a été présenté par *Mac Culloch et Pitts* 1943. Ils ont modélisés le neurone par deux fonctions :

1. Une fonction de sommation qui élabore un potentiel P, égal à la somme pondérée des entrées *Xn* du neurone.

$$P=\sum_{n} W_n x_n \qquad (1.1)$$

Ou *Wn* représente les poids synaptiques du neurone.

2. Une fonction seuil ou d'activation, *f*, qui calcule l'état de sortie S du neurone en fonction de son potentiel :

$$S=f(p) \qquad (1.2)$$

La fonction d'activation peut être quelconque : fonction linaire, fonction sigmoïde, fonction seuil ou autres.

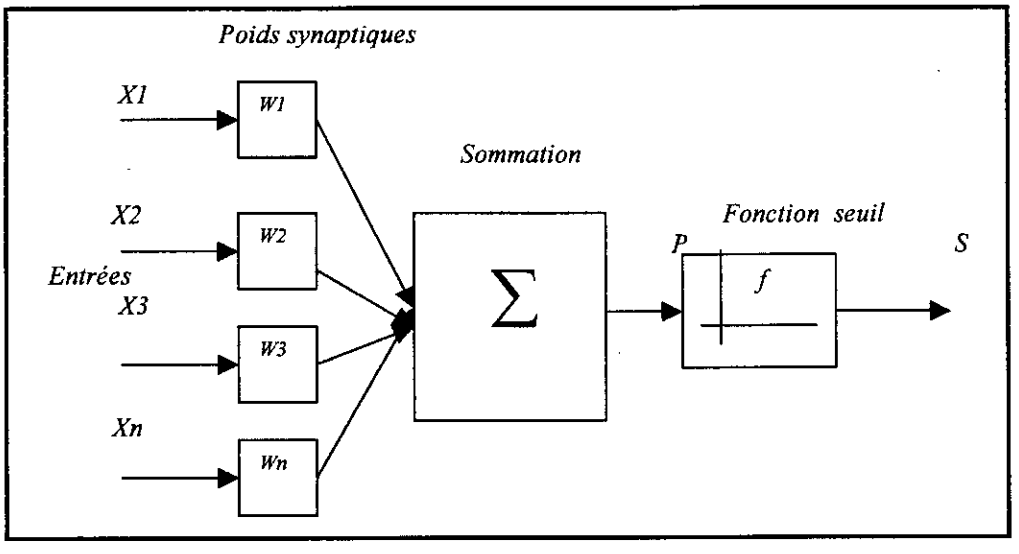


Figure (I.2) : Modèle mathématique du neurone

La figure ci-dessus présente le modèle mathématique du neurone.

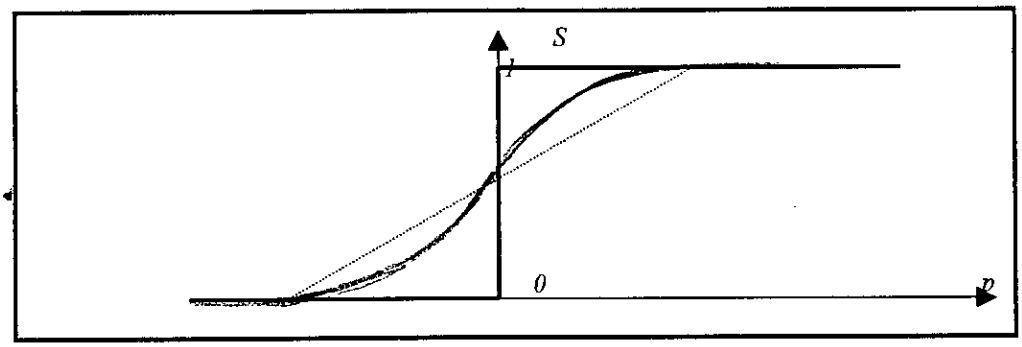


Figure (I.3) : Différentes formes de fonctions d'activation

I.5. Construction des réseaux de neurones :

Un réseau de neurone peut être représenté par un graphe direct composé d'un ensemble de nœuds ou éléments processeurs, fortement interconnectés par des liens orientés ou connexions. Les réseaux de neurones peuvent se distinguer par :

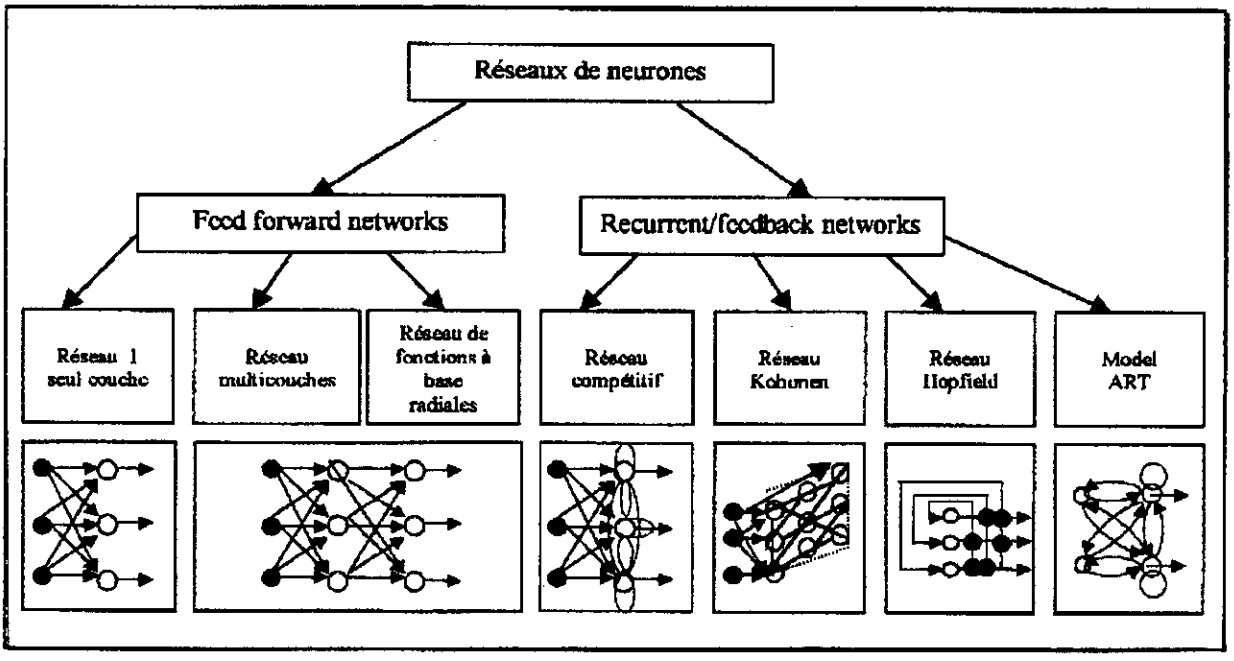
- l'architecture.
- le mode d'apprentissage.

I.5.1. Architecture des réseaux de neurones :

Du point de vue architectural *(figureI.4)*, les réseaux de neurones peuvent être regroupés en deux catégories :

- Les réseaux à circulation de l'information vers l'avant « feed forwad networks » dans lesquels les neurones sont organisés en couches. Dans cette catégorie, on distingue les réseaux possédant une seule couche (exp.perceptron) et les réseaux multicouches possédant une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées.

- les réseaux récurrents ou bouclés « recurrent / feedback networks »dans lesquels sont classés les réseaux compétitifs, le réseau de Kohonen, le réseau de Hopfield et les modèles ART (théorie de la résonance artificielle).



figure(I.4): Différentes architectures des réseaux de neurones [2]

I.5.2. Réseaux statiques et dynamiques :

On peut classer les réseaux en deux grandes catégories, selon la dépendance de l'évolution ceux-ci en fonction explicite du temps.

Dans le cadre des réseaux statiques, le temps n'est pas un paramètre significatif. En d'autres termes, la modification de l'entrée n'entraîne qu'une modification stable de la sortie, mais n'entraîne pas de retour d'information vers cette entrée.

Les réseaux dynamiques, comme leur nom l'indique, contient des rebouclages partiels ou totaux entre neurones, et on a donc une évolution dépendante du temps. Il faut bien distinguer la dépendance théorique, pour laquelle l'état du réseau à un certain instant dépend de son état à l'instant ou aux instants précédents, du temps nécessaire à obtenir une réponse, dans le cas d'une réalisation matérielle ou d'une simulation sur ordinateur, que le réseau soit statique ou dynamique. On peut ajouter que, formellement, le fonctionnement d'un réseau quelconque dépend de son histoire passée, par le biais de l'apprentissage. Le perceptron multicouche ordinaire ou la carte auto-organisatrice sont des réseaux statiques. Par contre, le réseau de Hopfield ou perceptron avec le rebouclage sont des réseaux dynamiques.**[5]**

I.5.3. Apprentissage des réseaux de neurones :

L'apprentissage permet au réseau de neurones de modifier sa structure interne (poids synaptiques)pour s'adapter à l'environnement de l'application.

Il existe trois modes d'apprentissage :

- Supervisé.
- Renforcé.
- Non supervisé.

I.5.3.1.L'apprentissage supervisé :

Dans cet apprentissage, un professeur qui connaît parfaitement la sortie désirée ou correcte, guide le réseau en lui apprenant à chaque étape le bon résultat. Donc l'apprentissage ici, consiste à comparer le résultat obtenu et le résultat désiré. C'est par exemple le cas du réseau multicouche à rétropropagation du gradient ou encore le cas des fonctions à base radiale.**[9]**

I.5.3.2.L'apprentissage renforcé :

Un professeur supposé présent, mais la réponse exacte n'est pas présentée au réseau. Cependant, on indique au réseau s'il a calculé la bonne réponse ou non. Le réseau doit donc utiliser cette information pour améliorer ses performances.

Typiquement, les poids synaptiques dont les unités possèdent la bonne réponse sont renforcés et ceux dont les unités possèdent la mauvaise réponse sont rejetées. c'est le cas par exemple des ART.

Dans ce cas, il est facile de comprendre que l'apprentissage est plus difficile, la difficulté majeure consistant pour le réseau à identifier les étapes du processus qui sont responsables de l'échec ou du succès(en anglais, credit assignment problem).

I.5.3.3.L'apprentissage non supervisé :

Le réseau modifie ses paramètres en tenant compte seulement des informations locales ou la Fourniture à un réseau autonome d'une quantité suffisante d'exemples contenant des corrélations (autrement dit de la redondance), ces réseaux n'ont pas besoin de sorties désirées préétablies. Ce sont souvent appelés auto-organisateurs ou encore à apprentissage compétitif.**[9]**

I.5.4.Réseau à une seule couche:

Il est basé sur la règle d'apprentissage de perceptron , mais avant de présenter cette règle on va présenter d'abord la règle de Hebb.

Règle d'apprentissage de Hebb (ou règle de corrélation) :

L'apprentissage de Hebb est le premier mécanisme d'évaluation des synapses, proposé par D.O.Hebb en 1949.une interprétation de cette règle pour les réseaux de neurones est la suivante :

- si deux neurones connectés entre eux sont activés en même temps, la connexion qui les relie doit être renforcée.
- Dans le cas contraire elle n'est pas modifiée.

Sa formulation est la suivante :

$$Wij(t+1)=Wij(t)+\eta yiyj \qquad (1.3)$$

Ou ,*Wij* est le poids de la connexion reliant les neurones i et j.

t: représente l'étape d'apprentissage.

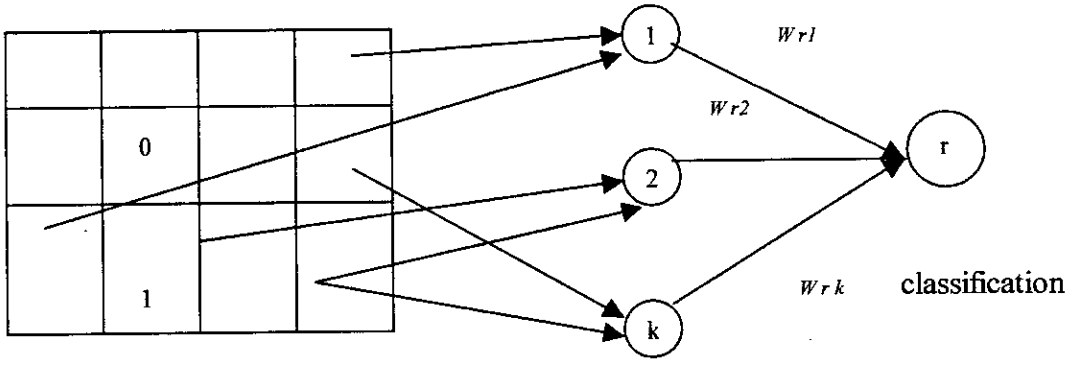
η: est un nombre compris entre 0 et 1, représentant le taux d'apprentissage.

yi, yj : sont les activations des neurones i et j respectivement,

On suppose que *yi, yj* sont booléens.(on prend les valeurs des activations dans *(0,1)*ou dans *(1,-1)).*

I.5.4.1. Le perceptron :

Le perceptron est le premier modèle opérationnel de réseaux de neurones. il est basé sur le neurone de Mc Culloch et Pitts et décrit un algorithme capable de modifier les poids en fonction des valeurs qu'on souhaite lui faire apprendre. Le terme perceptron couvre maintenant une classe de modèles beaucoup plus grande que le perceptron élémentaire d'origine, proposé par *Rosenblatt* en 1957.**[5]**



Figure(I.5) : Schéma d'un perceptron élémentaire avec k nœuds détecteurs de caractéristiques et un nœud de réponse r

Le perceptron est composé de trois types de cellules différentes.

- la première couche, appelée rétine, contient les cellules d'entrées. chaque cellule se contente de recopier la valeur qu'elle reçoit de l'extérieur.

- la seconde couche est composée de k cellules dites associatives ou détecteurs de caractéristiques. chaque cellule a des connexions entrantes pouvant parvenir de toutes ou d'une partie des cellules de la rétine avec connexion sortante reliée par un nœud réponse.

- La dernière couche, composée d'un seul nœud r à espace d'états binaire (1ou -1) est caractérisée par un seuil θr et des forces de connexions indiquées par les poids *Wri.*
  La décision du nœud-réponse r est prise suivant le seuillage de la somme pondérée des entrées .

Les perceptrons élémentaires sont des machines à règle de décision linéaire, partageant l'espace des vecteurs en deux classes séparées, par un hyperplan de l'espace à k dimensions. Ils ne sont donc capables d'apprendre que les classifications linéairement séparables.

L'apprentissage du perceptron est un apprentissage supervisé qui se fait par correction d'erreur. la fonction d'activation est une fonction discrète. Les sorties prennent des valeurs binaires(0ou1). La règle d'apprentissage du perceptron est la suivante :

si la sortie actuelle égale à 0 et doit être égale à 1 alors :

$$Wij(t+1)=Wij(t)+\eta xi$$

si la sortie actuelle est égale à 1 et doit être égale à 0 alors : ***(I.4)***

$$Wij(t+1)=Wij(t)-\eta xi$$

si la sortie est correcte alors :

$$Wij(t+1)=Wij(t)$$

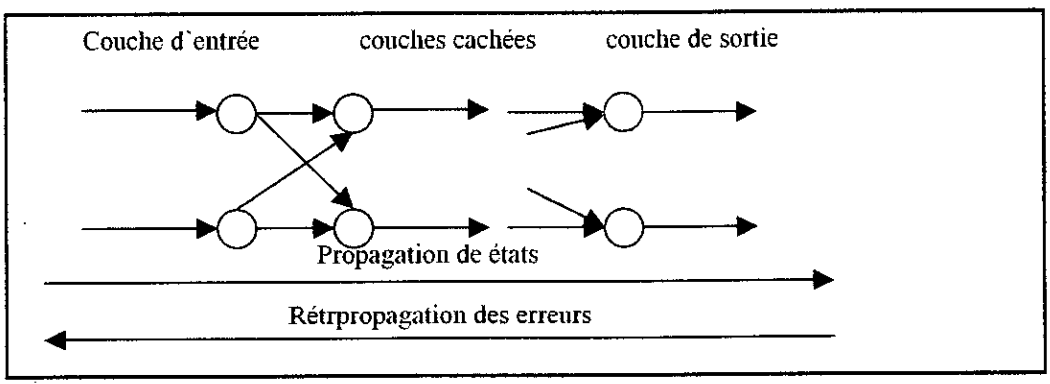
Ou xi : l'entrée du neurone i.

η : représente le coefficient d'apprentissage.

Wij : les poids synaptiques de connexion entre neurone i et le neurone j.

I.5.5.Réseaux multicouches:

Son modèle est suivant la figure(I.6):



Figure(I.6): Réseau multicouche [2]

I.5.5.1. la Règle de Widrow-Hoff ou règle delta :

Nous avons vu, que le perceptron est limité à des sorties binaires, Widrow et Hoff ont étudiés l'algorithme d'apprentissage du perceptron en considérant une fonction d'activation contenue et différentielle. Cette règle est aussi connue sous le nom de la méthode des

moindres carrées ou en anglais 'least mean square'(LMS)ou encore règle delta.. Le principe est le suivant :

- calculer l'erreur quadratique selon la formule :

$$E=\sum_{j=1}^{n}(dj-yj)^2 \qquad (1.5)$$

avec $$yj=\sum_{i=1}^{m} xiWij \qquad (1.6)$$

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

$$Wij(t+1)=Wij(t)+\eta(dj-yj)xi \qquad (1.7)$$

Ou,

n : le nombre de neurones à la sortie.

dj :est la sortie désirée.

yj :est la sortie calculée.

xi : l'entrée i du neurone j.

m :le nombre de neurone à l'entrée.

η : coefficient d'apprentissage.

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

I.5.5.2.La Règle delta généralisée :

La règle delta généralisée ou encore appelée règle de la rétropropagation du gradient est une généralisation de la règle de Widrow Hoff à un réseau multicouche utilisant des fonctions d'activation différentiables et un apprentissage supervisé. Nous verrons en détail (chapitre*II*) la formulation de cette règle. **[2]**

I.5.6.Réseaux compétitif:

L'apprentissage compétitif se fait par coopération et compétition entre les neurones. Les algorithmes d'apprentissages compétitifs tentent de minimiser une fonction de distorsion entre un vecteur d'entrée *Xi* et un vecteur de poids *Wi*. La règle de modification des poids synaptiques est donnée par l'équation suivante :

$$Wij(t+1) = Wij(t) + \eta(xi - \overset{+}{Wij}) \qquad (1.8)$$

Wij :le poids synaptique entre le neurone *i* et le neurone *j*.

η :le coefficient d'apprentissage .

t :le temps d'apprentissage.

$\overset{+}{Wij}$: le poids synaptique du neurone gagnant.

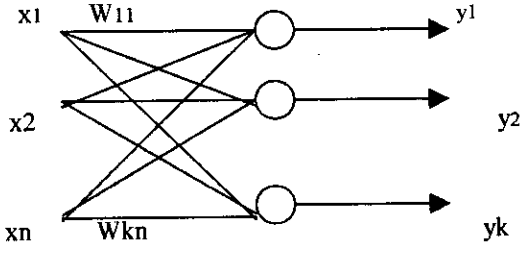
I.5.7. Réseaux Kohonen:

Une façon particulière de concevoir la classification et la reconnaissance de formes consiste à considérer le stimulus à reconnaître comme un vecteur dans l'espace de la perception.

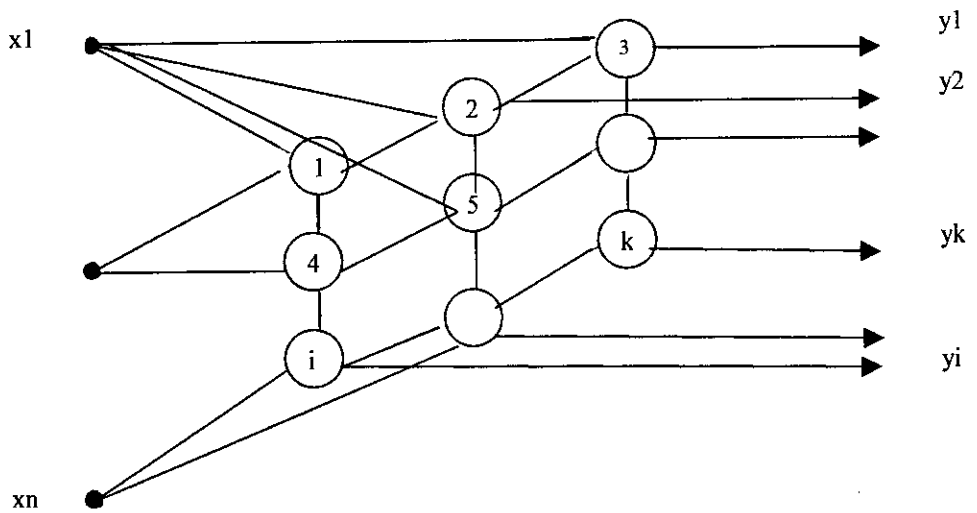
Les prototypes qui constituent les connaissances préalables sont aussi, des vecteurs étiquetés dans cet espace, et la classification se ramène à un problème topologique(ou géométrique , si l'on préfère): trouver le ou les prototypes les plus voisins du stimulus à identifier. On se trouve dans des cas ou la notion de réseaux de neurones formels commence à perdre son sens.**[8]**

Ce modèle de réseau a été présenté par KOHONEN en 1982 en se basant sur des constations biologiques. l'idée a donc été d'essayer de projeter des stimulus voisins, situés dans un espace perceptif multidimensionnel, dans un espace géométrique beaucoup plus simple, mono ou plus fréquemment bidimensionnel, en conservant au mieux la topologie. il est alors littéralement possible de voir des relations de voisinage, et d'en tirer des méthodes de classifications. Nous allons décrire sommairement sa méthode dite cartes auto-organisatrices.**[3]**

Les cartes organisatrices de KOHONEN sont réalisées à partir d'un réseau de k neurones à n entrées. le réseau possède ainsi k sorties. Ces neurones sont connectés suivant une structure mono ou bidimensionelle. Chaque neurone possède donc des voisins dans cette structure(figure(I.7). Les entrées sont des vecteurs à n composantes ;toutes totalement connectées aux k neurones du réseau par n×k connexions modifiables. **[3]**



(a)



(b)

Figure(I.7): Réseau auto-organisatrice de kohonen.

a) Structure mono-dimentionnelle b) Structure bi-dimensionnelle [3]

On suppose que les poids des connexions modifiables sont initialement aléatoire. Considérons un vecteur d'entrées x. la sortie du réseau est le vecteur y à k composantes :

$$yi = \sum_{j=1}^{n} wij.xj \qquad (1.9)$$

On note Wi les vecteur poids du neurone i, c'est à dire, le vecteur à n composantes $\{Wi1, Wi2, ..., Win\}$. Selon le vecteur x et la configuration initiale des poids, il existe un neurone i0 dont la sortie est la plus grande. Ce neurone sera appelé neurone gagnant.

On considère que dans un voisinage de i_0, les neurones i sont actifs(yi=1), et qu'ailleurs ils sont inactifs(yi=0)

La règle d'adaptation des poids est :

$$\Delta Wi = \eta(t)\Delta(i,i0)(x - Wi) \qquad (1.10)$$

$\eta(t)$:est le coefficient d'apprentissage.

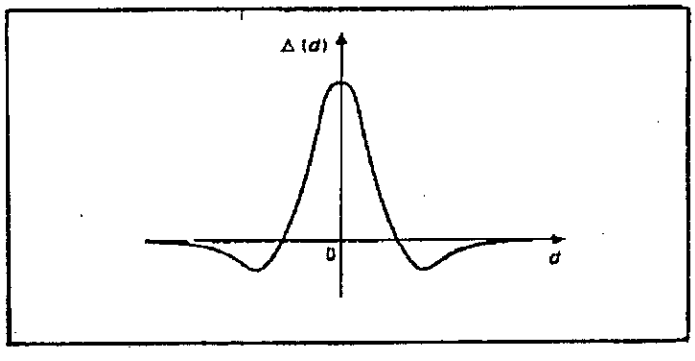
Pour obtenir une convergence presque sure, le coefficient d'apprentissage $\eta(t)$ est pris variable avec le temps. Son évolution est en 1/t.

$(i,i0)$est la fonction de voisinage .elle est maximale pour le neurone gagnant i0,et décroît plus ou moins rapidement sur les neurones voisins, devient négative au-delà, pour s'annuler loin du neurone gagnant. Cette fonction d'inhibition latérale, qui a la forme représentée sur la figure*(I.8)* ,est dite « en chapeau mexicain ». mais on peut par simplification la représenter par une fonction constante sur une certaine distance , nulle au-delà.**[3]**

Sa largeur est un paramètre d'apprentissage : grande au début ceci afin d'ordonner rapidement le réseau, puis de plus en plus faible par la suite, elle fixe la carte auto-organisatrice dans sa configuration finale. dans le cas le plus simple on utilise :

$$\Delta(i,i0) = \delta iio \qquad (1.11)$$

Ou δiio est une impulsion de Kronecker



Figure(I.8) :Fonction d'inhibition latérale

Le résultat se résume en 7 étapes :

1. choix du nombre k de neurones et de la structure du réseau.
2. Initialisation du voisinage et du pas d'adaptation $\eta(t)$. initialisation aléatoire des poids.
3. Prélèvement d'une entrée x dans la base d'apprentissage.
4. Recherche du neurone i_0 'le plus proche'(le gagnant).
5. Modification des poids *Wi* pour i tel que $\Delta(i,i_0)\neq 0$($i\in$voisinage de i_0).
6. Modification du voisinage$\Delta(i,i_0)$ et du pas d'adaptation $\eta(t)$.
7. Retour à l'étape 3 sauf si le test d'arrêt (le nombre d'itérations) est vérifié.

Le résultat de l'apprentissage est que chaque neurone soit représentatif d'un certain type d'entrées, qu'en moyenne, *Wi* tend vers *x*. or lorsque le voisinage se réduit le poids *Wi* n'est mis à jour que pour les vecteurs *x* qui sont les plus proches de *Wi* . si on représente l'espace d'entrée en domaine *Ri* ,alors le vecteur poids *Wi* du neurone i est le barycentre des vecteurs *x* du domaine *Ri* que ce neurone représente.**[3]**

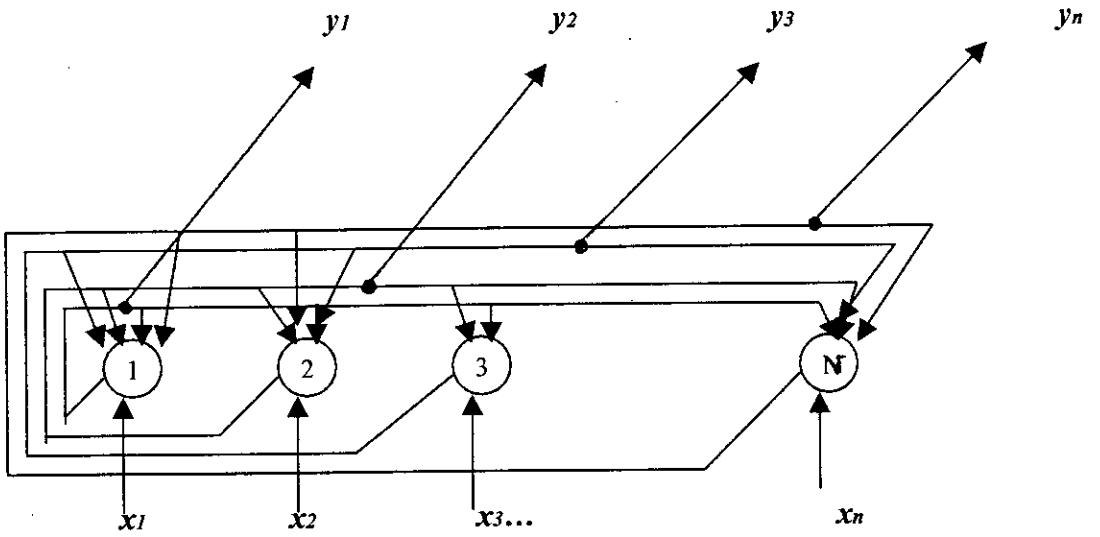
I.5.8.Réseaux Hopfield :

C'est une architecture simple qui possède certains fonctionnalités de type cognitif (mémoire associative).

La mémoire associative est un dispositif capable de mémoriser des informations que l'on peut ensuite retrouver, non par leur adresse comme dans une mémoire classique, mais en fournissant des données mêmes incomplètes ou bruitées relatives aux informations stockées : Ce type de mémoire s'appelle aussi Mémoire Adressable par le Contenu (CAM).

Dans le modèle de Hopfield, un neurone est décrit par une variable y, qui est la sortie du neurone, qui peut prendre deux états –1 ou +1 .l'état d'un réseau de N neurones(figure)est donc décrit par un vecteur binaire $y\in\{-1,+1\}$ tous les neurones sont connectés entre eux.

Il en résulte qu'il y a *N* connexions portant des poids. Les interconnexions entre les neurones sont symétriques.**[3]**



Figure(I.9): Modèle de Hopfield

Les influences mutuelles des neurones sont contenues dans la matrice des poids *W*. Comme les interconnexions entre les mesures sont symétriques, la matrice *W* est symétrique :

$$Wij = Wji \qquad (1.12)$$

L'apprentissage consiste à calculer les coefficients synaptiques *Wij* connaissant L vecteurs *x1,x2,...,xL* de dimensions *N* dont on veut faire des états stables du réseau. On appelle ces vecteurs les prototypes appris par le réseau. La matrice des poids est donnée par la relation suivante :

$$W = \sum_{i=1}^{L} xi \,.xi^{T} \qquad (1.13)$$

On procède de la manière suivante pour faire évoluer y à l'instant *(t)* du réseau :

- on choisit un neurone *k*
- on calcule le potentiel $pk(t)$ du neurone *k*.

$$pk(t) = \sum_{j=1}^{N} Wij \,.yj(t) \qquad (1.14)$$

- l'état du neurone k est alors mis à jour selon le signe du potentiel synaptique :

$$yk\ (t+1) = 1\, ou\ -1 \qquad (1.15)$$

- on choisit une nouvelle valeur de *k* et on recommence.

Au cours de ces itérations, les états des neurones finissent par se stabiliser à des valeurs qui dépendent des états initiaux et de la matrice des poids *W*.

Un neurone est dit stable lorsque : $yj(t+1)=yj(t)$.

L'utilisation du réseau, après apprentissage, consiste à imposer à chaque neurone une composante d'un vecteur stimulus, et à laisser le réseau évoluer librement(relaxation)sous l'effet de sa dynamique. La règle d'apprentissage garantit d'atteindre un état stable(attracteur) après un nombre fini d'étapes. si l'on mesure la distance de Hamming entre cet attracteur et chacun des prototypes appris, on constate que généralement l'attracteur est très voisin ou confondu avec un des prototypes : le réseau se comporte donc comme une mémoire auto-associative, restituant une information apprise sous l'effet d'une stimulation .

On remarque cependant l'apparition quelquefois d'attracteurs ne correspondant à aucun prototype : on les appelles les états parasites. Cela se produit si l'état initial $y=\{y1,y2,...,yN\}$ du réseau est très éloigné d'un état mémorisé, ou si un grand nombre de prototypes ont été appris. Cependant ce réseau ne permet pas de mémoriser un trop grand nombre d'exemples soit 0.14 fois le nombre de neurones de ce dernier.

Le réseau de Hopfield peut présenter un comportement indésirable : l'oubli catastrophique. ainsi, un réseau peut apprendre K exemples sans problèmes, et les oublier tous dés lors qu'on souhaite lui en apprendre un de plus.

I.6. Propriétés des réseaux de neurones :

- **La capacité d'apprentissage :**la capacité d'apprentissage ou d'adaptation se traduit par la capacité du réseau de neurones à apprendre à résoudre des problèmes à partir d'exemples de façon similaire à l'être humain ou à l'animal.**[5]**

- **La capacité de généralisation**

Elle se traduit par la capacité d'un système à apprendre et à retrouver, à partir d'un ensemble d'exemples des règles qui permettent de résoudre un problème donné non appris.**[5]**

- **Le parallélisme**

Cette notion se situe à la base de l'architecture des réseaux de neurones considérés comme un ensemble d'entités élémentaires qui travaillent simultanément. Le parallélisme permet une rapidité de calcul supérieure mais exige de penser et de poser les problèmes différemment.**[5]**

I.7.Domaines d'application des réseaux de neurones :

Les réseaux de neurones sont bien adaptés à la résolution des problèmes ayant les caractéristiques suivantes :

1. les règles qui permettent de résoudre le problème sont inconnues ou très difficiles à expliciter.
2. Le problème fait intervenir des données bruitées.
3. Le problème nécessite une grande rapidité de traitement.

Les domaines d'applications suivants possèdent pratiquement toutes les caractéristiques exposées précédemment. Ils constituent les principales applications des réseaux de neurones :

- Reconnaissance de formes
- Traitement du signal
- Vision, parole
- Robotique
- Prévision et modélisation **[3]**

I.8. Conclusion :

L'étude des modèles des réseaux de neurones nous a permis de dégager les principales caractéristiques qui les différencient des méthodes classiques utilisées dans la résolution des problèmes posés par plusieurs domaines :l'intelligence artificielle, le traitement du signal, la reconnaissance de formes. Ces caractéristiques sont :

- leur parallélisme intrinsèque
- leur capacité d'adaptation (capacité d'apprentissage)
- la mémoire distribuée : la mémorisation de la fonction à réaliser par le réseau est distribuée sur plusieurs neurones, ce qui introduit une résistance au bruit car la perte d'un neurone ne correspond pas à la perte de la fonction mémorisée.
- Leur capacité de généralisation.

Le développement des domaines d'applications des réseaux de neurones a conduit les chercheurs à passer de la simulation numérique de ces réseaux à leur implémentation sur des structures matérielles diverses en vue d'une utilisation effective afin de profiter au mieux de leurs avantages.

Le modèle multicouche est le plus utilisé parmi les différents modèles de réseaux car il permet d'approximer n'importe qu'elle fonction de transfert donnée, après un choix approprié de ses paramètres. De ce fait c'est le modèle le plus intéressant à implémenter.

CHAPITRE II
ALGORITHME DE RETROPROPAGATION DU GRADIENT

II.1.Introduction :

L'incapacité des réseaux mono-couches (perceptron) à résoudre des problèmes simples, notamment la classification de données en plusieurs classes, a entraîné un ralentissement de la recherche sur les réseaux de neurones. Ce n'est que dans le courant des années 70,que des travaux démontrèrent qu'en ajoutant une ou plusieurs couches entre la couche de neurones d'entrée et la couche de sortie, on est certain de pouvoir approximer à la sortie du réseau n'importe qu'elle fonction de l'espace d'entrée. mai la théorie ne donne aucune indication sur le nombre et la taille de ces couches intermédiaires (appelées couche cachées). Ce nombre devant être d'autant plus grand que la fonction à implémenter est irrégulière.

Dans la quasi-totalité des cas, on constate qu'une couche cachée est suffisante.

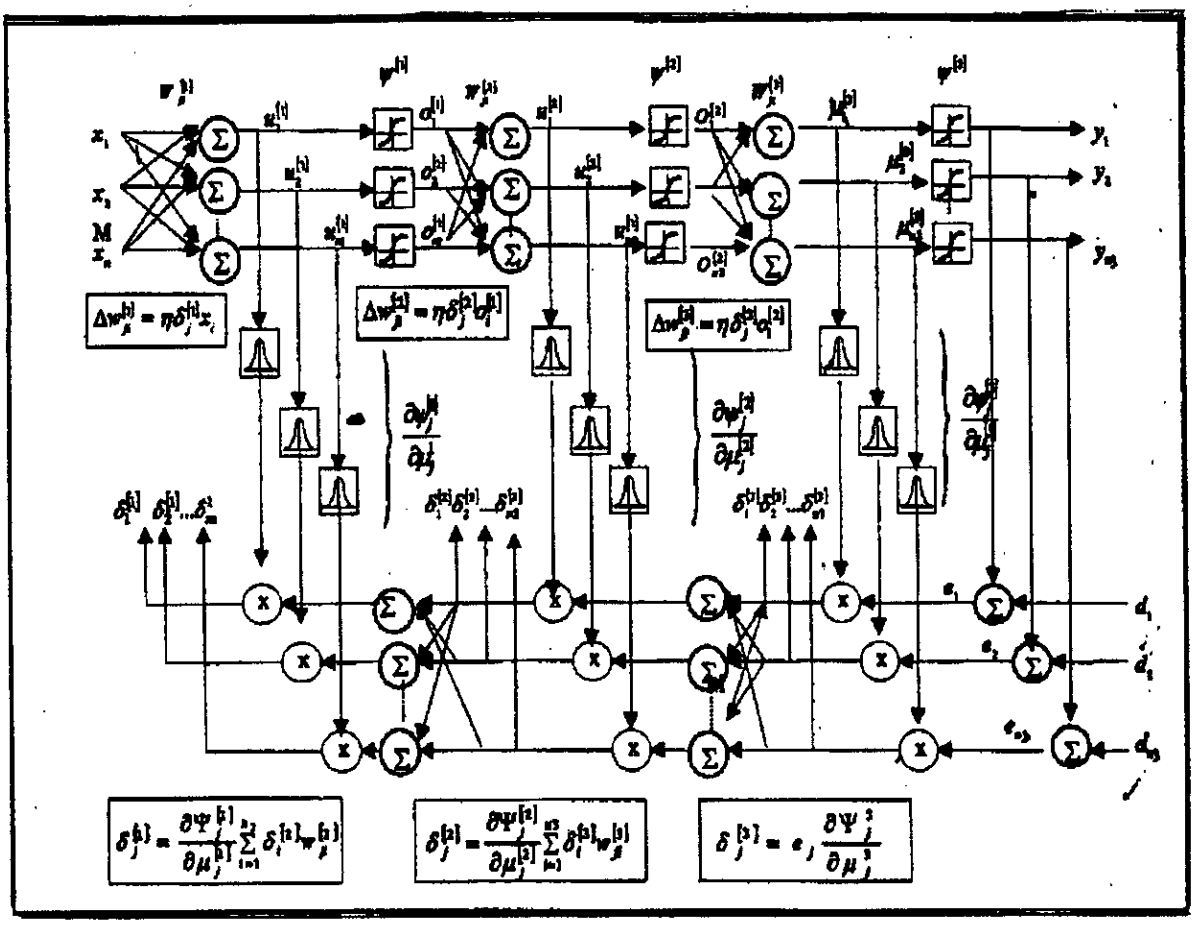
L'apprentissage d'un réseau multicouche ne peut pas être réalisé par la règle de WIDROW-HOFF. En effet, si cette méthode est directement applicable pour ajuster les poids de la dernière couche, elle ne l'est pas pour ceux des couches internes, car on ne connaît pas la sortie désirée pour ces couches et par conséquent , on ne connaît pas directement le terme d'erreur associé à chaque couche interne.

La solution à ce problème a été apporté par l'algorithme de 'rétropropagation du gradient d'erreur', appelé «backpropagation algorithm».**[3]**

II.2. La perceptron multicouche :

On considère que le perceptron multicouche (MLP)est constitué de trois couches de neurones, la première couche avec n_0 entrées et n_1 neurones, la seconde avec n_2 neurones et la sortie avec n_3 neurones.

La figure *(II.1)* montre une représentation fonctionnelle de l'algorithme de rétropropagation de gradient pour le réseau de neurone(MLP).**[2]**



Figure(II.1): Représentation fonctionnelle de l'algorithme (RPG) [2]

II.3. L'Algorithme :

L'algorithme de rétropropagation du gradient est utilisé dans le perceptron multicouche. La phase importante ou ce dernier est utilisé est bien évidemment l'apprentissage : on présente au réseau des entrées et on lui demande de modifier sa pondération de telle sorte que l'on retrouve la sortie correspondante . L'algorithme consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau. La seconde étape compare la sortie calculée à la sortie désirée. On modifie alors les poids de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée et la sortie désirée soit minimisée.**[2]**

On répète ce processus sur tous les exemples jusqu'à l'obtention d'une erreur de sortie considérée comme négligeable.

L'algorithme da la rétropropagation standard, utilise la méthode de plus forte descente pour la minimisation de la moyenne carrée de l'erreur locale, qui est donné par

$$E_p = \frac{1}{2}\sum_{j=1}^{n3}(d_{jp} - y_{jp})^2 = \frac{1}{2}\sum_{j=1}^{n3} e^2{}_{jp} \qquad (2.1)$$

et l'erreur globale donnée par :

$$E = \sum_p E_P = \frac{1}{2}\sum_p \sum_j (d_{jp} - y_{jp})^2 \qquad (2.2)$$

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

Le problème de l'apprentissage peut être formulé comme suit : étant donné l'état actuel des poids synaptiques, on doit déterminer l'augmentation ou la diminution ΔW^s_{ji} (s étant le numéro de la couche :s=1,2,3) des poids afin de minimiser l'erreur globale, E.

ΔW^s_{ji} s'écrit :

$$\Delta W^s_{ji} = -\eta \frac{\partial E_P}{\partial W^s_{ji}} \qquad (2.3)$$

η est le coefficient d'apprentissage.

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

Pour la couche de sortie(s=3)

$$\Delta W_{ji}^{3} = -\eta \frac{\partial E_P}{\partial W_{ji}^{3}} = -\eta \frac{\partial E_P}{\partial \mu_j^3} \cdot \frac{\partial \mu_j^3}{\partial W_{ji}^{3}} \qquad (2.4)$$

Avec $$\mu_j^3 = \sum_{i=1}^{n2} W_{ji}^3 x_i^3 = \sum_{i=1}^{n2} W_{ij}^3 O_i^2 \qquad (2.5)$$

$$j = 1,..n3$$

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

$$\delta_j^3 = -\frac{\partial E_p}{\partial \mu_j^3} = -\frac{\partial E_p}{\partial e_j} \cdot \frac{\partial e_j}{\partial \mu_j^3} \qquad (2.6)$$

ou

$$\frac{\partial E_p}{\partial e_{jp}} = e_{jp} \qquad (2.7)$$

$$\frac{\partial e_{jp}}{\partial \mu_j^3} = \frac{\partial (d_{jp} - y_{jp})}{\partial \mu_j^3} = -\frac{\partial y_{jp}}{\partial \mu_j^3} = \frac{\partial \psi_j^3}{\partial \mu_j^3} \qquad (2.8)$$

Donc
$$\delta_j^3 = e_j^p \cdot \frac{\partial \psi_j^3}{\partial \mu_j^3} \qquad (2.9)$$

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

$$\Delta W_{ij}^3 = \eta \delta_j^3 x_i^3 = \eta \delta_j^3 O_i^2 \qquad (2.10)$$

Ou,
$$\delta_j^3 = e_{jp}(\psi_j^3)' = (d_{jp} - y_{jp}) \frac{\partial \psi_j^3}{\partial \mu_j^3} \qquad (2.11)$$

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

$$\Delta W_{ji}^2 = -\eta \frac{\partial E_P}{\partial W_{ji}^2} = -\eta \frac{\partial E_P}{\partial \mu_j^2} \cdot \frac{\partial \mu_j^2}{\partial \mu_{ji}^2} \qquad (2.12)$$

$$= \eta . \delta_j^2 x_i^2 = \eta \delta_j^2 O_j^1$$

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

$$\delta_j^2 = -\frac{\partial E_P}{\partial \mu_J^2} = -\frac{\partial E_p}{\partial O_j^2} \cdot \frac{\partial O_j^2}{\partial \mu_j^2} \qquad (2.13)$$

avec $$\mu_j^2 = \sum_{i=1}^{n1} W_{ij}^2 x_i^2 = \sum_{i=1}^{n1} W_{ji}^2 O_i^1 \qquad (2.14)$$

$$j = 1,...n2$$

sachant que

$$O_j^2 = \psi_j^2(\mu_j^2) \qquad (2.15)$$

on obtient

$$\delta_j^2 = -\frac{\partial E_P}{\partial O_j^2} \cdot \frac{\partial \psi^2}{\partial \mu_j^2} \qquad (2.16)$$

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

$$\frac{-\partial E_p}{\partial O_j^2} = -\sum_{i=1}^{n3} \frac{\partial E_p}{\partial \mu_j^3} \cdot \frac{\partial \mu_j^3}{\partial O_i^2} = \sum_{i=1}^{n3} (-\frac{\partial E_p}{\partial \mu_j^3}) \frac{\partial}{\partial O_j^2} [\sum_{i=1}^{n3} W_{ij}^3 x_i^3] = [\sum_{i=1}^{n3} \delta_i^3 \frac{\partial}{\partial O_i^2} [W_{ij}^3 O_i^2]$$

$$= \sum_{i=1}^{n3} \delta_i^3 W_{ji}^3 \qquad (2.17)$$

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

$$\delta_j^2 = \frac{\partial \psi_j^2}{\partial \mu_j^2} \sum_{i=1}^{n3} \delta_i^3 W_{ji}^3 \qquad (2.18)$$

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

$$\Delta W_{ij}^1 = \eta \delta_j^1 x_i^1 = \eta \delta_j^1 O_i^0 = \eta \delta_j^1 x_i \qquad (2.19)$$

Ou l'erreur locale est déterminée par :

$$\delta_j^1 = \frac{\partial \psi_j^1}{\partial \nu_j^1} \sum_{i=1}^{n2} \delta_j^2 W_{ij}^2 \qquad (2.20)$$

$$\mu_j^1 = \sum_{i=1}^{n0} W_{ji}^1 x_i \qquad (2.21)$$

$$j = 1,...n1$$

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

1. initialiser les poids synaptiques W_{ji}^s à des valeurs aléatoires(généralement entre –0.5/fan-in et +0.5/fan-in, ou le fan-in représente le nombre de neurones contenus dans la couche directement inférieur).

2. Présenter un vecteur d'entrée et la sortie correspondante (désirée) et calculer les sorties actuelles de tous les neurones en utilisant les valeurs de W_{ji}^s .

3. Spécifier la sortie désirée et évaluer les erreurs locales δ_j^s pour toutes les couches .

4. Ajuster les poids synaptiques en accord avec la formule itérative

$$\Delta W_{ij}^s = \eta \delta_j^s x_i^s \qquad \text{(s=3,2,1)}$$

5. Calculer l'erreur E_p en utilisant l'équation (2.1)
6. Répéter les étapes 2 à 5 avec le même vecteur d'entrée jusqu'à ce que l'erreur E_p soit minime.

7. Répéter 2 à 6 pour chaque vecteur d'entrée x (pour chaque exemple d'apprentissage).

Tous les exemples d'apprentissage sont présentés périodiquement jusqu'à ce que les poids synaptiques soient stabilisés, i.e. jusqu'à ce que l'erreur pour tout l'ensemble complet soit acceptable et le réseau converge.

En pratique, une grande valeur du coefficient d'apprentissage η est préférable car elle entraîne une convergence rapide. Malheureusement, une grande valeur de η peut aboutir à une oscillation ou même à une divergence. Pour surmonter ce problème, un terme momentum est introduit dans les équations de mise à jour des poids synaptiques, tel que :

$$W_{ji}^{s}(t+1) = W_{ji}^{s}(t) + \Delta W_{ji}^{s}(t)$$

$$\Delta W_{ji}^{s}(t) = \eta \delta_{j}^{s} O_{i}^{s-1} + \alpha \Delta W_{ji}^{s}(t-1)$$

avec $\eta > 0$, $0 \leq \alpha \leq 1$

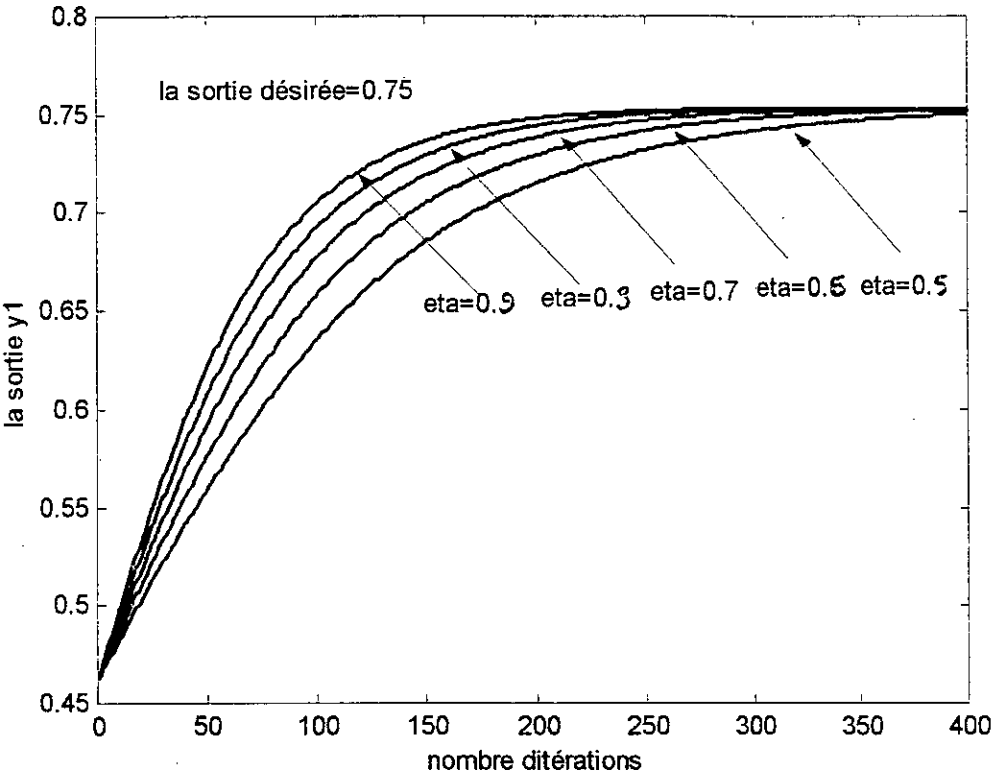
s=1,2,3.

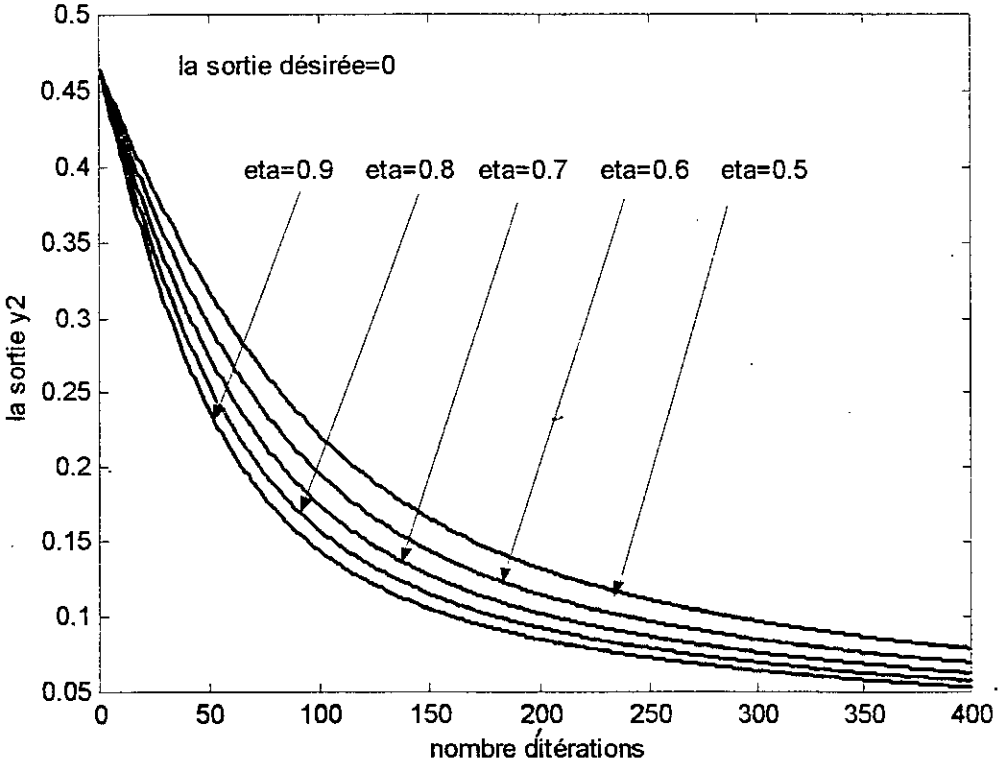
II.4.Simulation:

Si on prend un exemple d'un réseau multicouche tel que :

- couche d'entrée : 64 nœuds (neurones).
- Couche cachée : 12 nœuds .
- Couche de sortie : 2 nœuds.

On a programmé ce réseau par MATLAB pour étudier la convergence de la sortie et de l'erreur de chaque nœud de la couche de sortie. On a obtenu les courbes suivantes :


la sortie désirée=0.75
eta=0.9
eta=0.3
eta=0.7
eta=0.6
eta=0.5
la sortie y1
nombre ditérations


la sortie désirée=0
eta=0.9
eta=0.8
eta=0.7
eta=0.6
eta=0.5
la sortie y2
nombre ditérations

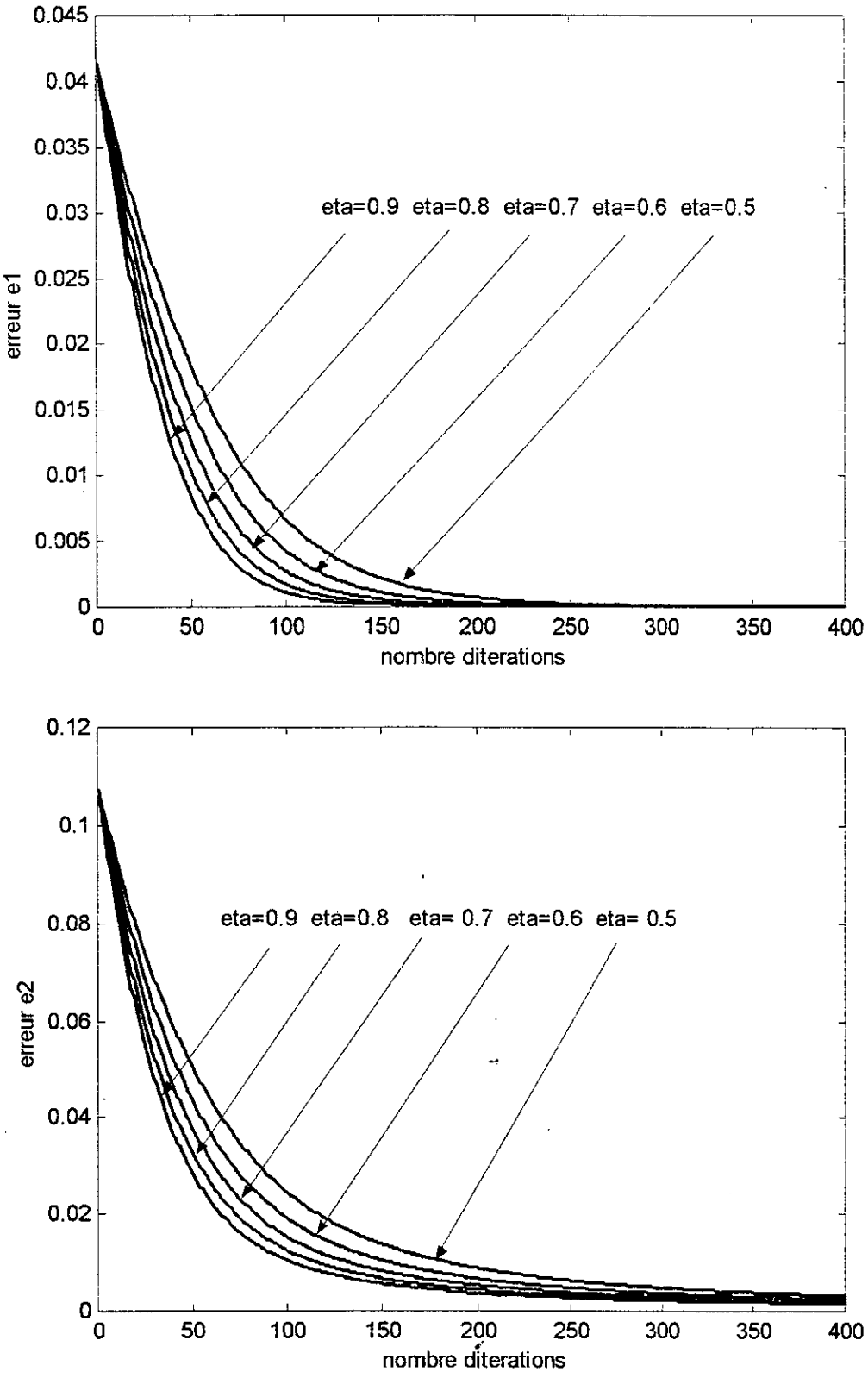


Figure (II.2): Evolution des erreurs et des sorties en fonction du coefficient d'apprentissage

II.5.Conclusion :

Dans ce chapitre nous avons présenté un algorithme d'apprentissage (Algorithme de Rétropropagation du gradient) le plus utilisé et le plus pratique pour les réseaux multicouches.

L'algorithme du Rétropropagation du gradient (RPG) se présente en deux étapes essentielles :

Première étape : propagation vers l'avant des entrées jusqu'à obtenir une sortie calculée par le réseau.

Deuxième étape : comparaison entre le sortie calculée et la sortie désirée, on modifie par la suite les poids de telle sorte qu'à la prochaine itération , l'erreur calculée soit minimisée .

Une fois que l'algorithme converge ,c'est à dire la sortie converge vers une valeur désirée, on prend par la suite les valeurs des synaptiques w, pour les stocker dans des mémoires, dans la phase de reconnaissance.

CHAPITRE III
LA TECHNOLOGIE FPGA

III.1. Introduction:

Il y a deux aspects importants dans la conception architecturale des circuits VLSI, l'une liée à la complexité fonctionnelle de chaque processus élémentaire (PE), l'autre à la structure d'interconnexion du réseau, ainsi que deux aspects correspondant à l'architecture des RNAs, qui sont respectivement les unités de traitement et la forme de connectivité des poids.

L'intérêt principal dans un algorithme orienté vers la conception d'un processeur de réseau est :

étant donné un algorithme, comment un processeur réseau est systématiquement dérivé. En se basant sur cette méthodologie systématique d'implémentation, quel est le type de conception spécialement facile à dériver.

Les implémentations analogiques et digitales des réseaux de neurones constituent une partie de recherche dans ce domaine.**[5]**

III.2. En vue d'implémentation digitale des RNAs :

Il faut tenir compte des points suivants dans l'implémentation digitales des RNAs :

1- l'analyse du problème orienté à une application et la modélisation des caractéristiques du réseau peuvent être faites indépendamment de la conception du circuit,

2- par simulation on peut augmenter le précision de calcul (longueur du mot) pour chaque bloc de fonction avant de passer à son implémentation. La technologie et l'architecture déterminent seulement combien de neurones peuvent être intégrés dans le chip et à quel moment ils sont activés.

3- c'est la tâche du concepteur d'implémenter les algorithmes selon l'état de l'art dans le traitement du signal digital .

4 - en utilisant la logique digitale et les mémoires, il est plus facile de partitionner un grand problème afin d'être résolus par une petite implémentation (en terme matériel).

En utilisant la même logique et des mémoires, une implémentation digitale peut réaliser plus d'un réseau et combine les résultats d'une façon hiérarchique pour résoudre un grand nombre de problèmes.**[5],[2]**

III.3. Méthodologie de conception sur un FPGA :

III.3.1. Introduction :

Des projets nécessitant rapidité et des durées de conception plus courtes, des fonctionnalités de plus en plus nombreuses et complexes, tels sont les principaux défis auxquels sont confrontés aujourd'hui les concepteurs électroniciens, dans un environnement ou technologies et innovation amplifient les besoins. **[14]**

III.3.2. Styles de conception :

Du concept ASIC découle quatre grands styles de conception(Design Style) :
les circuits sur mesure (FULL-Custom),
les circuits précaractérisés (Standard Cells),
les circuits Prédiffusés (Gate Array),
les réseaux logiques programmables (FPGA).

A chaque famille sont associés un niveau de complexité, un type de conception et une méthodologie de conception.

III.3.2.1. Les circuits sur mesure « FULL CUSTOM » :

Les circuits intégrés à la demande (FULL CUSTOM) sont historiquement les premiers apparus. Chaque élément actif est dessiné sur mesure, avec pour objectif d'optimiser ses performances électriques et de réduire la surface utile de silicium. Les coûts d'études sont ici très élevés et ne peuvent être amortis que par une pondération en volume importante. Comme pour les circuits précaratérisés, toutes les étapes technologiques du processus de fabrication sont nécessaires à la réalisation du composant. L'optimisation en surface permet des prix de production unitaire faibles, si l'on adresse de très grandes quantités (~100 000). En pratique, les circuits à la demande associent très souvent des sous-blocs 'précaractérisés' et des blocs conçus 'sur mesure'. Le sur mesure peut s'imposer pour des développements justifiant de densités d'intégration très grandes (exemple :mémoire, capteurs d'images), ou de performances élevées (électronique rapide, analogique, faible bruit, électronique de puissance, basse consommation ,..etc).

III.3.2.2. Les circuits précaractérisés « STANDARD CELLS » :

Les circuits précaractérises (Standard cells) sont constitués à partir de cellules élémentaires plus ou moins complexes, existant virtuellement sous forme de description logicielle dans une bibliothèque informatique délivrée par le fondeur de silicium. La conception des circuits analogiques et numériques est possible, en fonction du contenu de la bibliothèque mise à disposition. La mise en œuvre de ces cellules implique l'utilisation de la panoplie complète des outils de CAO. La réalisation physique du composant passe par toutes les étapes technologiques du processus. Les circuits ainsi obtenus présentent des performances élevées.

Le prix de production unitaire peut être réduit pour de fortes quantités du fait d'une densité d'intégration accrue : seules seront gravées dans le silicium les cellules ou opérateurs nécessaires à la réalisation du circuit.

III.3.2.3. Les circuits prédiffusés :

Les réseaux prédiffusés sont des matrices de portes, ou de cellules plus ou moins élaborées, déjà diffusées dans le silicium, mais non interconnectées. Pour que le réseau devient opérationnel, il faut donc réaliser les interconnexions, ce qui implique l'élaboration du ou des derniers masques de fabrications. En d'autres termes, le prédiffusés correspond à un produit semi-fini qui nécessite une collaboration étroite entre le futur utilisateur et le fabricant du circuit. On parlera ici de fabricants de semi-conducteurs sans que cela implique qu'ils soient nécessairement les réels producteurs de circuits.

Des bibliothèques de fonctions élémentaires (portes, bascules....) fournies par le fondeur, permettent au concepteur de retrouver les fonctions standards classiques sur son outil CAO.

Le cycle de conception et de fabrication des masques de personnalisation du circuit est court, mais le nombre et la complexité des cellules sont limités par construction. Le coût en production est relativement faible.

III.3.2.4.Les composants programmables :

Considérés comme un des fers de lance de l'électronique moderne, les composants programmables ont connu ces derniers temps un développement considérable dans de nombreux domaines. Ils disposent, en effet, d'un avantage conséquent aussi bien dans la conception d'un produit que lors de sa production ou sa maintenance.

Apres un début pénible d'un concept trop novateur, PLD et FPGA progressent et gagnent ainsi du terrain sur les circuits processeurs qu'ils visent à remplacer ou plutôt à compléter. Ce développement est accentué par les demandes incessantes de certains utilisateurs expérimentés. Ces derniers ont vite compris l'intérêt majeur apporté par une réalisation assez simple d'un composant complexe qui se fait donc sans masque en interconnectant des macro-cellules logiques ou de portes prédiffusés de façon modifiable ou non.

Les principaux attraits de ces composants sont : une rapidité importante qui satisfait les exigences de certains algorithmes fonctionnant en temps réels, ainsi qu'une souplesse qui rend possible des réalisations divers. Afin d'optimiser encore plus ces circuits, on améliore leur densité ainsi que leur vitesse pour cela, les fabricants ont développés des composants structurels et technologiques diversifiés. Autre développement majeur est celui des logiciels de développement, qui peuvent actuellement être installés sur un simple micro ordinateur, et déjà réalisé l'exploit de concevoir n'importe qu'elle application en configurant chez soit un circuit contenant plusieurs milliers de portes logiques.

III.3.3. PLD (Les réseaux logique programmables) :

Les réseaux logiques programmables sont des circuits très proches des composants standards, en ce sens qu'ils sont complètement achevés et disponible sur catalogue. Cependant, ils sont personnalisables par l'utilisateur lui-même, à l'aide d'un outil de développement .

On distingue les produits suivants :

1. PROM (Programmable Read Only Memory) :

La structure d'une PROM, si on représente le contenu du décodeur duquel sont issues les lignes de sélection, peut se représenter schématiquement par :

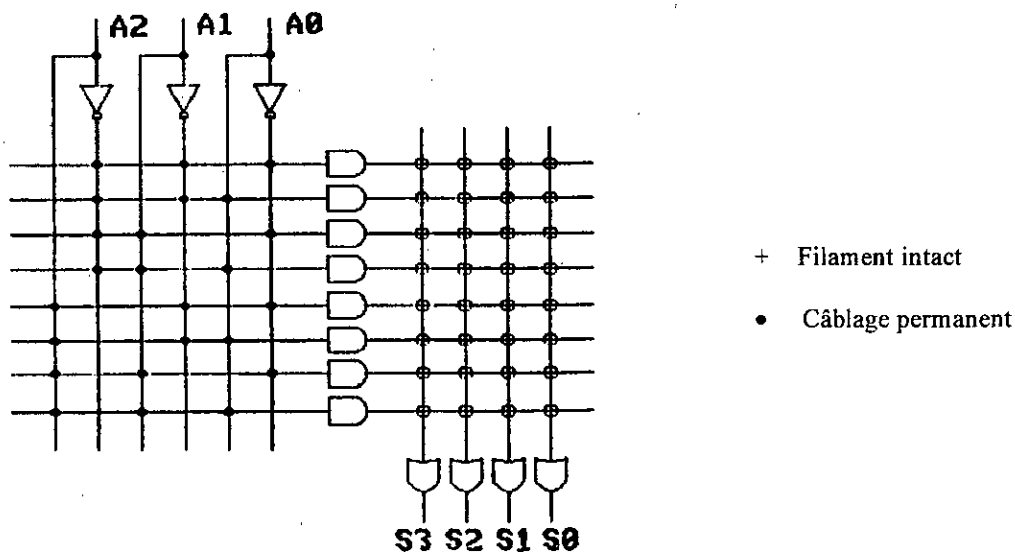


Figure (III.1): Structure logique d'une PROM à fusibles

Une ligne horizontale est activée si tous les fils d'adresse correspondants sont à 1 (portes ET), et une ligne de sortie est activée si l'une des horizontales connectées est à 1 (portes OU). On remarque que le coté gauche est une matrice de connexions pré-programmées, correspondant à la sélection d'adresse, tandis que les liaisons coté droit sont programmables (fusibles ou jonctions claquables par exemple) pour fixer le contenu de la mémoire.

Les autres circuits logiques programmables sont basés sur une structure similaire convertisseurs de codes, tables de mémorisation.

2. ***PAL (Programmable Array Logic) :***

Les circuits PAL reprennent le principe du PROM avec une matrice de connexions programmable cote ET et prédéterminée cote OU .

3. ***PLA (Programmable Logic Array) :***

Les PLAs sont des circuits dont les deux matrices (coté ET et coté OU) sont programmables.

III.3.4. Les FPGA:

Sous ce terme on trouve aujourd'hui les composants que l'on pourrait définir comme des PLD complexes ou encore comme des prédiffusés programmables.

On retrouve les circuits MPGAs 'Mask Programmable Gate Arrays'. Un MPGA est un circuit VLSI (very large scale integration) dans lequel toutes les couches des masques qui définissent la circuitérie de la puce sont prédéfinis par le fabriquant à l'exception des couches métalliques qui permettent de réaliser les connexions entre les transistors de la matrice MPGA. l'avantage principal des MPGAs est qu'ils permettent l'implémentation de circuits plus large.**[2]**

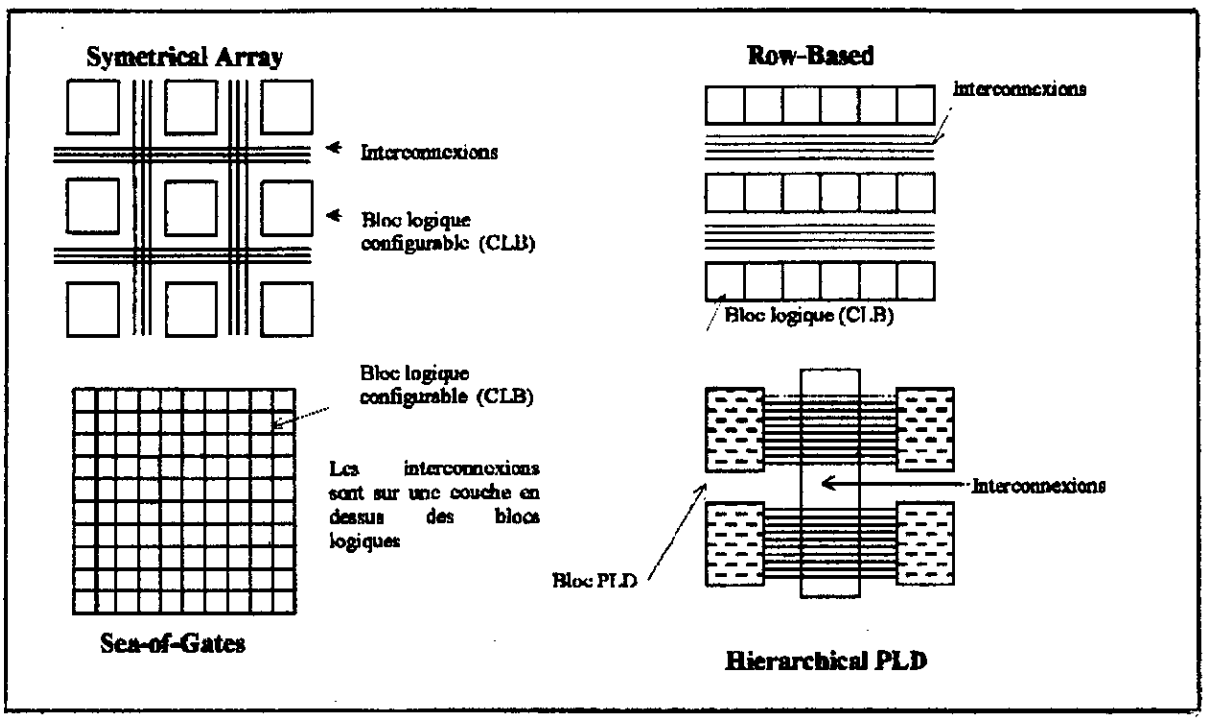
Ceci est du principalement à la structure de leur interconnexion qui est proportionnelle à la logique utilisée. Néanmoins ,ces circuits nécessitent un retour chez le manufacturier ; par conséquent le temps de fabrication est assez élevé.

Dans l'industrie électronique, il est vital d'atteindre le marché avec de nouveaux produits en un temps le plus court que possible. Une telle réduction du temps de développement (conception) et de production est devenue essentielle. D'un point de vu économique, les circuits FPGAs ont émergés comme une solution ultime aux problèmes de risques car ils permettent une production instantanée et des prototypes à faible prix comparé aux MPGAs.

D'un point de vue technologique, un FPGA combine la programmabilité d'un PLD et la structure des interconnexions d'un MPGAs. Ce qui résulte en des composants logiques programmables et à très grande densité d'intégration.

La figure *(III.2)* montre les quatre classes principales commercialement disponibles des FPGAs :'Symmetrical array' ,'Row-based array',' Sea-of-gates', 'Hierarchical PLD' .

Deux paramètres caractérisant un FPGA : les blocs logiques et les ressources d'interconnexions. Un circuit logique peut entre implémenté dans un FPGA en partitionnant la logique dans les blocs logiques individuels et en interconnectant ces blocs à travers les switches nécessaires.



Figure(III.2): Les quatre classes des FPGA commercialement disponibles [2]

Les interconnexions entre les blocs logiques doivent être programmées pour réaliser la connexion désirée. Il existe deux types d'interconnexions :

- les interconnexions reprogrammables,
- les interconnexions programmables une seule fois « one time programmable ».

puisqu'il existe plusieurs façons pour implémenter un élément programmant, il est devenu d'usage de parler de la technologie de programmation. Les technologies de programmation qui sont actuellement utilisées dans les produits commerciaux sont: les cellules statiques RAM (SRAM), les « antifuses », les transistors EPROM .

Particulièrement, la compagnie Xilinx offre des circuits FPGAs reprogrammables, utilisant la technologie de programmation SRAM. Dans ce type de circuits, la connexion entre les différents blocs logiques est assurée par un programme enregistré dans une mémoire. Ce programme est chargé automatiquement lors de chaque mise en route dans une mémoire interne du FPGA . La modification du programme peut changer le fonctionnement du circuit même en cours de marche du système, cette flexibilité peut être assez intéressante pour certaines applications.

III.4. Conclusion :

Dans ce chapitre nous avons présenté Le fondement de base de la technologie FPGA.
Dans la conception des circuits intégrés on peut citer deux méthodes de conceptions :

1. PLD (Les réseaux logique programmables),
2. les FPGA .

Cette dernière constituée par des blocs logiques et des ressources d'interconnexions, un circuit logique peut être implementé dans un FPGA tel que la logique est partitionnée dans des blocs logiques individuels et en interconnectant ces blocs à travers des switches nécessaires. Malheureusement on n'a pas pu entamer la partie implémentation réel en FPGA du circuit digitale qu'est le réseau de neurones, on a seulement simuler les différents circuits et les architectures à l'aide du simulateur XILINX.

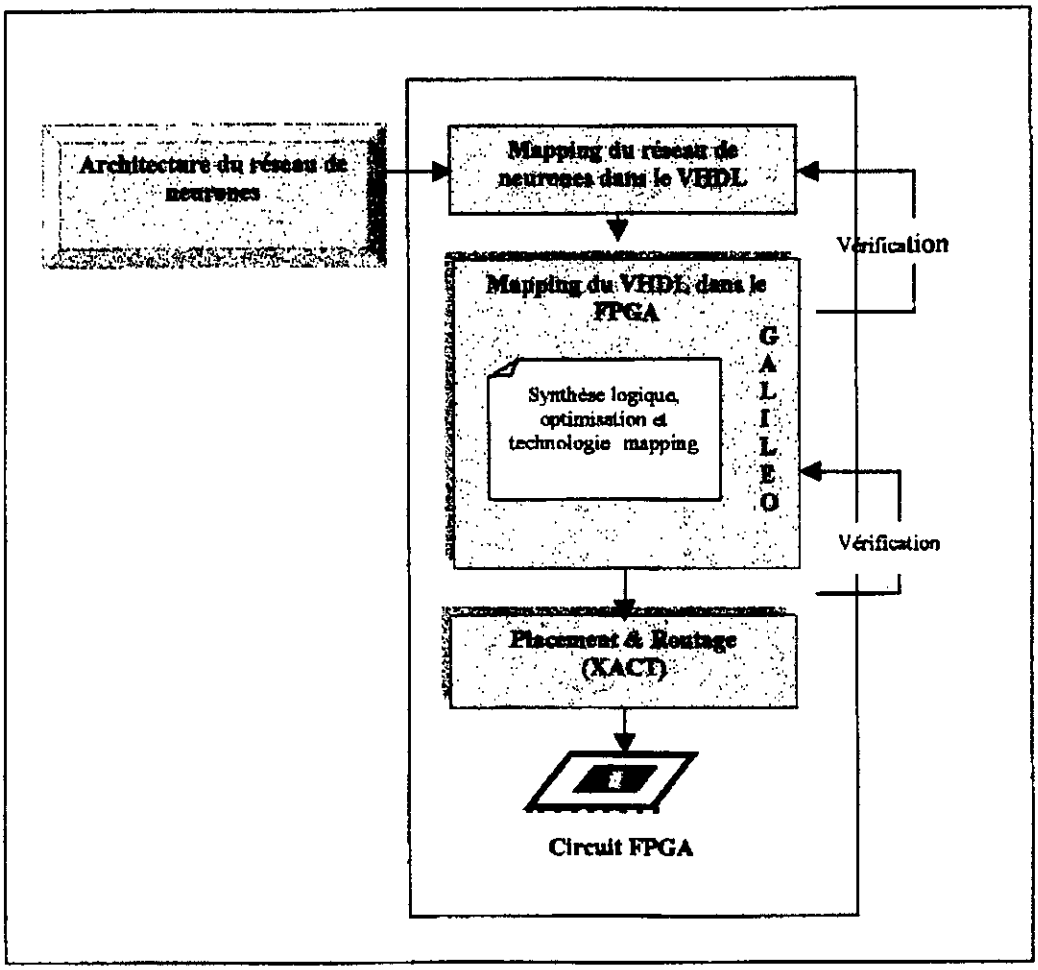
CHAPITRE IV
LES ARCHITECTURES HARDWARES DES RNAs

IV.1.Introduction :

L'approche de conception est l'approche descendante basée principalement sur la synthèse avec le VHDL. Une architecture est fixée pour le réseau de neurones, cette phase est suivi par le mapping de cette architecture en une description VHDL. Par la suite, un outil de synthèse GALILEO qui réalise le mapping du VHDL en technologie FPGA, selon les contraintes de vitesse ou surface.

Le résultat de synthèse est une netlist en format XNF. Celle-ci est utilisée à l'entrée de l'outil XACT de la compagnie XILINX qui réalise le placement et le routage du réseau de neurones.

Une vérification de la fonctionnalité du circuit est nécessaire à chaque phase.



Figure(IV.1):Méthodologie de conception du réseau de neurones[2]

Lors de la conception d'un réseau de neurones, les paramètres qu'ils faut prendre en considération lors de la conception d'un réseau de neurones sont :

- Le parallélisme qui permet le traitement en temps réel.
- Les performances relativement par rapport à la complexité de connexions synaptiques.
- La flexibilité ou adaptabilité du circuit .
- La surface du silicium

Pour atteindre ces objectifs, une bonne implémentation VLSI doit avoir les propriétés architecturales suivantes :

- simplicité de conception caractérisée par une architecture basée sur des copies de quelques cellules simples.
- Régularité de la structure permettant de réduire les interconnexions.
- Possibilité d'extension et de réduction de l'architecture.

La conception d'une architecture complètement parallèle nécessite :

- Le parallélisme entre toutes les couches du réseau de neurones : cela signifie qu'au moins un multiplieur doit être utilisé pour chaque couche .
- Le parallélisme des nœuds ou neurones : pour réaliser cette condition, un seul neurone suffit.
- Le parallélisme des connexions synaptiques : c'est le plus haut degré de parallélisme, souhaité dans un réseau de neurones. Pour atteindre ce parallélisme, il faudrait prévoir un multiplieur pour chaque connexion .

Cependant, l'utilisation d'un grand nombre de multiplieurs pour un grand nombre de connexions synaptiques est une pénalité sévère pour les circuits digitaux et plus particulièrement pour les circuits FPGAs à cause de leurs ressources limités(complexité d'intégration et problèmes de delai).

Par conséquent, nous considérons souvent uniquement le parallélisme des nœuds(architecture séquentielle), par ailleurs, le transfert de données entre couche se fait en série car un neurone est conçu pour calculer une seule connexion synaptique à la fois .

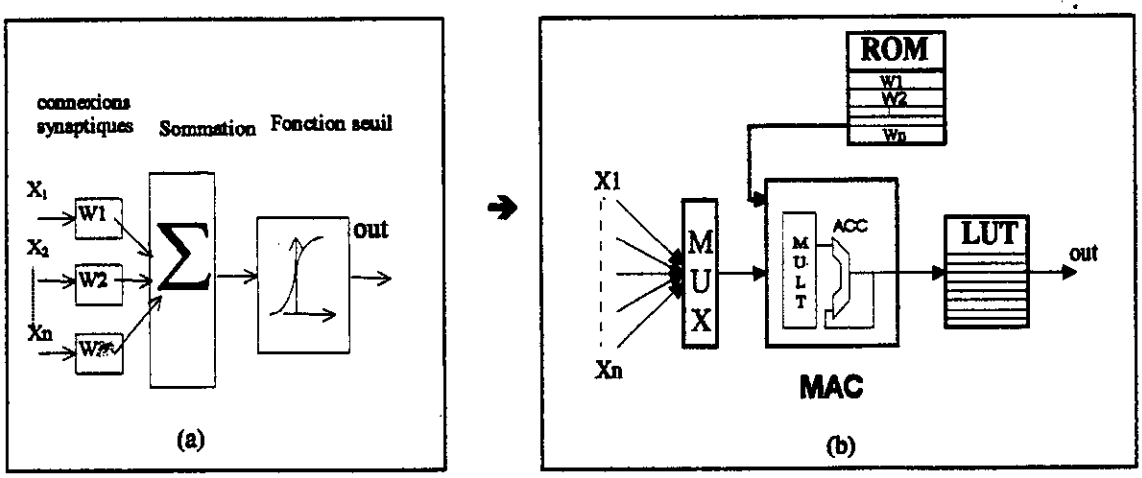
Dans ce chapitre on va présenté deux architectures digitales des RNAs.

VI.2. Architecture séquentielle d'un RNA:

VI.2.1.Architecture du neurone :

Nous proposons une architecture FPGA, pour le modèle de neurone artificiel (voir figure(VI.2(a)),comme l'illustre la figure(VI.2(b)),ce modèle est principalement basé sur :

- un circuit mémoire (ROM) permettant de stocker les poids synaptiques.
- Un circuit multiplieur accumulateur (MAC)qui calcul les sommes pondérée.
- Une table de transfert (LUT)ou ROM qui implémente la fonction d'activation.**[2]**



Figure(IV.2(a)):Neurone artificiel ***Figure(IV.2(b)):Modèle FPGA équivalent***

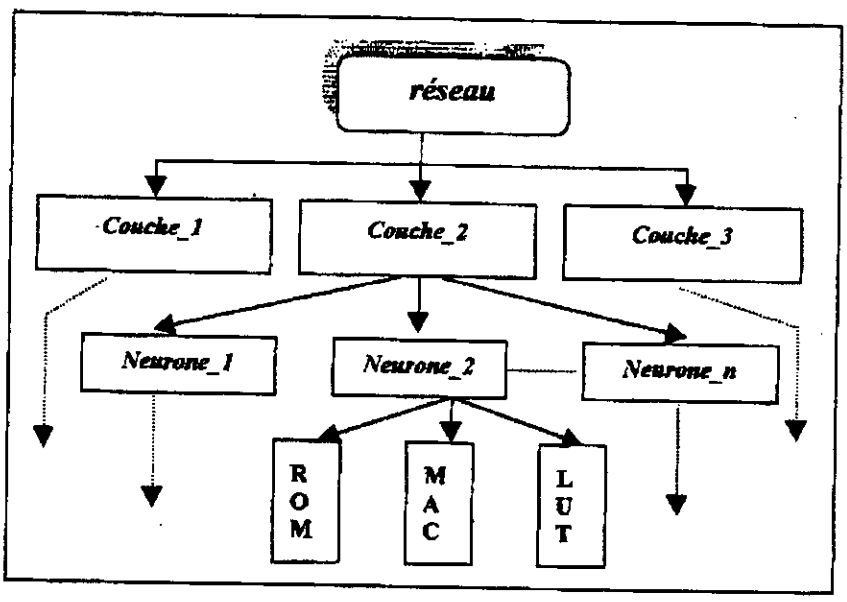
IV.2.2.Architecture du RNA :

L'architecture FPGA résultante du réseau est présentée suivant la figure(VI.3).Elle présente les caractéristiques suivantes :

- pour un même neurone, seulement un seul circuit MAC est utilisé pour calculer la somme des produits.
- Chaque MAC possède sa propre ROM ou sont mémorisés les poids synaptiques. La profondeur de chaque ROM est égale au nombre de nœuds relativement à sa couche d'entrée, pour la même couche, un calcul parallèle des neurones est réalisé.
- Le calcul entre les couches se fait en série.
- Le réseau de neurone est contrôlé par une unité de contrôle.
- Pour la même couche , un calcul parallèle des neurones est réalisé.

Comme nous pouvons le constater l'architecture résultante :

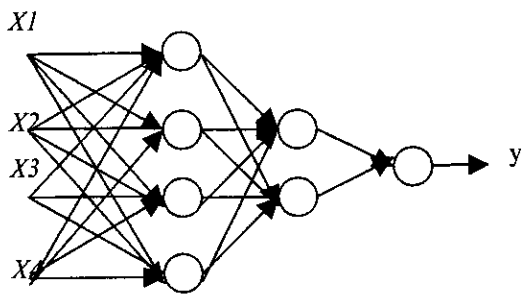
- intègre un haut degré de parallélisme.
- Une simplicité de conception caractérisée par une répétition(ou copie) de cellules simples.
- Une régularité de la structure.



Figure(IV.3): Approche pour la description d'un réseau de neurones

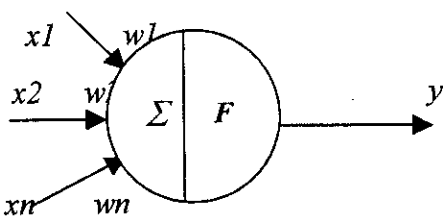
VI.2.3.Architecture détaillée :

Dans notre étude ,on prend un réseau de neurone(simple) pour pouvoir présenter les architectures utilisées, que ce soit séquentielle ou systolique.
Donc le réseau de neurones à implémenté est le suivant (figure(IV.4)).



Figure(IV.4) :réseau de neurones à implémenté

Le modèle mathématique du neurone est le suivant :



$$\Leftrightarrow \quad y=F(\sum x(i)\, w(i))$$

Ou F est une fonction d'activation , on suppose pour simplifier l'étude que le nombre des vecteurs égalent à quatre, chaque vecteur quatre bits.

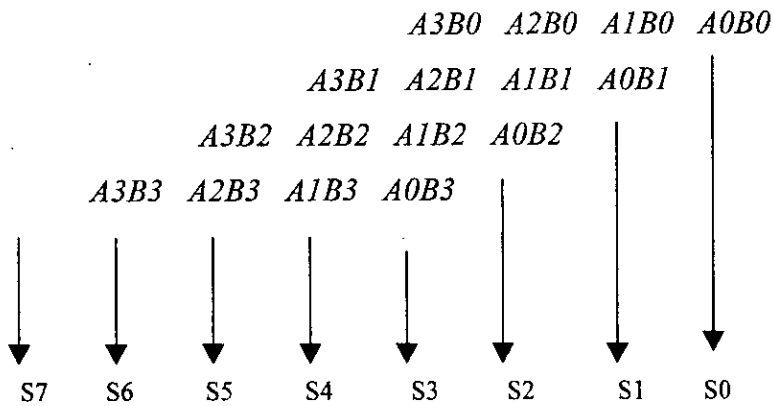
Maintenant on va donner les architectures des blocs consituants un neurone(Multiplieur, Aditionneur complet, table de look-up,mux à 4 bits).

VI.2.3.1.Le Multiplieur:

La conception de multiplieur est la suivante:

Si on veut faire la multiplication de deux mots de 4bits : $A \times B = S$

$$\begin{array}{r} A3\ A2\ A1\ A0 \\ \times \\ B3\ B2\ B1\ B0 \\ \hline \end{array}$$



Pour construire cette opération il faut: d'abord présenter l'additionneur complet d'un seul bit:

A	B	Cin	S	Cou
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$Cout = cin(A \oplus B) + AB$, $S = (A \oplus B) \oplus S$

Figure(IV.5) : Table de vérité d'un additionneur

Et voici l'architecture en porte logique et le chronogramme de l'additionneur complet.

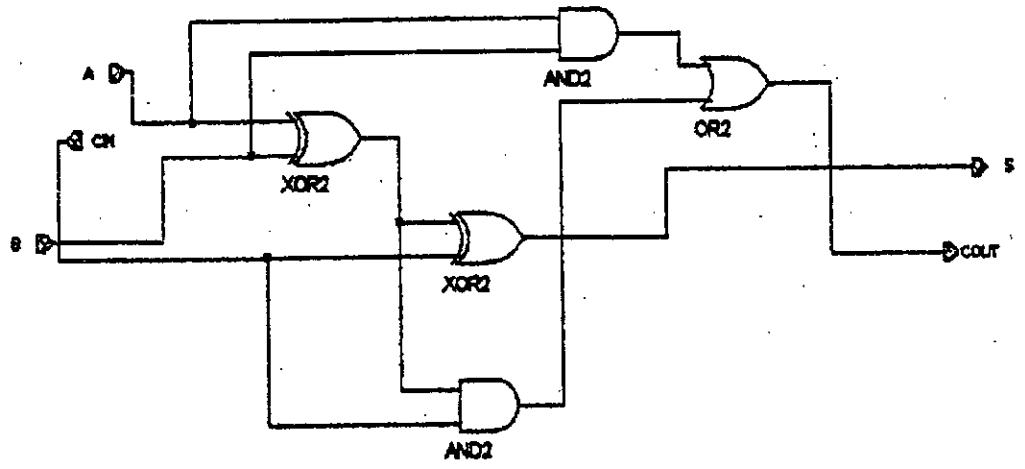


Figure (IV.6) : Représentation d'un additionneur en porte logique .

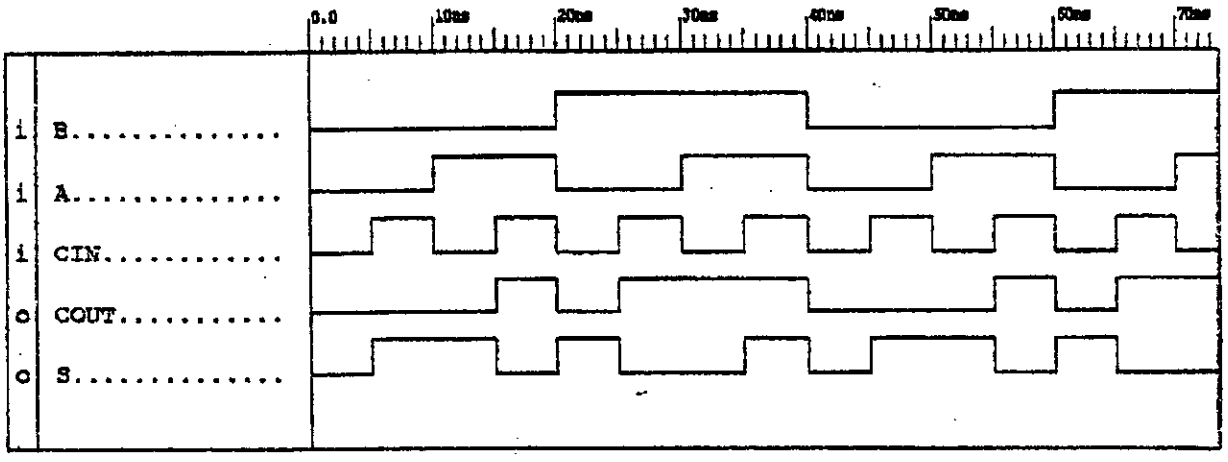
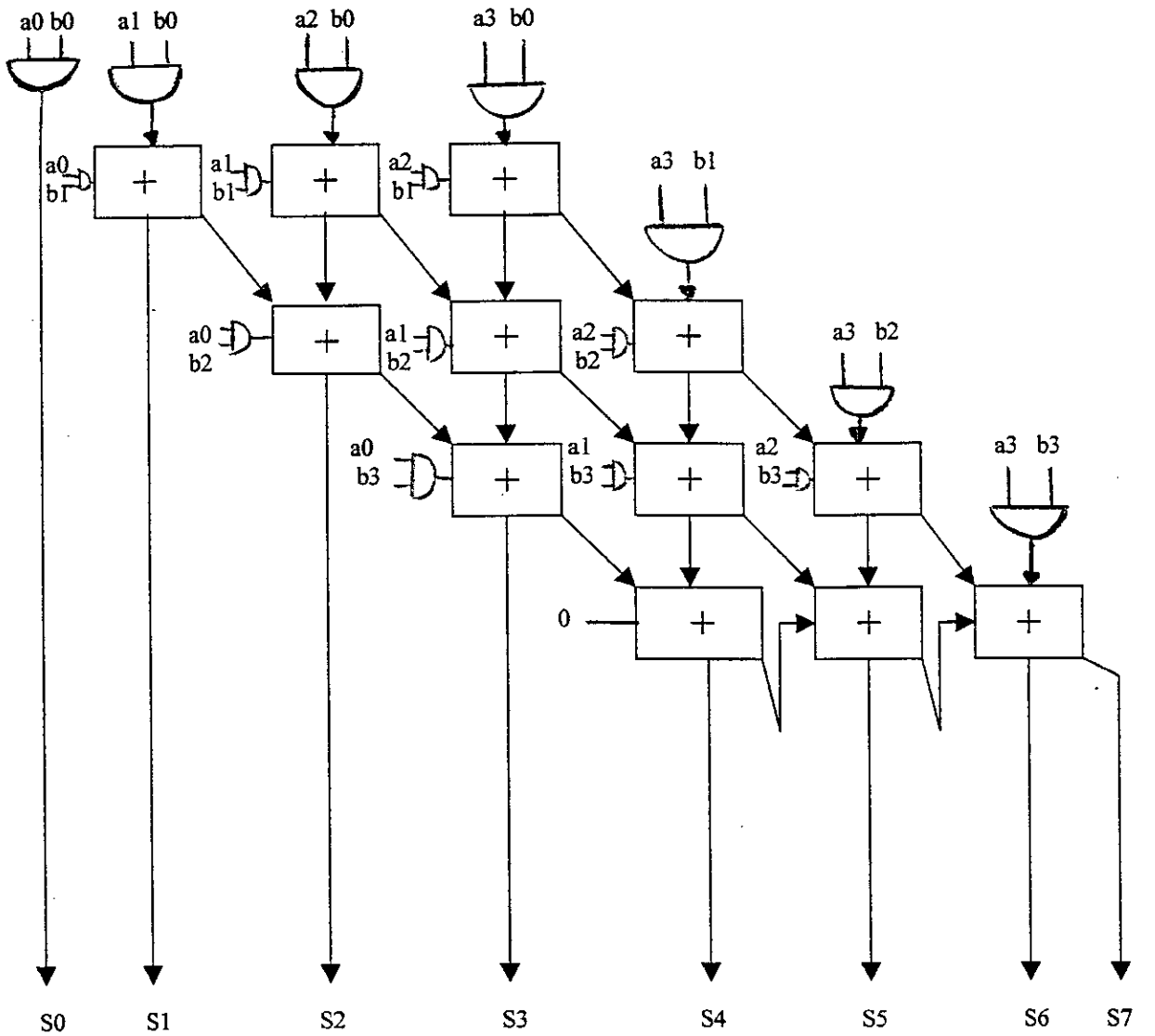


Figure (IV.7) : Chronogramme d'un additionneur d'un bit .

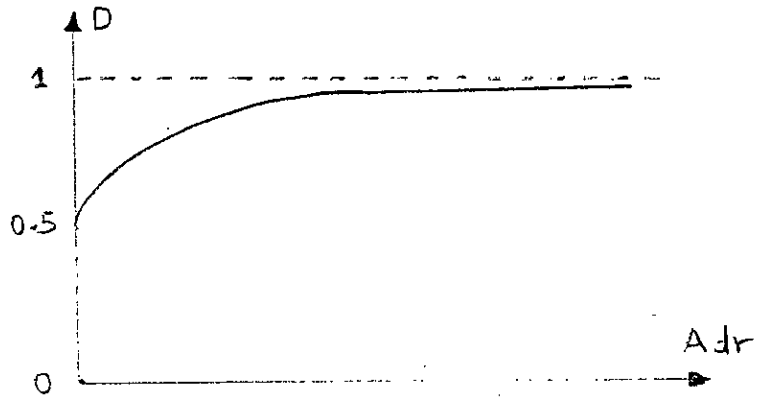
La figure (IV.8) présente le schéma d'un multiplieur de quatre bits.



Figure(IV.8): Schéma d'un multiplieur de quatre bits

VI.2.3.2. Look-up table(LUT):

La réalisation de la table de Look-up est faite pour la réalisation de la fonction d'activation selon la figure(VI .9).



Figure(IV.9):Fonction d'activation

Les données et les adresses sont des mots de quatre bits.

$$D = d_3\, d_2\, d_1\, d_0$$

Exemple:

Si on veut réaliser une fonction d'activation discrète en portes logiques, on doit d'abord ségmenter les axes D et Adr.

Pour notre cas on aura:

Base2		Base10
0.001	⟶	0.125
0.010	⟶	0.250
0.011	⟶	0.375
0.100	⟶	0.500

le pas =0.125 (base 10) = 0.001(base 2)

Donc on peut construire la table de vérité facilement,

A3	A2	A1	A0	D3	D2	D1	D0
0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Donc les équations logiques sont données par :

$D0=\overline{A2}A1A0+\overline{A3}A2\overline{A1}+A3\overline{A2}(A1+A0)$

$D1=\overline{A3}A2A1+A3\overline{A2}$

$D2=\overline{A3A2}$

$D3=A3A2$

Figure(IV.10): La table de vérité et les équations logiques d'une table de look-up

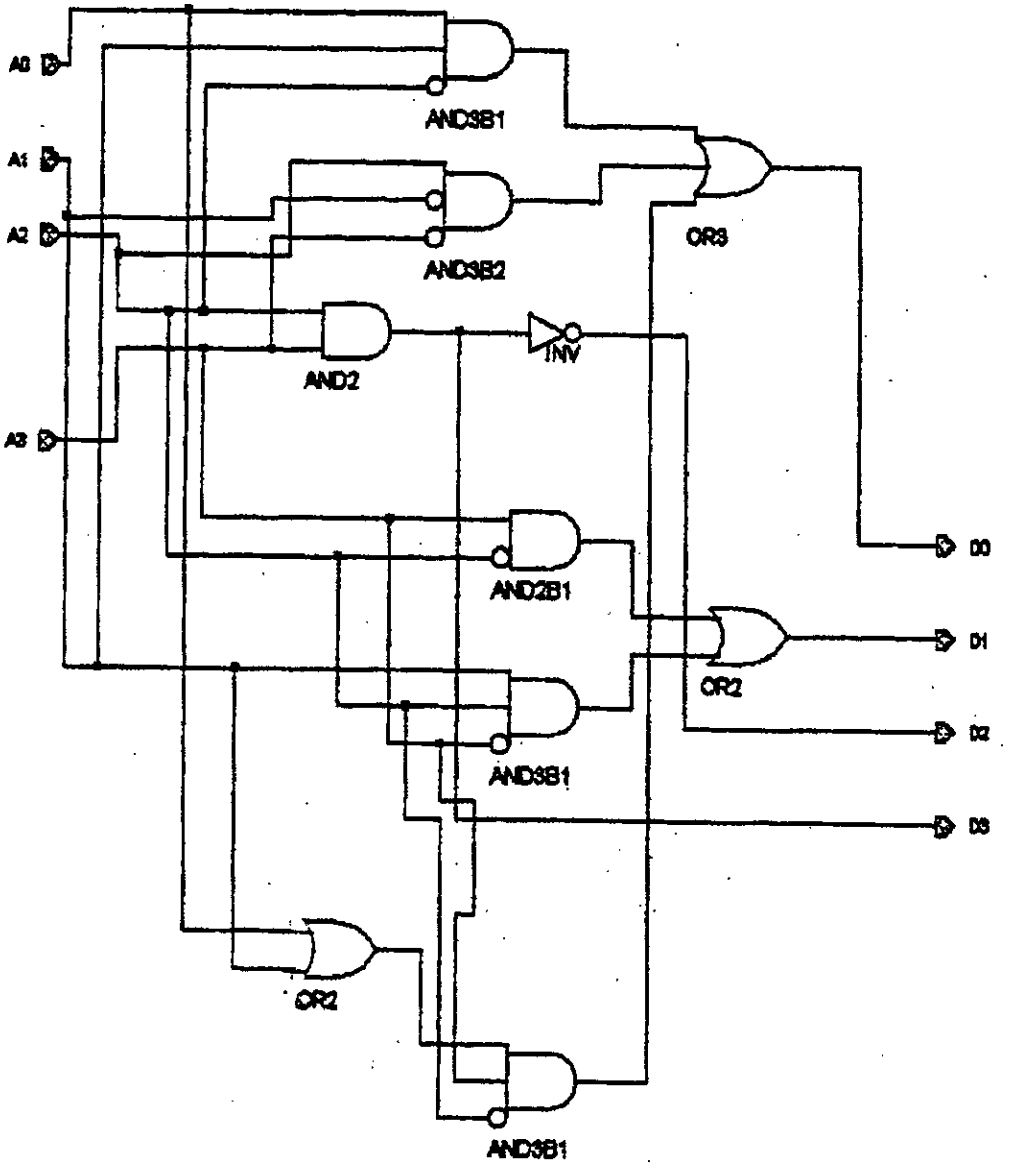
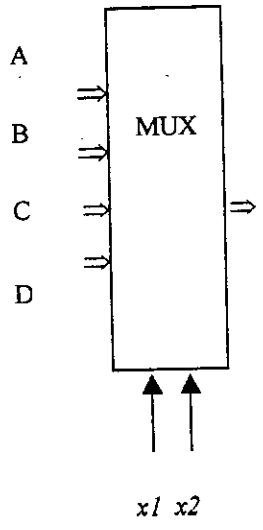


Figure (IV.11): Représentation d'une table de Look-up en portes logiques

IV.2.3.3. Multiplexeur à quatre bits:

Pour simplifier l'étude on prend un Multiplexeur à quatre entrées de quatre bits.



Y est un mot de 4bits

x1	x2	Y
0	0	A
0	1	B
1	0	C
1	1	D

Donc: $Y = A\,\overline{x1}\,\overline{x2} + B\,\overline{x1}\,x2 + C\,x1\,\overline{x2} + D\,x1\,x2$

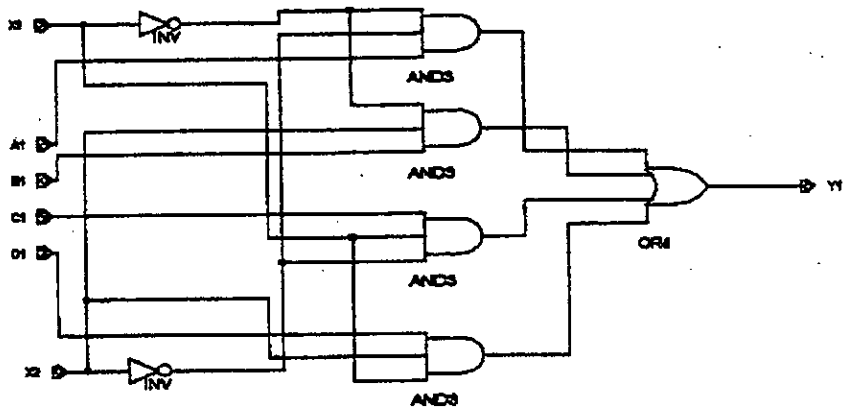
Comme A,B,C et D sont des mots de quatre bits, on peut écrire:

$$y1 = a1\,\overline{x1}\,\overline{x2} + b1\,\overline{x1}\,x2 + c1\,x1\,\overline{x2} + d1\,x1\,x2$$
$$y2 = a2\,\overline{x1}\,\overline{x2} + b2\,\overline{x1}\,x2 + c2\,x1\,\overline{x2} + d2\,x1\,x2$$
$$y3 = a3\,\overline{x1}\,\overline{x2} + b3\,\overline{x1}\,x2 + c3\,x1\,\overline{x2} + d3\,x1\,x2$$
$$y4 = a4\,\overline{x1}\,\overline{x2} + b4\,\overline{x1}\,x2 + c4\,x1\,\overline{x2} + d4\,x1\,x2$$

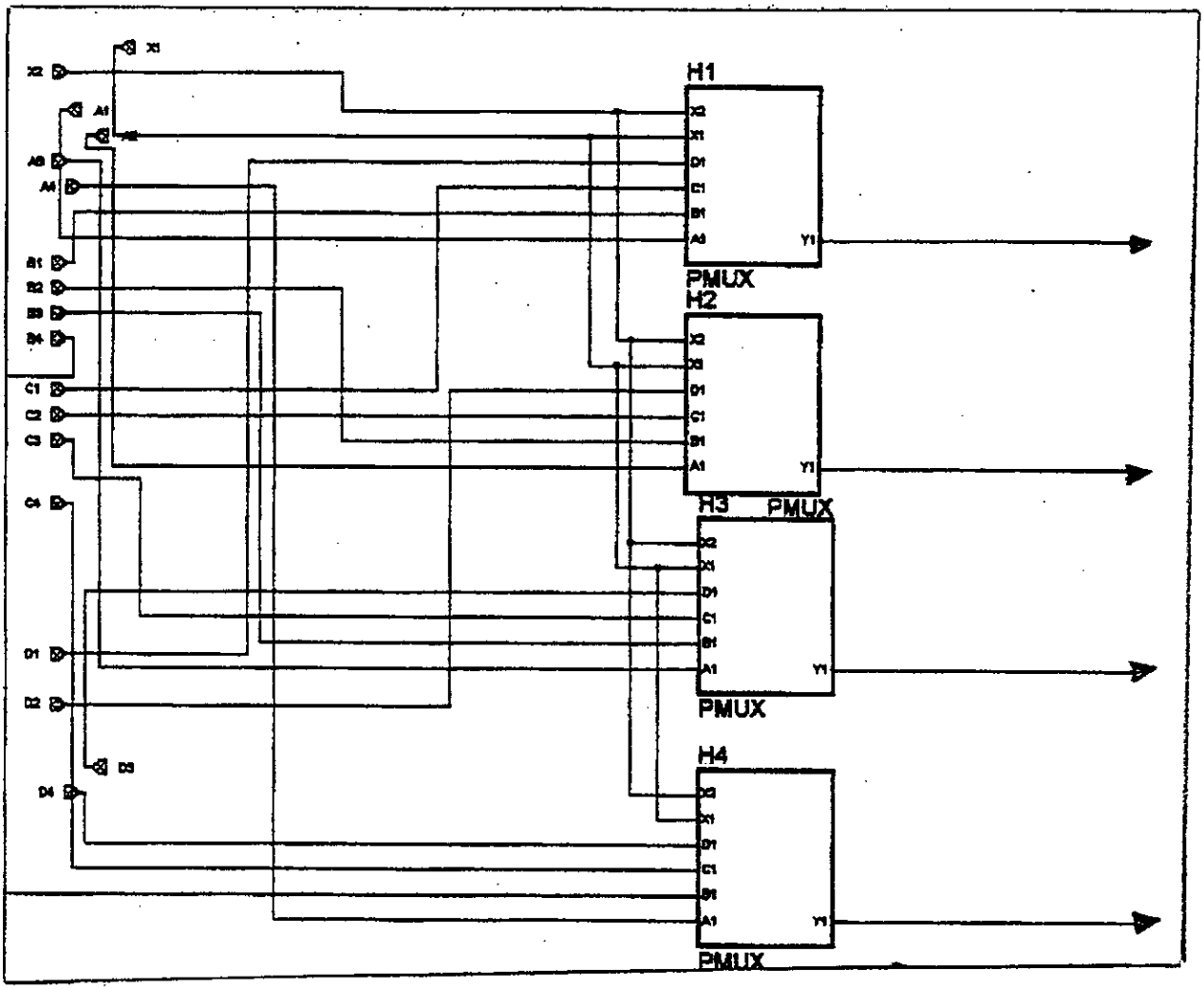
avec: $A = a1\,a2\,a3\,a4$ $B = b1\,b2\,b3\,b4$ $C = c1\,c2\,c3\,c4$ $D = d1\,d2\,d3\,d4$

$$Y = y1\,y2\,y3\,y4$$

On peut facilement représenter l'équation de y1 en portes logiques (figure IV.12), d'où on obtient un bloc qu'on vas l'utiliser pour réaliser l'architecture globale d'un multiplexeur de quatre bits(figure IV.13).

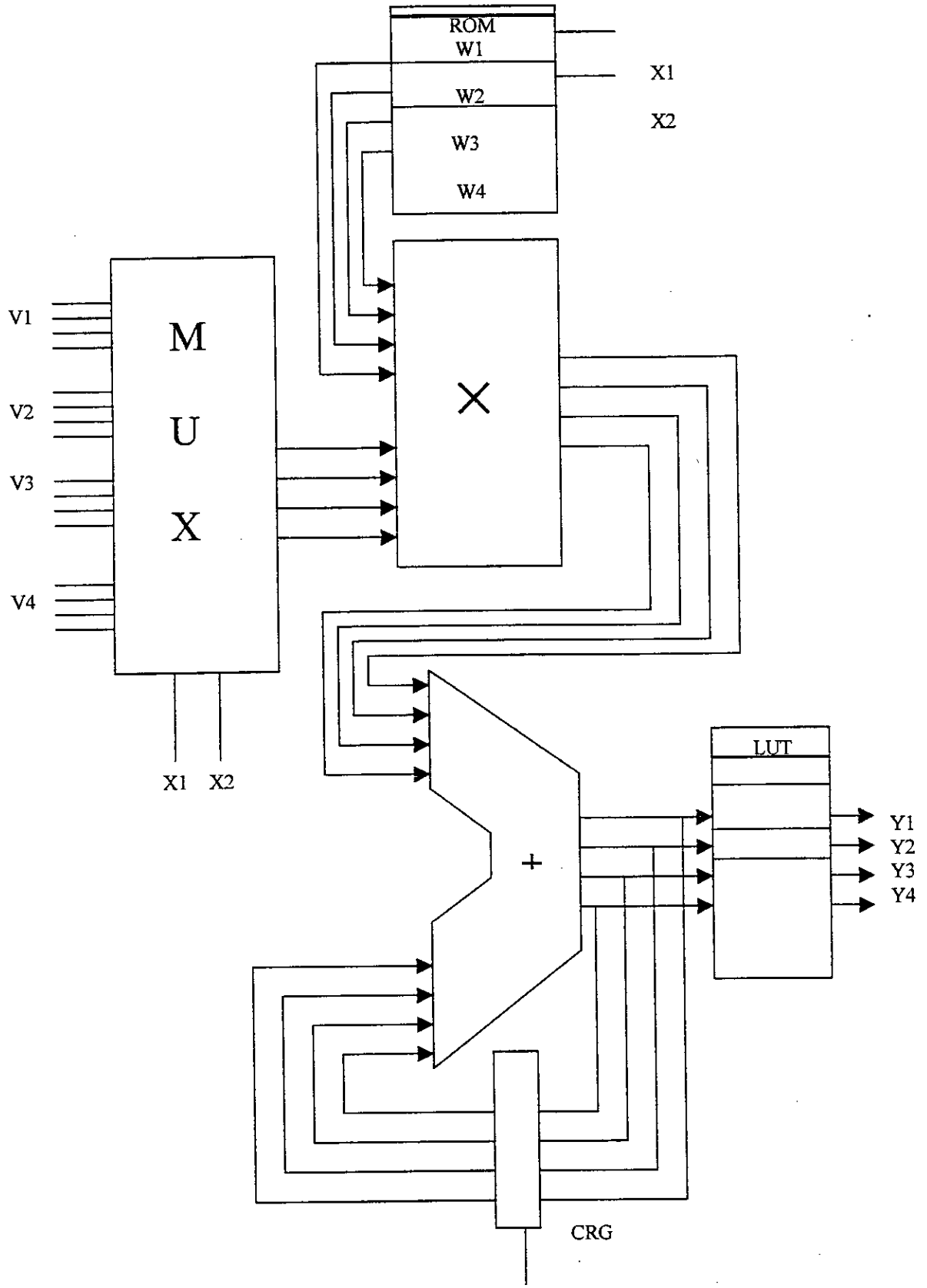


Figure(IV.12): Schéma d'un bloc constituant le multiplexeur de quatre bits



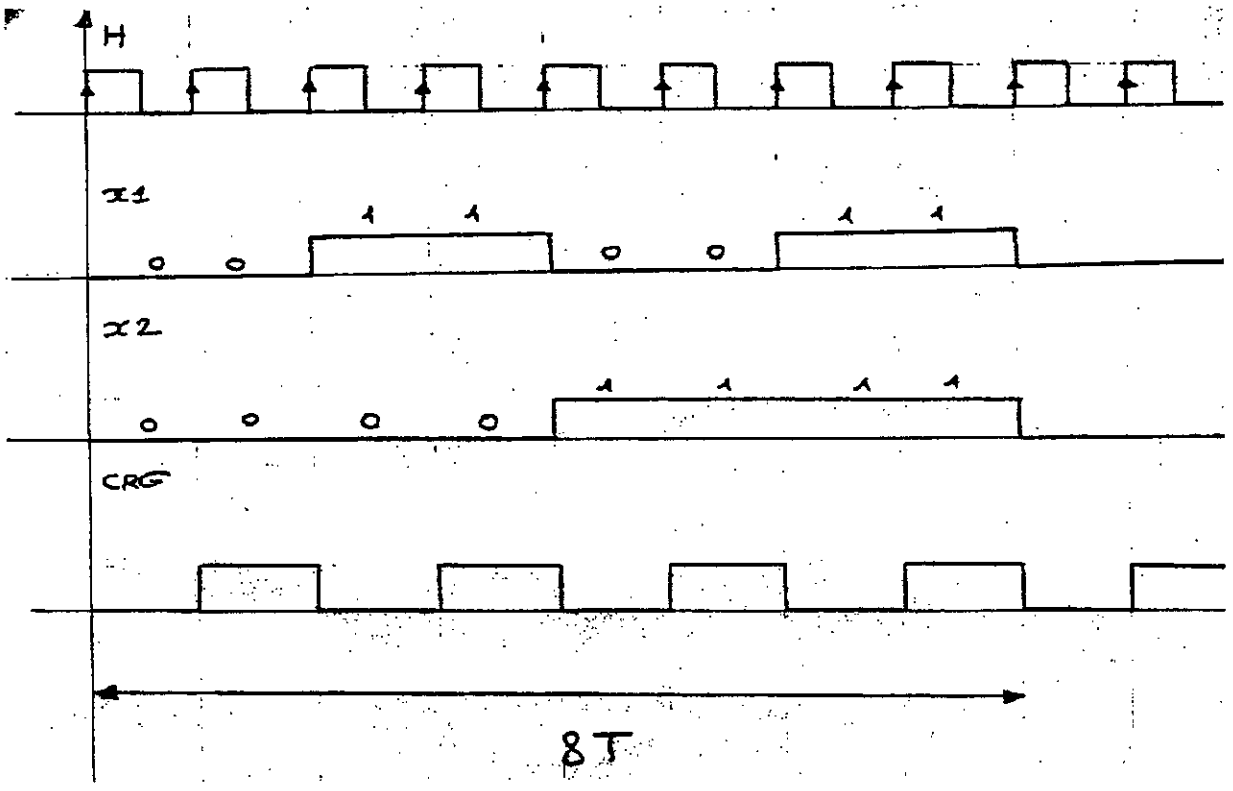
Figure(IV.13): Schéma complet d'un multiplexeur de quatre bits

Finalement on obtient l'architecture détaillée d'un neurone (figure IV.14)



Figure(IV.14): Architecture détaillée d'un neurone.

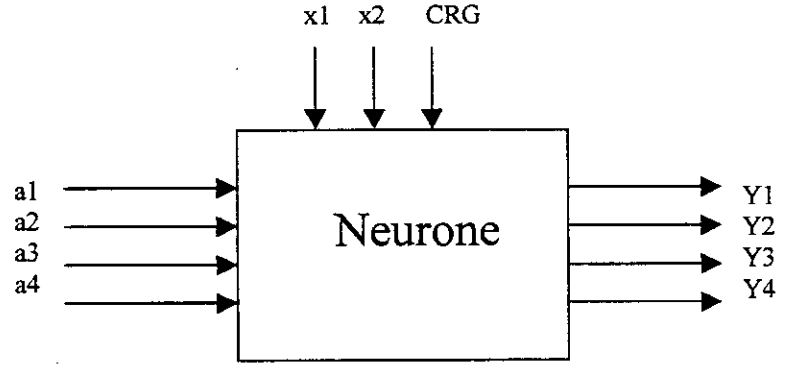
Les signaux x1,x2 et CRG sont des signaux de commande du neurone



Figure(IV.15): Chronogrammes de fonctionnment du neurone.

IV.2.3.4. Schéma complet du réseau de neurones :

On a deja développé ces blocs qui constituent le réseau de neurones, il reste seulement l'unité de commande que l'on va developper par la suite.



Figure(IV.16): Schéma bloc d'un neurone de 4 bits .

Et voici l'architecture complète d'un réseau de neurones :

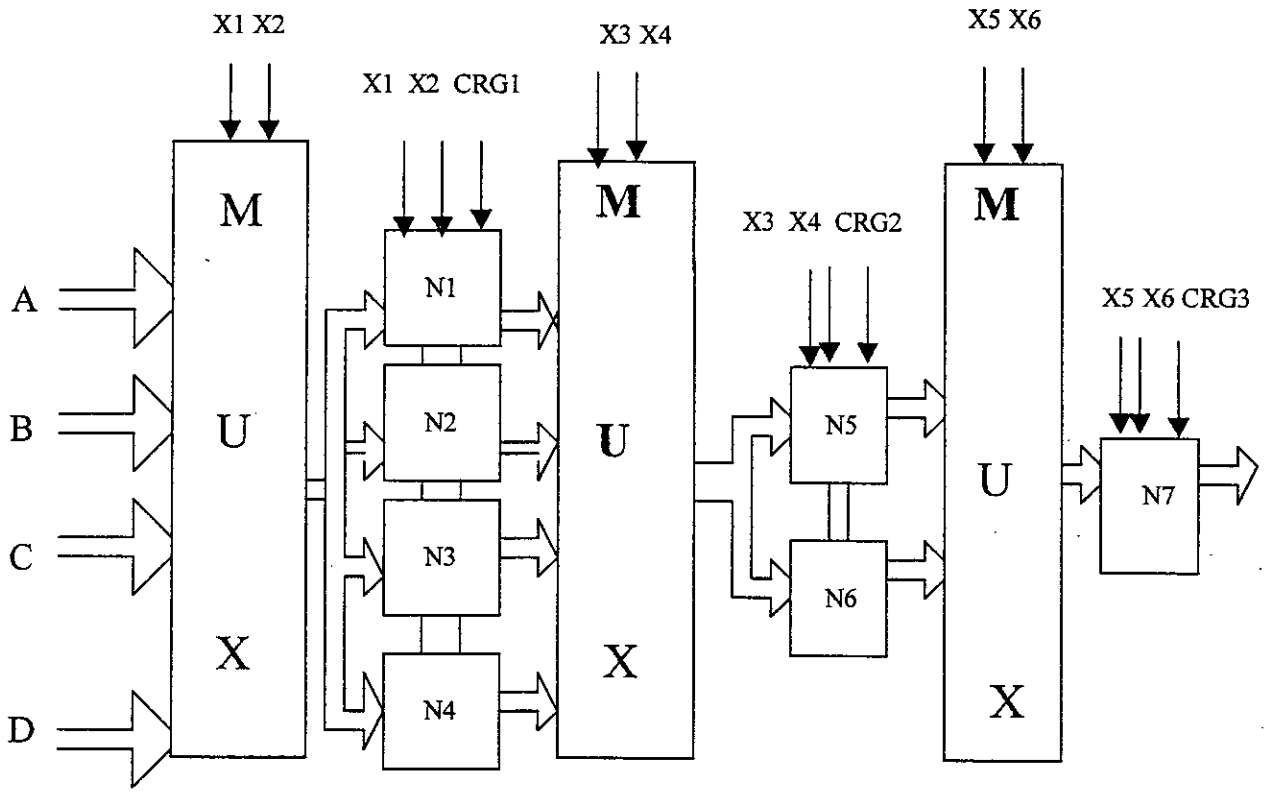


Figure (IV.17): Schéma complet d'un réseau de neurones.

IV.2.3.5. Unité de commande du réseau de neurones:

D'après le chronogramme de la figure (IV.15), on peut facilement déduire l'architecture de cette unité de commande, on remarque que ce chronogramme est périodique donc on peut prédire ces états, donc on peut par la suite construire une machine séquentielle.

Rappel :

Une machine séquentielle est représentée en général selon la figure (IV.18) :

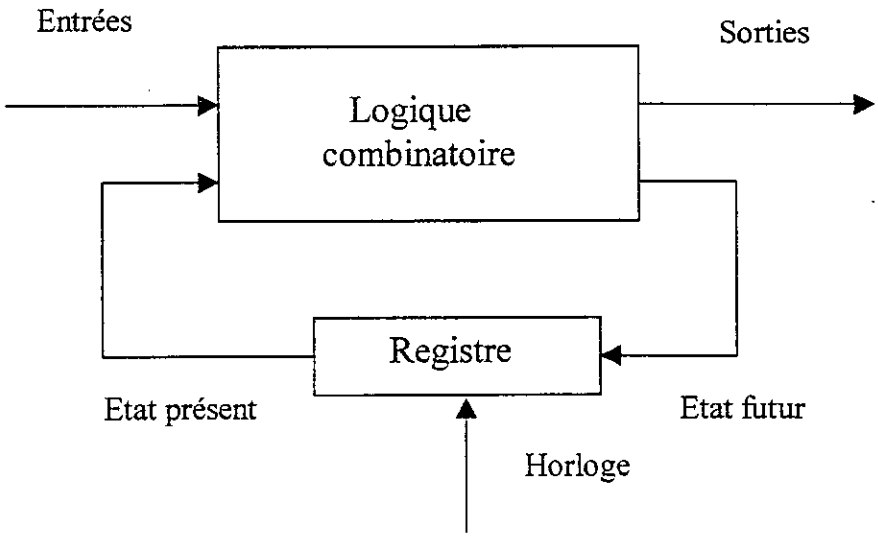


Figure (IV.18): Une machine séquentielle

Une telle machine est définie comme une fonction **M** tel que :

$M=(I,O,S,S_i,f,g)$

I : Ensemble des entrées.

O : Ensemble des sorties.

S : Ensemble des états.

S_i**:** Ensemble des états initiaux.

F : La fonction de l'état futur

$S, I \longrightarrow S$

G : La fonction de sortie.

$G : S, I \longrightarrow O$ pour la machine de Mealy.

$G : S, I \longrightarrow S$ pour la machine de Moor.

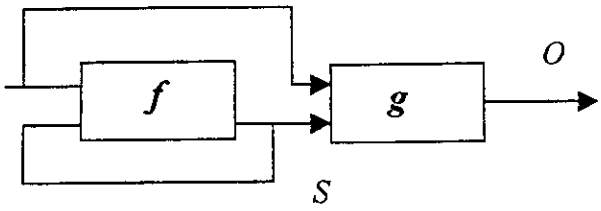


Figure (IV.19(a)): Machine de Mealy

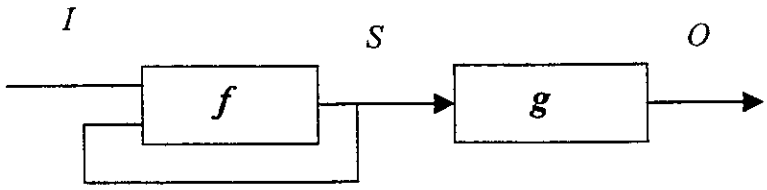


Figure (IV.19(b)): Machine de Moor

Table de transition:

X1	X2	CRG	X1+	X2+	CRG+
0	0	0	0	0	1
0	0	1	1	0	0
1	0	0	1	0	1
1	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

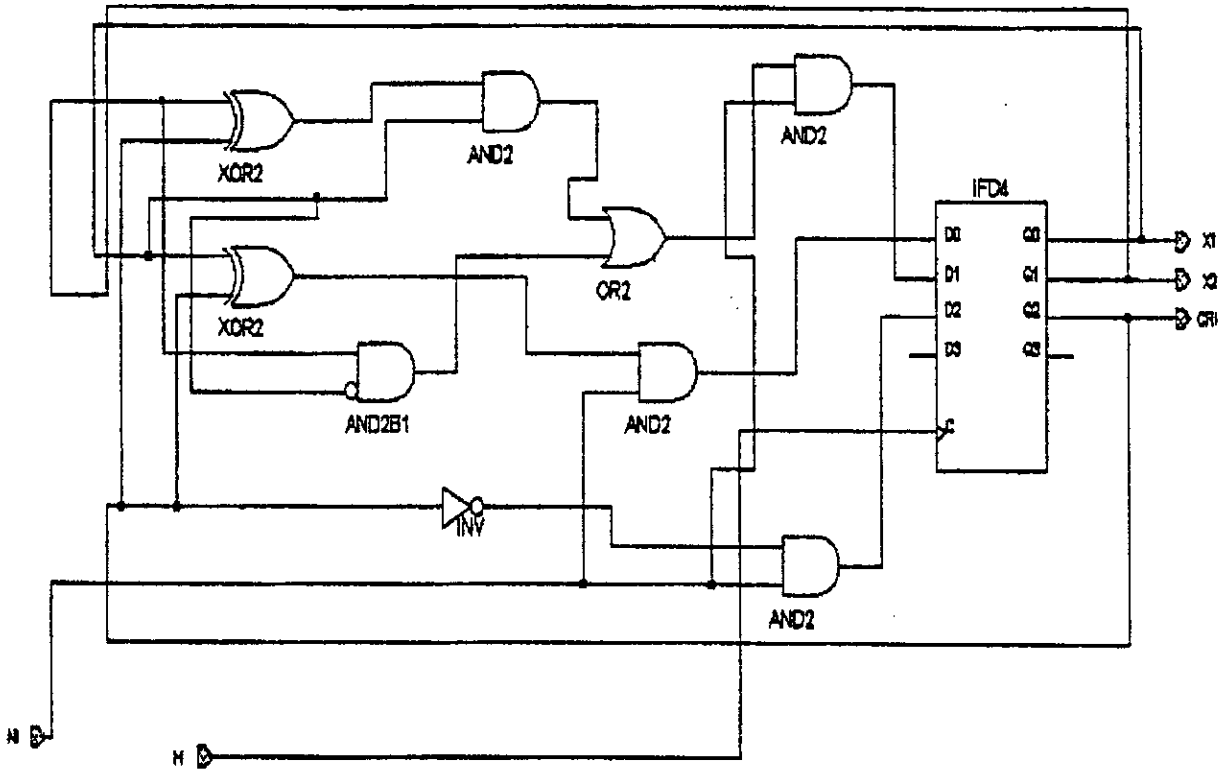
On peut facilement trouver que les états futurs (les sorties) :

$$CRG+ = \overline{CRG}$$

$$X1+ = X1 \oplus CRG$$

$$X2+ = \bar{X}1X2 + X1(X2 \oplus CRG)$$

Finalement l'architecture en portes logiques de l'unité de commande d'un réseau de neurones est donnée par la figure (IV.20).



Figure(IV.20) : Représentation de l'unité de commande en portes logiques

IV.3. Architecture syslolique d'un RNA:

IV.3.1. Introduction :

En 1978, HT. Kung et CE Leiserson ont introduit le concept de structure systolique, qui est une simple machine parallèle spécialisée, agencée en forme de réseau.

Ces réseaux se composent d'un grand nombre de cellules élémentaires identiques et localement interconnectées. Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats aux cellules voisines, un temps de cycle plus tard. Seules les cellules situées aux frontières du réseau communiquent avec le monde extérieur, les cellules évoluent en parallèle, sous le contrôle d'une horloge globale (synchronisation).

Les réseaux systoliques sont de conception facile pour la classe des algorithmes représentés par des vecteurs, matrices ou des algorithmes récursifs.

Ils sont implémentés pour effectuer un traitement parallèle des données d'un algorithme manipulant des opérations matricielles comme la multiplication (vecteur-vecteur, matrice-vecteur, matrice-matrice) et la résolution des systèmes d'équations linéaires
Les figures ci-dessous présentent quelques types de réseaux systoliques :

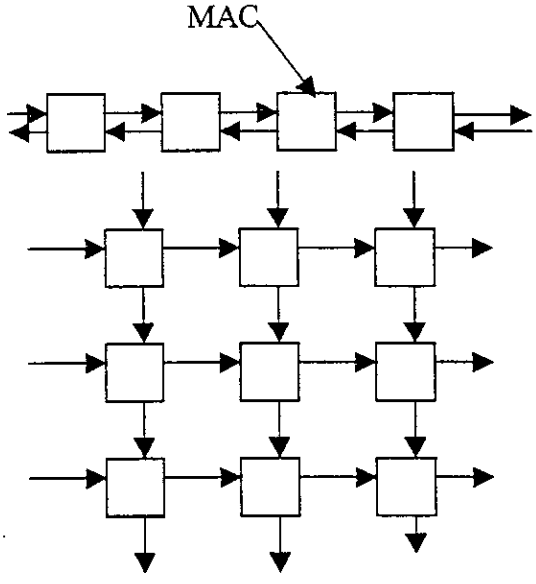
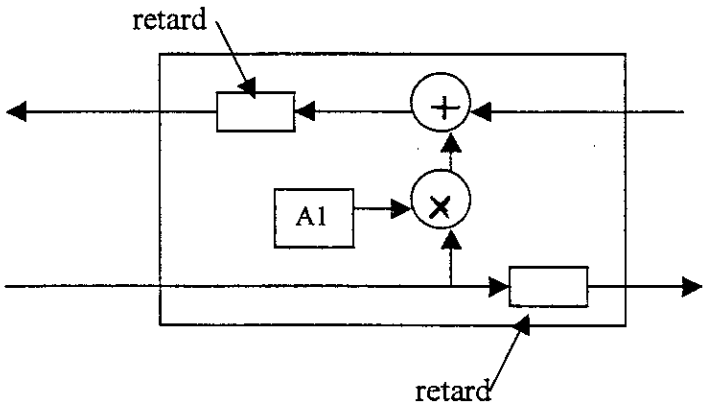


Figure (IV.21): Exemples des réseaux systoliques

La connexion entre processeur peut être unidimensionnelle formant un réseau linéaire ou multidimensionnelle en formes géométriques variantes.

IV.3.2. Le MAC :

Le MAC est un nœud qui exécute les opérations de multiplications et d'additions.
Dans notre implémentation on utilise ce type de MAC :



Figure(V.22): Schéma d'un MAC à utilisé

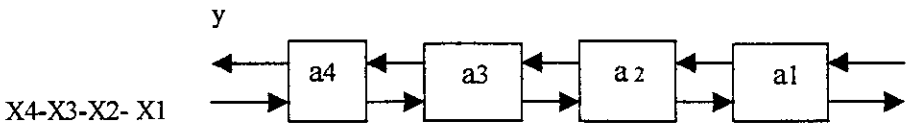
IV.3.3. Exemples d'applications:

Ex1:

Si on veut réaliser le produit vecteur *(X)*et le vecteur*(A)*:

Telque $X=(x1\ x2\ x3\ x4)$ et $A=(a1\ a2\ a3\ a4)$ alors le produit de deux vecteurs X et A est un scalaire y, tel que: $y = a_1x_1+a_2x_2+a_3x_3+a_4x_4$

L'architecture est la suivante:



Figure(IV.23):Architecture systolique linéaire

(a1 a2 a3 a4) sont stockés dans des registres ou des mémoires internes ou externes au réseau, on peut avoir ici une mémoire centralisée ou une mémoire décentralisée.

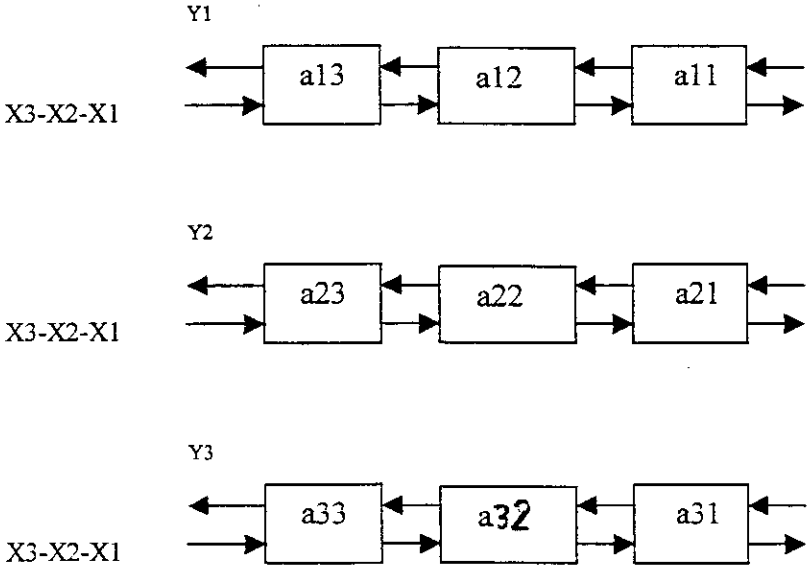
Ex2:

Soit à réaliser le produit matrice-vecteur, le résultat éventuellemnt est un vecteur.

$$\begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix} \times \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} = \begin{bmatrix} Y1 \\ Y2 \\ Y3 \end{bmatrix}$$

alors on aura deux sortes d'architecture :

La première est indiquée suivant la figure*(IV.23)*



Figure(IV.24): Architecture linéaire pour le produit matrice-vecteur

La deuxième est plus optimale que la première, on va l'utiliser par la suite pour implémenter le réseau de neurones.

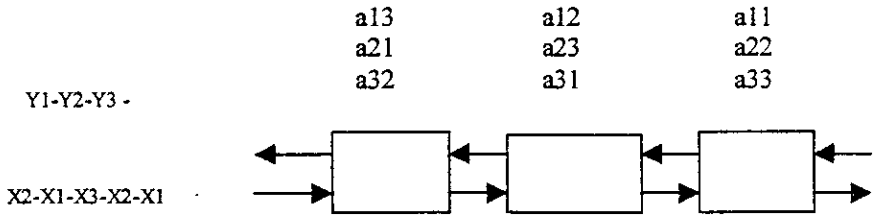


Figure (IV.25):Architecture linéaire pour le produit matrice-vecteur

IV.3.4. Implémentation d'un filtre recursif RIF à l'aide de l'architecture systolique:

IV.3.4.1. MAC:

On peut donner une architecture digitale du MAC comme suit :

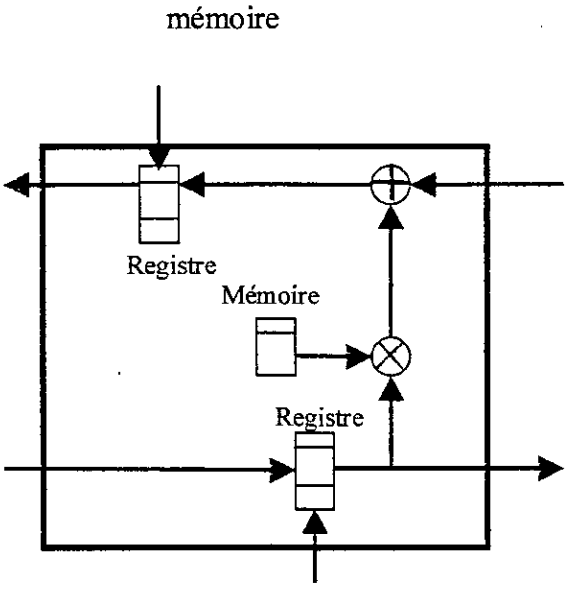


Figure (IV.26): Architecture d'un MAC

IV.3.4.2. Implémentation d'un filtre recursif :

Notre objectif est l'implémentation d'un filtre RIF dont l'équation est donnée par:

$$y(n)=a0x(n)+a1x(n-1)+a2x(n-2)+b1y(n-1)+b2y(n-2)$$

$$z(n)=a0x(n)+a1x(n-1)+a2x(n-2)$$

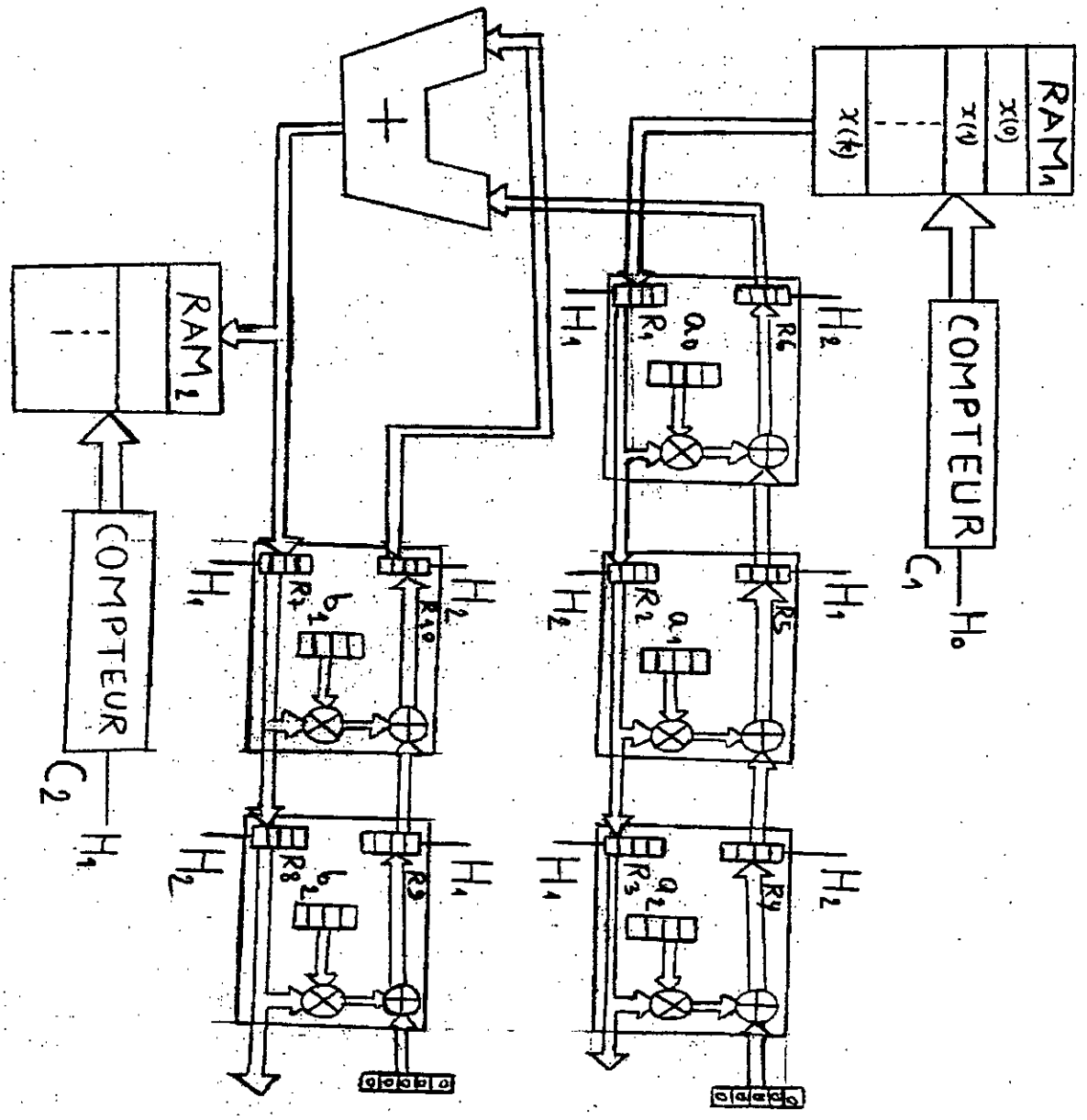


Figure (IV.27): Architecture d'un filtre recursif

Les signaux de commandes du filtre sont présentés par la figure (IV.28(a)), ils présentent le résultat de notre stratégie de conception.

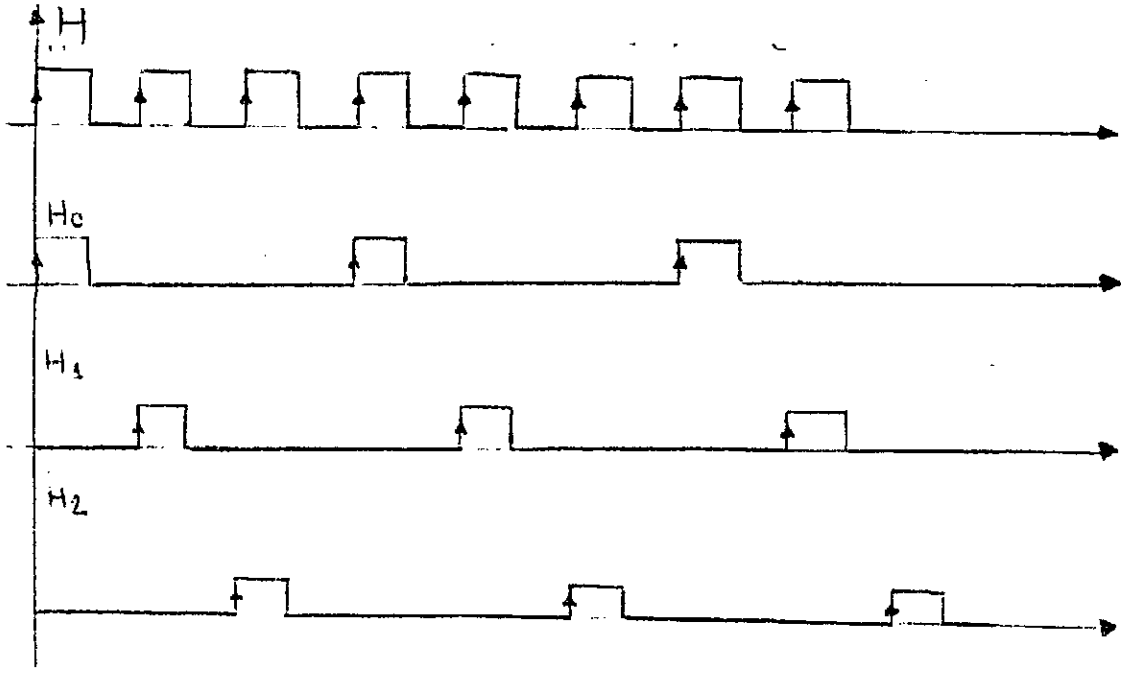


Figure (IV.28(a)): Résultat de notre stratégie de conception (Chronogrammes des signaux de commandes du filtre)

Il faut introduire le signal d'initialisation IN pour implémenter l'architecture digitale de l'unité de commande du filtre. Donc les résultats avec le logiciel XILINX donne la figure ci-dessous.

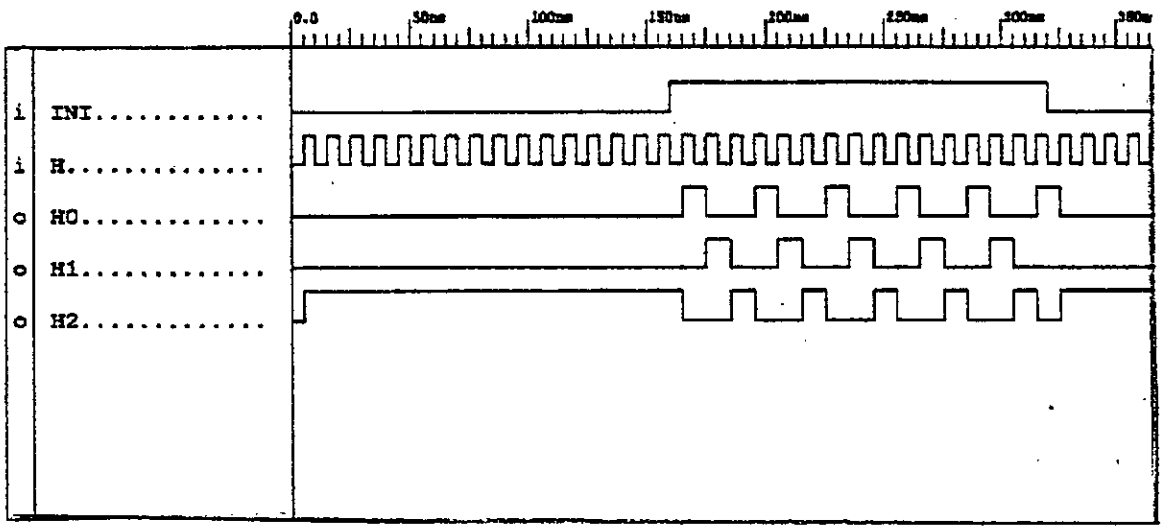


Figure (IV.28(b)): Chronogrammes des signaux de commandes pour le filtre récursif

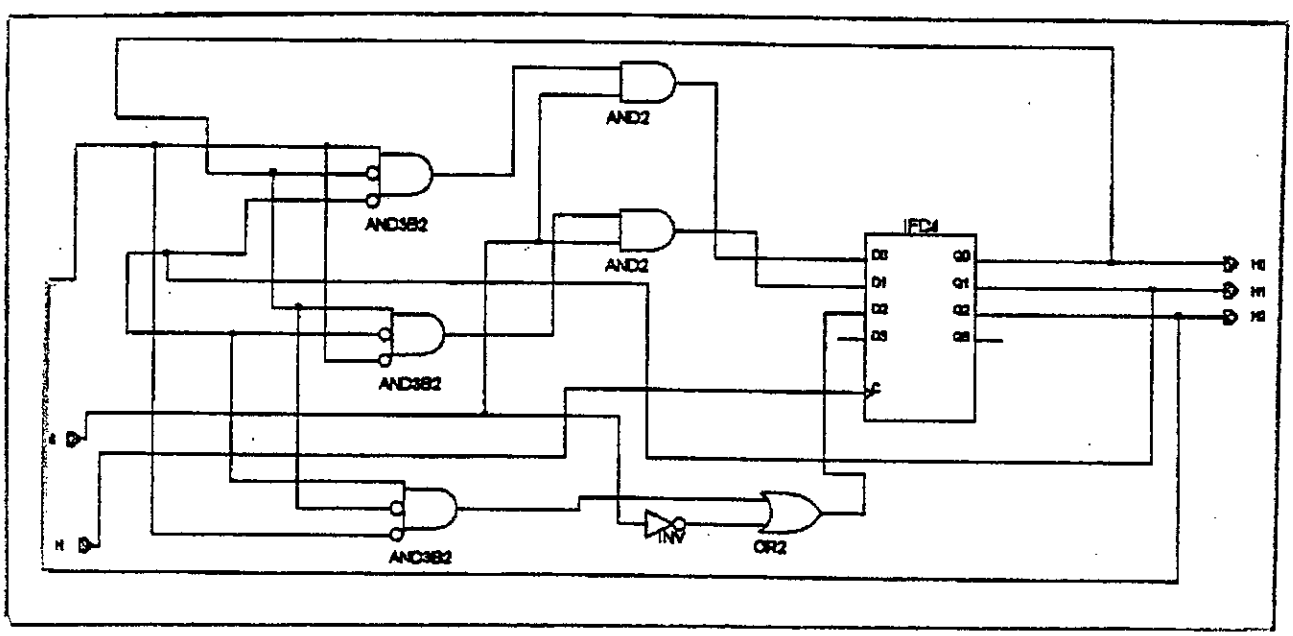
On peut déduire facilement les équations logiques et l'architecture de l'unité de commande du filtre, c'est la même procédure pour l'unité de commande du neurone.

Table de transition:

H0 H1 H2	H0+ H1+ H2+
1 0 0	0 1 0
0 1 0	0 0 1
0 0 1	1 0 0

On peut facilement trouver que les états futurs (les sorties):

$$H0+ = \overline{H0}\ \overline{H1}\ H2$$
$$H1+ = H0\ \overline{H1}\ \overline{H2}$$
$$H2+ = \overline{H0}\ H1\ \overline{H2}$$



Figure(IV.29): Représentation de l'unité de commande du filtre récursif en portes logiques

IV.3.4.3. Fonctionnment du système:

À t =0 :

- **Activation** du compteur par H0 (sélection de x(0)).
- **Sauvegarde :** R1=0, R3=0, R5=0, R7=0, R9=0.
- **Sauvegarde :** R2=0, R4=0, R6=0, R8=0, R10=0.

À t =T :

- **Activation par H1:** R1=x(0), R3=0, R5=0, R7=0, R9=0.
- **Sauvegarde :** R2=0, R4=0, R6=0, R8=0, R10=0.

À t =2T :

- **Activation par H2:** R2=x(0), R4=0, R6= a0x(0), R8=0, R10=0.
- **Sauvegarde :** R1=x(0), R3=0, R5=0, R7=0, R9=0.

À t =3T :

- **Activation** du compteur par H0 (sélection de x(1)).
- **Sauvegarde :** R2=x(0), R4=0, R6= a0x(0), R8=0, R10=0.
- **Sauvegarde :** R1=x(0), R3=0, R5=0, R7=0, R9=0.

À t =4T :

- **Activation par H1:** R1=x(1), R3=x(0), R5=a1x(0), R7=a0x(0), R9=0.
- **Sauvegarde :** R2=x(0), R4=0, R6=a0x(0), R8=0, R10=0.

À t =5T :

- **Activation parH2:** R2=x(1), R4=a2x(0), R6= a0x(1)+a1x(0), R8=a0x(0), R10=b1a0x(0).
- **Sauvegarde :** R1=x(1), R3=x(0), R5=a1x(0), R7=a0 x(0), R9=0.

À t =6T :

- **Activation** du compteur par H0 (sélection de x(2)).
- **Sauvegarde :** R1=x(1), R3=x(0), R5=a1x(0), R7=a0x(0), R9=0.
- **Sauvegarde :** R2=x(1), R4=a2x(0), R6=a0x(1)+a1x(0), R8=a0x(0), R10=b1a0x(0).

À t =7T :

- **Activation par H1:** R1=x(2), R3=x(1), R5=a1x(1)+a2x(0), R7=b1a0x(0)+a0x(1)+a1x(0), R9=b2a0x(0).
- **Sauvegarde :** R2=x(1), R4=a2x(0), R6=a0x(1)+a1x(0), R8=a0x(0), R10=b1a0x(0).

À t =8T :

- **Activation par H2:** R2=x(2), R4=a2x(1), R6= a0x(2)+a1x(1)+a2x(0)=**z(2)**, R8=b1a0x(0)+a1x(1)+a2x(0). R(10)=b1(b1a0x(0)+a0x(1)+a1x(0))+b2a0x(0).
- **Sauvegarde :** R1=x(2), R3=x(1), R5=a1x(1)+a2x(0), R7=b1a0x(0)+a0x(1)+a1x(0), R9=b2a0x(0).

À t =9T :

- **Activation** du compteur par H0 (sélection de x(3)).
- **Sauvegarde :** R2=x(2), R4=a2x(1), R6= a0x(2)+a1x(1)+a2x(0)=**z(2)**, R8=b1a0x(0)+a1x(1)+a2x(0) = **0**, R10=b1(b1a0x(0)+a0x(1)+a1x(0)+b2a0x(0)) = **0**.
- **Sauvegarde :** R1=x(2), R3=x(1), R5=a1x(1)+a2x(0), R7=b1a0x(0)+a0x(1)+a1x(0), R9=b2a0x(0).

À t =10T :

- **Activation par H1:** R1=x(3), R3=x(2), R5=a1x(2)+a2x(1),
  R7=**z(2)**, R9=b2(b1a0x(0)+a1x(1)+a2x(2))=**0.**
- **Sauvegarde :** R2=x(2), R4=a2x(1), R6=a0x(2)+a1x(1)+a2x(0),
  R8=b1a0x(0)+a1x(1)+a2x(1)=**0.**
  R10=b1(b1a0x(0)+a0x(1)+a1x(0)+b2a0x(0))=**0.**

À t =11T :

- **Activation parH2:** R2=x(3), R4=a2x(2), R6= a0x(3)+a1x(2)+a2x(1)=**z(3)**
  R8=**y(2)**,
  R10=b1z(2).
- **Sauvegarde :** R1=x(3), R3=x(2), R5=a1x(2)+a2x(1), R7= z(2)=**y(2)**,
  R9=0.

À t =12T :

- **Activation** du compteur par H0 (sélection de x(4)).
- **Sauvegarde :** R1=x(3), R3=x(2), R5=a1x(2)+a2x(1), R7=y(2),
  R9=0.
- **Sauvegarde :** R2=x(3), R4=a2x(2), R6=a0x(3)+a1x(2)+a2x(1)=**z(3),**
  R8=y(2), R10=b1y(2).

À t =13T :

- **Activation par H1:** R1=x(4), R3=x(3), R5=a1x(3)+a2x(2),
  R7=b1y(2)+z(3)=**y(3),** R9=b2y(2).
- **Sauvegarde :** R2=x(3), R4=a2x(2), R6=a0x(3)+a1x(2)+a2x(1)=**z(3),**
  R8=y(2) , R10=b1y(2).

À t =14T :

- **Activation par H2:** R2=x(4), R4=a2x(3), R6= a0x(4)+a1x(3)+a2x(2),
  R8=b1y(2)+z(3)=y(3), R10=b1(b1y(2)+z(3))+b2y(2).
- **Sauvegarde :** R1=x(4), R3=x(3), R5=a1x(3)+a2x(2), R7=b1y(2)+z(3)=**y(3)**,
  R9=b2y(2).

À t =15T :

- **Activation** du compteur par H0 (sélection de x(5)).
- **sauvegarde:** R2=x(4), R4=a2x(3), R6= a0x(4)+a1x(3)+a2x(2),
  R8=b1y(2)+z(3)=y(3), R10=b1(b1y(2)+z(3))+b2y(2).
- **Sauvegarde :** R1=x(4), R3=x(3), R5=a1x(3)+a2x(2), R7=b1y(2)+z(3)=**y(3)**,
  R9=b2y(2).

À t =16T :

- **Activation par H1:** R1=x(5), R3=x(4), R5=a1x(4)+a2x(3),
  R7=b1y(3)+b2y(2)+z(4)=**y(4)**,
  R9=b2(b1y(2)+z(3)).
- **Sauvegarde :** R2=x(4), R4=a2x(3), R6=a0x(4)+a1x(3)+a2x(2),
  R8=y(3), R10=b1(b1y(2)+z(3))+b2y(2).

À t =17T :

- **Activation parH2:** R2=x(5), R4=a2x(4), R6= a0x(5)+a1x(4)+a2x(3),
  R8=y(4), R10=b1y(4)+b2y(3).
  R10=b1a0x(0).
- **Sauvegarde :** R1=x(5), R3=x(4), R5=a1x(4)+a2x(3), R7=y(4),
  R9=b2(b1y(2)+z(3))=b2y(3).

À t =18T :

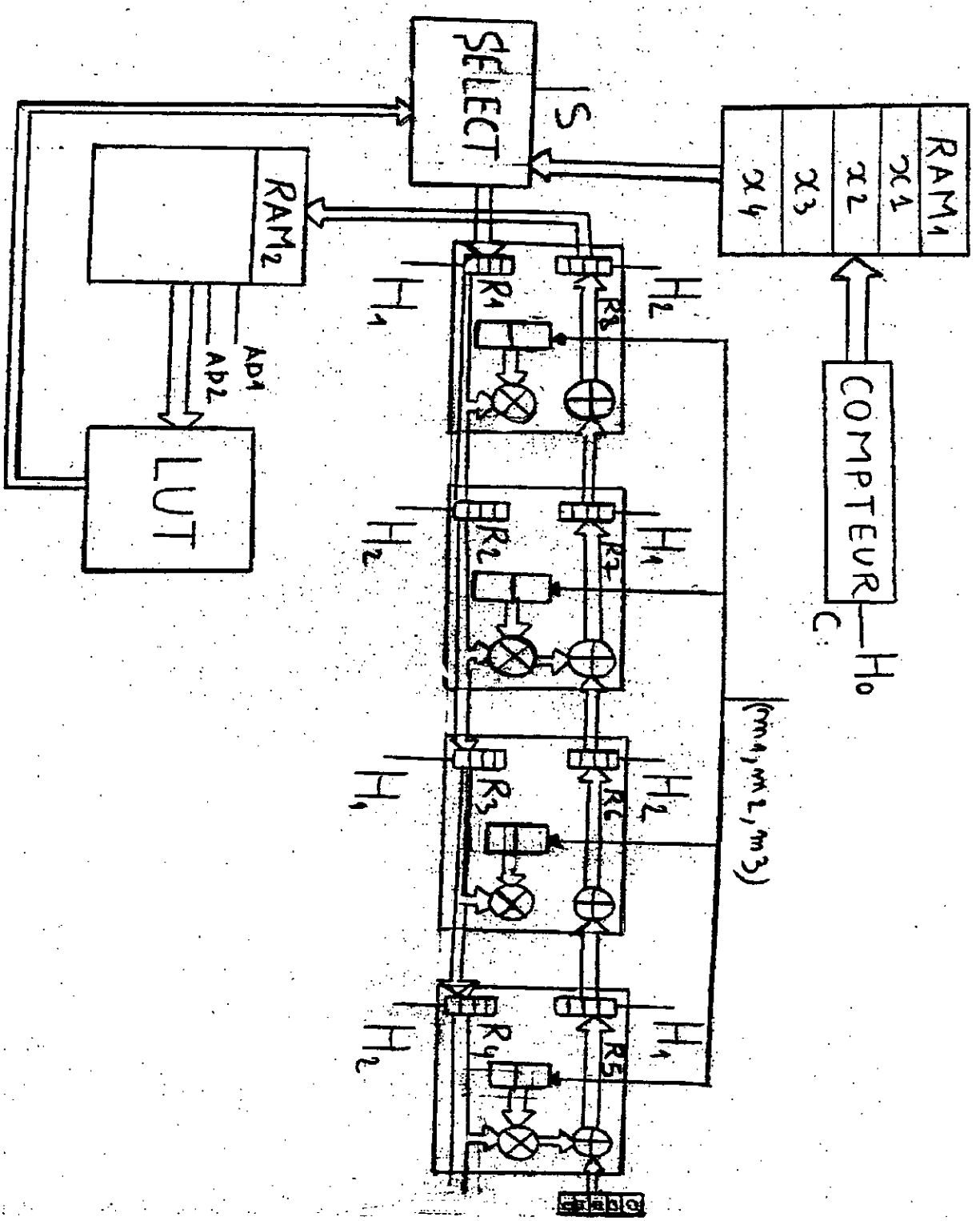
- **Activation** du compteur par H0 (sélection de x(6)).
- **sauvegarde:** R2=x(5), R4=a2x(4), R6= a0x(5)+a1x(4)+a2x(3)=z(5),
  R8=y(4), R10=b1y(4)+b2y(3).
- **Sauvegarde :** R1=x(4), R3=x(3), R5=a1x(3)+a2x(2), R7=b1y(2)+z(5)=**y(3)**,
  R9=b2y(2).

À t =19T :

- **Activation par H1:** R1=x(6), R3=x(5), R5=a1x(4),
  R7=b1y(4)+b2y(3)+z(5)=**y(5)**,
  R9=b2y(4).
- **Sauvegarde :** R2=x(5), R4=a2x(4), R6=**z(5),**
  R8=y(4), R10=b1y(4)+b2y(3).

IV.3.5. Implémentation de l'architecture systolique d'un RNA :

Finalement on trouve notre architecture systolique d'un réseau de neurones.



Figure(IV.30): Architcture systolique d'un réseau de neurones

IV.3.5.1.Fonctionnment du système:

Wij,Zij,Gij: Sont des synapses des couches d'entrées,cachées et de sorties respectivement tel que:

i= numéro de la sortie d'une couche donnée,

j= numéro du mac .

À t =0 :

- **Activation** du compteur par H0 (sélection de x(1)).
- **Sauvegarde :** R1=0, R3=0, R5=0, R7=0.
- **Sauvegarde :** R2=0, R4=0, R6=0, R8=0.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =T :

- **Activation par H1:** R1=x(1), R3=0, R5=0, R7=0.
- **Sauvegarde :** R2=0, R4=0, R6=0, R8=0.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =2T :

- **Activation par H2:** R2=x(1), R4=0, R6=0, R8=x(1)w11.
- **Sauvegarde :** R1=x(1), R3=0, R5=0, R7=0.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =3T :

- **Activation** du compteur par H0 (sélection de x(2)).
- **Sauvegarde :** R1=x(1), R3=0, R5=0, R7=0.
- **Sauvegarde :** R2=x(1), R4=0, R6=0, R8=x(1)w11.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =4T :

- **Activation par H1:** R1=x(2), R3=x(1), R5=0, R7=w12x(1).

- **Sauvegarde :** R2=x(1), R4=0, R6=0, R8=w11x(1).
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =5T :

- **Activation par H2:** R2=x(2), R4=x(1), R6=w13x(1), R8=x(1)w12+x(2)w11.
- **Sauvegarde :** R1=x(2), R3=x(1), R5=0, R7=w12x(1).
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =6T :

- **Activation** du compteur par H0 (sélection de x(3)).
- **Sauvegarde :** R1=x(2), R3=x(1), R5=0, R7=w12x(1).
- **Sauvegarde :** R2=x(2), R4=x(1), R6=w13x(1), R8=x(1)w12+x(2)w11.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =7T :

- **Activation par H1:** R1=x(3), R3=x(2), R5=w14x(1), R7=w12x(2)+w13x(1).
- **Sauvegarde :** R2=x(2), R4=x(1), R6=x(1)w13, R8=x(2)w11+x(1)w12.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =8T :

- **Activation par H2:** R2=x(3), R4=x(2), R6=w13x(2)+w14x(1),
  R8=x(3)w13+x(2)w12+x(1)w11.
- **Sauvegarde :** R1=x(3), R3=x(2), R5=w14x(1), R7=w12x(2)+w13x(1).
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =9T :

- **Activation** du compteur par H0 (sélection de x(4)).
- **Sauvegarde :** R1=x(3), R3=x(2), R5=w14x(1), R7=w12x(2)+w13x(1).
- **Sauvegarde :** R2=x(3), R4=x(2), R6=w13x(2)+w14x(1),
  R8=x(3)w13+x(2)w12+x(1)w11.
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =10T :

- **Activation par H1:** R1=x(4), R3=x(3), R5=x(2)w14,
  R7=w12x(3)+w13x(2)+w14x(1).
- **Sauvegarde :** R2=x(3), R4=x(2), R6=w13x(2)+w14x(1),
  R8=w11x(3)+w12x(2)+w13x(3).
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =11T :

- **Activation par H2:** R2=x(4), R4=x(3), R6=w13x(3)+w14x(2),
  R8=x(4)w11+x(2)w13+x(3)w12+x(1)w14=**y(0)**.
- **Sauvegarde :** R1=x(4), R3=x(3), R5=x(2)w14, R7=w12x(3)+w13x(2)+w11x(4)
- **S=0, adr1,adr2=0,0 ,m1,m2,m3=000**

À t =12T :

- **Activation** du compteur par H0 (sélection de x(1)).
- **Sauvegarde :** R1=x(4), R3=x(3), R5=x(2)w14,
  R7=w12x(3)+w13x(2)+w11x(4).
- **Sauvegarde :** R2=x(4), R4=x(3), R6=w13x(3)+w14x(2),
  R8=y(0).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =13T :

- **Activation par H1:** R1=x(1), R3=x(4), R5=w24x(3),
  R7=w21x(4)+w13x3+w14x(2).
- **Sauvegarde :** R2=x(4), R4=x(3), R6=w13x(3)+w14x(2),
  R8=w11x(4)+w12x(3)+w13x(3)+w14x(4)
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =14T :

- **Activation par H2:** R2=x(1), R4=x(4), R6=w23x(4)+w24x(3),
  R8=x(1)w21+w22x(4)+w13x(3)+w14x(2).
- **Sauvegarde :** R1=x(1), R3=x(4), R5=w24x(3),
  R7=w22x(4)+w13x(3)+w14x(2).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =15T :

- **Activation** du compteur par H0 (sélection de x(2)).
- **Sauvegarde :** R1=x(2), R3=x(1), R5=w24x(4),
  R7=w22x(1)+w23x(4)+w24x(3).
- **Sauvegarde :** R2=x(1), R4=x(4), R6=x(4)w23+w24x(3),
  R8=w21x(1)+w22x(4)+w13x(3)+w14x(2).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =16T :

- **Activation par H1:** R1=x(2), R3=x(1), R5=w24x(4),
  R7=w22x(1)+w23x(4)+w24x(3).
- **Sauvegarde :** R2=x(1), R4=x(4), R6=x(4)w23+x(3)w24,
  R8=w21x(1)+w22x(4)+w13x(3)+w14x(2).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =17T :

- **Activation par H2:** R2=x(2), R4=x(1), R6=w23x(1)+w24x(4),
  R8=w21x(2)+w22x(1)+w23x(4)+w24x(3)=**y(1)**.
- **Sauvegarde :** R1=x(2), R3=x(1), R5=w14x(4),
  R7=w22x(1)+w23x(4)+w24x(3).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=001**

À t =18T :

- **Activation** du compteur par H0 (sélection de x(3)).
- **Sauvegarde :** R1=x(2), R3=x(1), R5=w24x(4),
  R7=w22x(1)+w23x(4)+w24x(3).
- **Sauvegarde :** R2=x(2), R4=x(1), R6=x(1)w23+w24x(4),
  R8=w21x(2)+w22x(1)+w13x(4)+w14x(3).
- **S=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =19T :

- **Activation par H1:** R1=x(3), R3=x(2), R5=w34x(1),
  R7=w32x(2)+w23x(1)+w24x(4).
- **Sauvegarde :** R2=x(2), R4=x(1), R6=x(1)w23+x(4)w24,
  R8=w21x(2)+w22x(1)+w23x(4)+w24x(3).
- **S=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =20T :

- **Activation par H2:** R2=x(3), R4=x(2), R6=w33x(2)+w34x(1),
  R8=w31x(3)+w24x(4)+w32x(2)+w23x(1).
- **Sauvegarde :** R1=x(3), R3=x(2), R5=w34x(1),
  R7=w32x(2)+w23x(1)+w24x(4).
- **S=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =21T :

- **Activation** du compteur par H0 (sélection de x(4)).
- **Sauvegarde :** R1=x(3), R3=x(2), R5=w34x(1),
  R7=w32x(2)+w23x(1)+w24x(4).
- **Sauvegarde :** R2=x(3), R4=x(2), R6=w33x(2)+w34x(1),
  R8=w31x(3)+w24x(4)+w32x(2)+w23x(1).
- **s=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =22T :

- **Activation par H1:** R1=x(4), R3=x(3), R5=w34x(2)
  R7=w32x(3)+w33x(2)+w34x(1).
- **Sauvegarde :** R2=x(3), R4=x(2), R6=w32x(2)+w24x(1),
  R8=w32x(3)+w24x(4)+w32x(2)+w23x(1).
- **S=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =23T :

- **Activation par H2:** R2=x(4), R4=x(3), R6=w33x(3)+w34x(2),
  R8=x(4)w31+x(3)w32+x(2)w33+x(1)w34=**y(2)**.
- **Sauvegarde :** R1=x(4), R3=x(3), R5=w34x(2), R7=w32x(3)+w33x(2)+w34x(1)
- **S=0, adr1,adr2=1,0 ,m1,m2,m3=010**

À t =24T :

- **Activation** du compteur par H0 (sélection de x(1)).
- **Sauvegarde:** R2=x(4), R4=x(3), R6=w33x(3)+w34x(2),
  R8=x(4)w31+x(3)w32+x(2)w33+x(1)w34=**y(2)**.
- **Sauvegarde :** R1=x(4), R3=x(3), R5=w34x(2), R7=w32x(3)+w33x(2)+w34x(1)

S=0, adr1,adr2=1,1,m1,m2,m3=0,1,1

À t =25T :

- **Activation par H1:** R1=x(1), R3=x(4), R5=w44x(3),
  R7=w42x(4)+w33x(3)+w34x(2).
- **Sauvegarde :** R2=x(4), R4=x(3), R6=w33x(3)+w34x(2),
  R8=w31x(4)+w32x(3)+w33x(2)+w34x(1).
- **S=0, adr1,adr2=1,1 ,m1,m2,m3=0,1,1**

À t =26T :

- **Activation par H2:** R2=x(1), R4=x(4), R6=w43x(4)+w44x(3),
  R8=x(1)w41+x(4)w42+w33x(3)+w34x(2).
- **Sauvegarde :** R1=x(1), R3=x(4), R5=w34x(3),
  R7=w42x(4)+w33x(3)+w34x(2).
- **S=0, adr1,adr2=1,1 ,m1,m2,m3=0,1,1**

À t =27T :

- **Activation** du compteur par H0 (sélection de x(2)).
- **Sauvegarde :** R1=x(1), R3=x(4), R5=w34x(3),
  R7=w42x(4)+w43x(3)+w34x(2).
- **Sauvegarde :** R2=x(1), R4=x(4), R6=w43x(4)+w44x(3),
  R8=x(1)w42+x(4)w42+x(3)w33+x(2)w34.
- **S=0, adr1,adr2=1,1 ,m1,m2,m3=0,1,1**

À t =28T :

- **Activation par H1:** R1=x(2), R3=x(1), R5=w44x(1),
  R7= w42x(1)+w43x(4)+w44x(3).
- **Sauvegarde :** R2=x(1), R4=x(4), R6=x(4)w43+x(3)w44,
  R8=x(1)w41+x(4)w42+x(3)w33+x(2)w34.
- **S=0, adr1,adr2=1,1 ,m1,m2,m3=0,1,1**

À t =29T :

- **Activation par H2:** R2=x(2), R4=x(1), R6=w43x(1)+w44x(4),
  R8=x(2)w41+x(1)w42+x(4)w43+x(3)w44=**y(3)**.
- **Sauvegarde :** R1=x(2), R3=x(1), R5=w44x(4),
  R7=w42x(1)+w43x(4)+w44x(3).
- **S=0, adr1,adr2=1,1 ,m1,m2,m3=0,1,1**

À t =30T :

- **Activation** du compteur par H0 (sélection de y(0)).
- **Sauvegarde:** R2=x(2), R4=x(1), R6=w43x(1)+w44x(4),
  R8=x(2)w41+x(1)w42+x(4)w43+x(3)w44=**y(3)**.
- **Sauvegarde :** R1=x(2), R3=x(1), R5=w44x(4),
  R7=w42x(1)+w43x(4)+w44x(3).
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,0**

À t =31T :

- **Activation par H1:** R1=y(0), R3=x(2), R5=z14x(1),
  R7=z12x(2)+w43x(1)+w44x(4).
- **Sauvegarde :** R2=x(2), R4=x(1), R6=w43x(1)+w44x(4),
  R8=w41x(2)+w42x(1)+w43x(4)+w44x(3).
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,0**

À t =32T :

- **Activation par H2:** R2=y(0), R4=x(2), R6=z13x(2)+z14x(1),
  R8=y(0)z11+x(2)z12+x(1)w43+x(4)w44.
- **Sauvegarde :** R1=y(0), R3=x(2), R5=x(1)z14, R7=z12x(2)+w43x(1)+w44x(4)
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,0**

À t =33T :

- **Activation** du compteur par H0 (sélection de y(1)).
- **Sauvegarde :** R1=y(0), R3=x(2), R5=x(1)z14,
  R7=z12x(2)+w43x(1)+w44x(4).
- **Sauvegarde :** R2=y(0), R4=x(2), R6=z13x(2)+z14x(1),
  R8=y(0)z11+x(2)z12+x(1)w43+x(4)w44.
- **S=1, adr1,adr2=0,1 ,m1,m2,m3=1,0,0**

À t =34T :

- **Activation par H1:** R1=y(1), R3=y(0), R5=z14x(2),
  R7=z12y(0)+z13x(2)+z14x(1).
- **Sauvegarde :** R2=y(0), R4=x(2), R6=z13x(2)+z14x(1),
  R8=z11y(0)+z12x(2)+w43x(1)+w44x(4)
- **S=1, adr1,adr2=0,1 ,m1,m2,m3=1,0,0**

À t =35T :

- **Activation par H2:** R2=y(1), R4=y(0), R6=z13y(0)+z14x(2),
  R8=y(1)z11+z12y(0)+z13x(2)+z14x(1).
- **Sauvegarde :** R1=y(1), R3=y(0), R5=z14x(2),
  R7=z12y(0)+z13x(2)+z14x(1).
- **S=0, adr1,adr2=0,1 ,m1,m2,m3=1,0,0**

À t =36T :

- **Activation** du compteur par H0 (sélection de y(2)).
- **Sauvegarde :** R1=y(1), R3=y(0), R5=z14x(2),
  R7=z12y(0)+z13x(2)+z14x(1).
- **Sauvegarde :** R2=y(1), R4=y(0), R6=y(0)z13+z14x(2),
  R8=z11y(1)+z12x(4)+z13x(2)+z14x(1).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,0**

À t =37T :

- **Activation par H1:** R1=x(2), R3=x(1), R5=w24x(4),
  R7=w22x(1)+w23x(4)+w24x(3).
- **Sauvegarde :** R2=y(1), R4=y(0), R6=y(0)z13+x(2)z14,
  R8=z11y(1)+z12y(0)+z13x(2)+z14x(1).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,0**

À t =38T :

- **Activation par H2:** R2=y(2), R4=y(1), R6=z13y(1)+z14y(0),
  R8=z11y(2)+z12y(1)+z13y(0)+z14x(2).
- **Sauvegarde :** R1=y(2), R3=y(1), R5=z14y(0),
  R7=z12y(1)+z13y(0)+z14x(2).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,0**

À t =39T :

- **Activation** du compteur par H0 (sélection de y(3)).
- **sauvegarde:** R2=y(2), R4=y(1), R6=z13y(1)+z14y(0),
  R8=z11y(2)+z12y(1)+z13y(0)+z14x(2).
- **Sauvegarde :** R1=y(2), R3=y(1), R5=z14y(0),
  R7=z12y(1)+z13y(0)+z14x(2).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,0,0**

À t =40T :

- **Activation par H1:** R1=y(3), R3=y(2), R5=z14y(1),
  R7=z12y(2)+z13y(1)+z14y(0).
- **Sauvegarde :** R2=y(2), R4=y(1), R6=y(1)z13+y(0)z14,
  R8=z11y(0)+z12y(1)+z13y(0)+z14x(2).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,0,0**

À t =41T :

- **Activation par H2:** R2=y(3), R4=y(2), R6=z13y(2)+z14y(1),
  R8=z11y(3)+z12y(2)+z13y(1)+z14y(0)=**S0**.
- **Sauvegarde :** R1=y(3), R3=y(2), R5=z14y(1),
  R7=z12y(2)+z13y(1)+z14y(0).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,0,0**

À t =42T :

- **Activation** du compteur par H0 (sélection de y(0)).
- **sauvegarde:** R2=y(3), R4=y(2), R6=z13y(2)+z14y(1),
  R8=s0.
- **Sauvegarde :** R1=y(3), R3=y(2), R5=z14y(1),
  R7=z12y(2)+z13y(1)+z14y(0).
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,1**

À t =43T :

- **Activation par H1:** R1=y(0), R3=y(3), R5=z24y(2),
  R7=z22y(3)+z13y(2)+z14y(1).
- **Sauvegarde :** R2=y(3), R4=y(2), R6=y(2)z13+y(1)z14,
  R8=z11y(3)+z12y(2)+z13y(1)+z14y(0).
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,1**

À t =44T :

- **Activation par H2:** R2=y(0), R4=y(3), R6=z23y(3)+z24y(2),
  R8=z21y(0)+z22y(3)+z14y(1)+z13y(2).
- **Sauvegarde :** R1=y(0), R3=y(3), R5=z24y(2),
  R7=z22y(3)+z13y(2)+z14y(1).
- **S=1, adr1,adr2=0,0 ,m1,m2,m3=1,0,1**

À t =45T :

- **Activation** du compteur par H0 (sélection de y(1)).
- **Sauvegarde:** R2=y(0), R4=y(3), R6=z23y(3)+z24y(2),
  R8=z21y(0)+z22y(3)+z14y(1)+z13y(2).
- **Sauvegarde :** R1=y(0), R3=y(3), R5=z24y(2),
  R7=z22y(3)+z13y(2)+z14y(1).
- **S=1, adr1,adr2=0,1 ,m1,m2,m3=1,0,1**

À t =46T :

- **Activation par H1:** R1=y(1), R3=y(0), R5=z24y(3),
  R7=z22y(1)+z23y(3)+z14y(2).
- **Sauvegarde :** R2=y(0), R4=y(3), R6=y(3)z23+y(2)z24,
  R8=z21y(0)+z22y(3)+z14y(1)+z13y(2).
- **S=1, adr1,adr2=0,1 ,m1,m2,m3=1,0,1**

À t =47T :

- **Activation par H2:** R2=y(1), R4=y(0), R6=z23y(0)+z24y(3),
  R8=z21y(1)+z22y(0)+z23y(3)+z14y(2).
- **Sauvegarde :** R1=y(1), R3=y(0), R5=z24y(3),
  R7=z22y(0)+z23y(3)+z14y(2).
- **S=1, adr1,adr2=0,1 ,m1,m2,m3=1,0,1**

À t =48T :

- **Activation** du compteur par H0 (sélection de y(2)).
- **sauvegarde:** R2=y(1), R4=y(0), R6=z23y(0)+z24y(3),
  R8=z21y(1)+z22y(0)+z23y(3)+z14y(2).
- **Sauvegarde :** R1=y(1), R3=y(0), R5=z24y(3),
  R7=z22y(0)+z23y(3)+z14y(2).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,1**

À t =49T :

- **Activation par H1:** R1=y(2), R3=y(1), R5=z24y(0),
  R7=z22y(1)+z23y(0)+z24y(3).
- **Sauvegarde :** R2=y(1), R4=y(0), R6=y(0)z23+y(3)z24,
  R8=z21y(1)+z22y(0)+z23y(3)+z14y(2).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,1**

À t =50T :

- **Activation par H2:** R2=y(2), R4=y(1), R6=z23y(1)+z24y(0),
  R8=z21y(2)+z22y(1)+z23y(0)+z24y(3)=**S1**.
- **Sauvegarde :** R1=y(2), R3=y(1), R5=z24y(0),
  R7=z22y(1)+z23y(0)+z24y(3).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,0,1**

À t =51T :

- **Activation** du compteur par H0 (sélection de s0).
- **sauvegarde:** R2=y(2), R4=y(1), R6=z23y(1)+z24y(0),
  R8=s1.
- **Sauvegarde :** R1=y(2), R3=y(1), R5=z24y(0),
  R7=z22y(1)+z23y(0)+z24y(3).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,1,0**

À t =52T :

- **Activation par H1:** R1=s0, R3=y(2), R5=g14y(1),
  R7=g12y(2)+z23y(1)+z24y(0).
- **Sauvegarde :** R2=y(2), R4=y(1), R6=y(1)z23+y(0)z24,
  R8=z21y(2)+z22y(1)+z23y(0)+z24y(3).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,1,0**

À t =53T :

- **Activation par H2:** R2=s0, R4=y(2), R6=g13y(2)+g14y(1),
  R8=g11s0+g12y(2)+z23y(1)+z24y(0).
- **Sauvegarde :** R1=s0, R3=y(2), R5=g14y(1),
  R7=g12y(2)+z23y(1)+z24y(0).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,1,0**

À t =54T :

- **Activation** du compteur par H0 (sélection de s1).
- **sauvegarde:** R2=s0, R4=y(2), R6=g13y(2)+g14y(1),
  R8=g11s0+g12y(2)+z23y(1)+z24y(0).
- **Sauvegarde :** R1=y(2), R3=y(1), R5=z14y(0),
  R7=z12y(1)+z13y(0)+z14x(2).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,1,0**

À t =55T :

- **Activation par H1:** R1=s1, R3=s0, R5=g14y(2),
  R7=g12s0+z13y(2)+g14y(1).
- **Sauvegarde :** R2=s0, R4=y(2), R6=y(2)g13+y(1)g14,
  R8=g11s0+g12y(2)+z23y(1)+z24y(0).
- **S=1, adr1,adr2=1,1 ,m1,m2,m3=1,1,0**

À t =56T :

- **Activation par H2:** R2=s1, R4=s0, R6=g13s0+g14y(2),
  R8=g11s1+g12s0+g13y(2)+g14y(1)**=V(resultat final)**
  ou on met g13=g14=0.
- **Sauvegarde :** R1=s1, R3=s0, R5=g14y(2),
  R7=g12s0+g13y(2)+g14y(1).
- **S=1, adr1,adr2=1,0 ,m1,m2,m3=1,1,0**

IV.4. Conclusion :

Dans ce chapitre nous avons implémenté des architectures hardwares des réseaux de neurones et des filtres récursifs en utilisants des architectures séquentielle et systolique, on a constaté qu'on peut étendre facilement les dimensions de ces architectures, de telle sorte qu'on ajoute des blocs ou des MACs. De plus on fait une petite modification en ce qui concerne les architectures des unités de commandes, donc on peut dire que les unités de commandes sont presque inéchangeables.

CHAPITRE V
LES PERFORMANCES DES ARCHITECTURES SEQUENTIELLE ET SYSTOLIQUE

V.1.Introduction:

Dans ce paragraphe on va faire une comparaison entre les architectures séquentielle et systolique, afin de déduire leurs performances en ce qui concerne .

- La complexité architecturale.
- la rapidité.
- le parallélisme.

V.2.La complexité architecturale:

Les composants	Architecture séquentielle	Architecture systolique
Registres	7	8
Multiplieurs	7	4
Additionneurs	7	4
ROM	7	4
RAM	0	2
Lut	7	1
MUX(sélecteur)	3	1
Compteurs	1 (simple)	1
Unité de commande	1	1

Donc d'après le tableau on constate que l'architecture séquentielle est plus gourmande en surface par rapport à l'architecture systolique.

Remarque :

Si on veut étendre le réseau de neurone , les modifications à faire sont :

- **L'architecture séquentielle :** Ajouter des neurones (nœuds) supplémentaires , augmenter la taille du MUX ,ROM et des LUT.

- **L'architecture systolique :** On ajoute seulement des MACs en cascades en plus ,et modification de la taille des mémoires RAM1 et RAM2 ainsi que la lut.

Finalement on remarque que : Si la taille du neurone augmente alors l'architecture séquentielle devient de plus en plus gourmande en surface par rapport à l'architecture systolique.

V.3.La rapidité:

Selon les chronogrammes, les unités de commandes des deux architectures, on présente ce tableau

	Séquentielle	**systolique**
Temps de propagation du réseau	**24 T**	**56 T**

Ou T est la période de l'horloge qui gère tous le système (le rythme de fonctionnement).

Donc on peut dire que :

L'architecture séquentielle est plus rapide que l'architecture systolique.

V.3. Le parallélisme:

Le parallélisme dans l'architecture séquentielle est un parallélisme des nœuds pour chaque Couche, ou bien on parle du parallélisme des connexions synaptiques.

On parle de parallélisme au niveau d'une couche donnée , cependant il n'y a pas de parallélisme entre les couches.

Pour l'architecture systolique le parallélisme est la base de cette architecture car toutes les MAC exécutent des taches en même temps.

Donc on parle ici de parallélisme de tâche pour avoir une rapidité de calcul au niveau du réseau de neurone.

V.4. Conclusion:

Dans ce dernier chapitre on a entamé la notion de performances en complexité architecturale, rapidité et en parallélisme. Finalement il reste au concepteur pour implémenter son architecture de choisir le type d'architecture (séquentielle ou systolique) selon son besoin et sa stratégie de conception.

CONCLUSION GENERALE

CONCLUSION GENERALE

La réalisation de ce projet nous à permis d'enrichir nos connaissances sur les réseaux de neurones, systèmes experts et surtout sur les différentes architectures digitales d'implémentation des réseaux de neurones.

La partie software consiste à simuler l'étape d'apprentissage d'un réseau de neurones avec le logiciel MATLAB, cependant il y a plusieurs types d'algorithmes d'apprentissage. On a pris dans notre cas l'algorithme le plus utilisé, c'est l'algorithme de rétropropagation du gradient.

Différentes architectures hardwares des réseaux de neurones sont présentées, pour cela nous avons présenté des architectures partielles (des blocs) afin de construire l'architecture globale, la simulation de ces circuits est faite par le logiciel XILINX, en faisant sortir les chronogrammes des signaux de commandes. L'étape de l'implémentation réel du circuit (RNA) en technologie FPGA n'a pas été réaliser a cause de manque du matériel qui réalise le mapping, le placement et le routage du circuit RNA en FPGA.

Pour un réseau de neurones donné l'architecture séquentielle est plus rapide que l'architecture systolique, tandis que l'architecture séquentielle est plus gourmande en surface par rapport à l'architecture systolique.

Notre étude à montré que nous pouvons implémenter tout type de filtre, et de réseaux de neurones de dimensions quelconques tel que on ajoutant seulement des modules identiques (MACs) que se soit pour l'architecture séquentielle ou systolique.

Comme perspective à ce travail nous pensons à :

- maîtriser les concepts théoriques des réseaux de neurones, la logique floue et les algorithmes génétiques,
- comprendre les différentes architectures hardwares d'implémentation des réseaux de neurones, la logique floue et les algorithmes génétiques,
- concevoir un système expert destiné pour une tâche complexe en utilisant une architecture hybride.

BIBLIOGRAPHIE

[1] M. Bellanger, « traitement numérique du signal », Edition Masson ,1993.

[2] N. Izeboudjen, « conception et implémentation en FPGA d'un classificateur neuronal des arythmies cardiaques, ENP,1999,thèse de magistère.

[3] R. Boubartakh, « implémentation des réseaux de neurones sur un PC. Application à la reconnaissance d'images, ENP,1996,thèse d'ingénieur.

[4] S. Belaifa, «fiabilité des systèmes VLSI : test, testabilité et vérification des machines séquentielles, ENP,1998, thèse d'ingénieur.

[5] K. Laidi, « implémentation orientée hardware des réseaux de neurones artificiels : Application à la navigation d'un robot mobile autonome,ENP,1998,thèse de magistère

[6] A. Chohra, « planification et contrôle de la navigation des véhicules autonomes intelligents(VAI) en environnements dynamiques, ENP,1999,thèse d'Etat.

[7] N. Khaber, « Structures systoliques et simulation à l'aide du C parallèle », ENP,1994,thèse d'ingénieur

[8] M.Weinfeld, « Réseaux de neurones », techniques de l'ingénieur,1990.

[9] E. Davalo, « des réseaux de neurones », Edition Eyrolles, 1993.

[10] G. Mingwu, Yao-Wenchang « quasi-universal swich matrices for FPD design »,IEEE transaction on computers, VOL 48,n10,octobre 1999.

[11] P. Chow, J. Rose « the design of a SRAM-Based Field- Programmable Gate array :circuit design and layout, IEEE transactions on very large scale integration (VLSI) systèms,Vol 7, N3, september 1999.

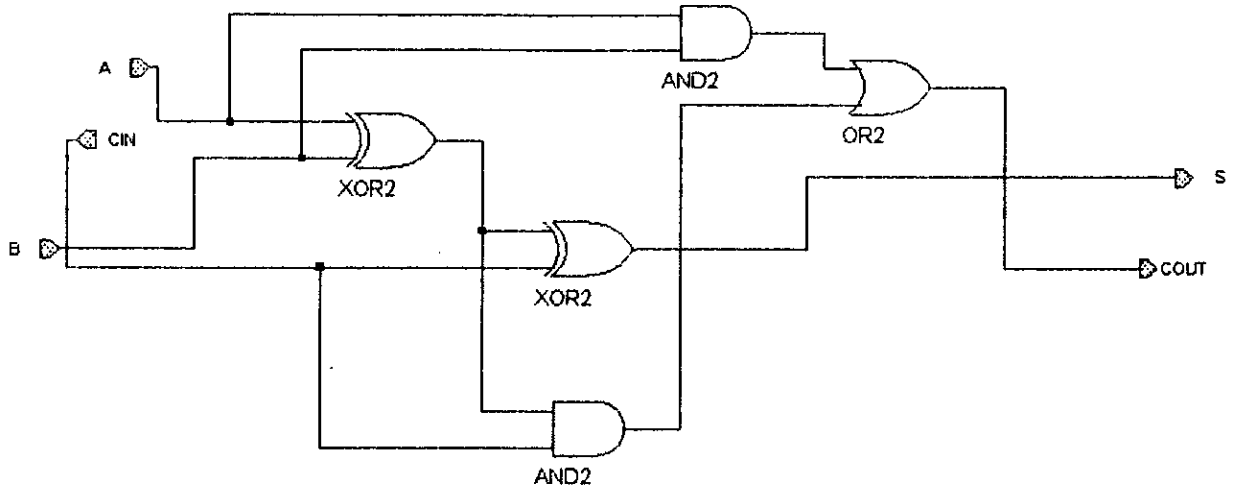
[12] M. Murakaw, S.Yoshizawa, « the GRD chip : Genetic Reconfiguration of DSP for neural network processing », IEEE transactions on computers, vol 48, N6, June 1996.

[13] H. Fellow, G.Rosendahl, « Competitive learning algorithmes and neurocomputer architecture »,IEEE transactions on computers, Vol 47, N8,August 1998.

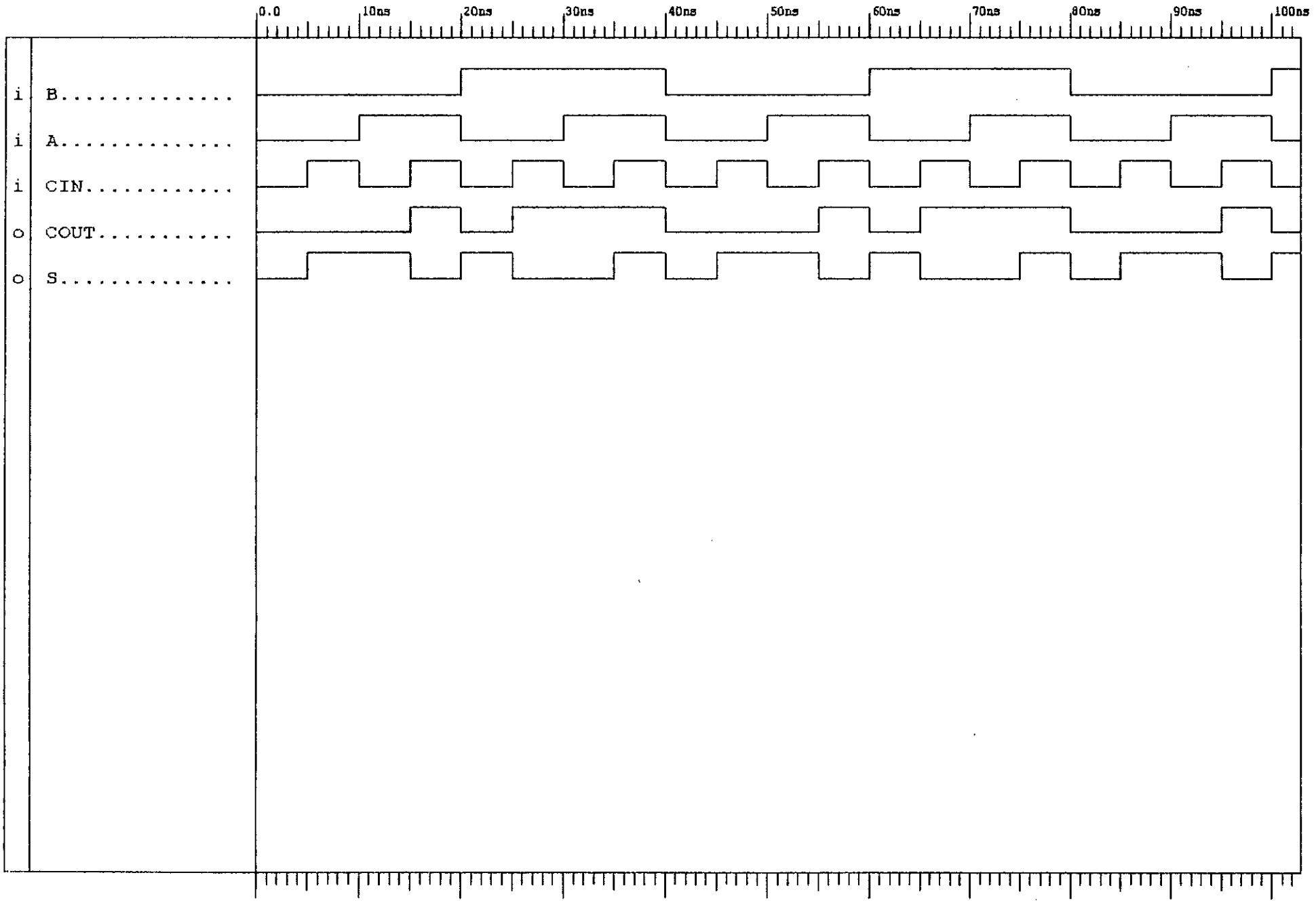
[14] N.Bahas, « conception logiciel d'un réseau de neurone multicouches pour la reconnaissance des chiffres 0 et 1 et conception matérielle et synthèse d'un neurone digitale, USTHB,1999,thèse d'ingénieur.

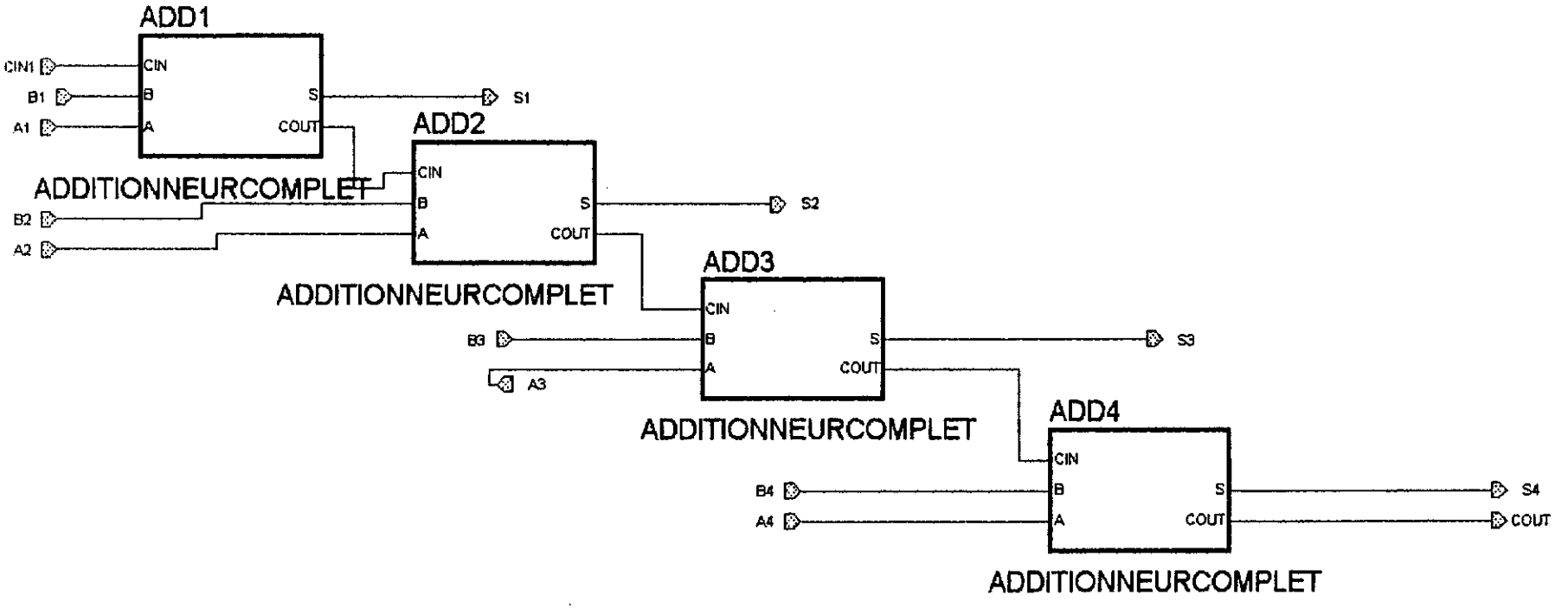
[15] M.Rebai, « implémentation d'un filtre recursif sur FPGA de Xilinx en mode ligne, USTHB,1999,thèse d'ingénieur.

ANNEXE

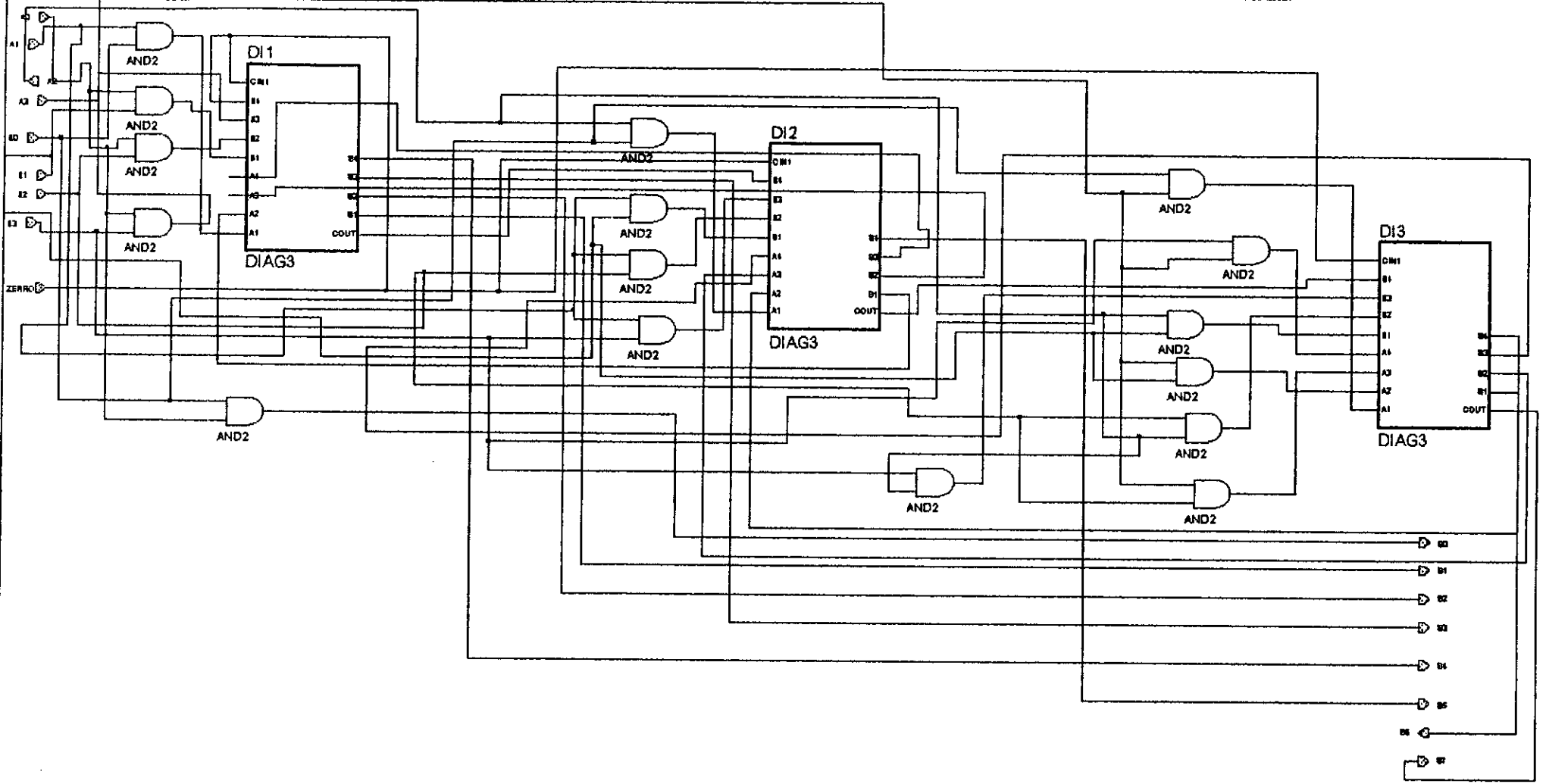


Xilinx Corporation	additioneur:
2100 Logic Drive	belaifa salah salim: additionneur complet
San Jose, CA 95124	Date: 14/11/2000

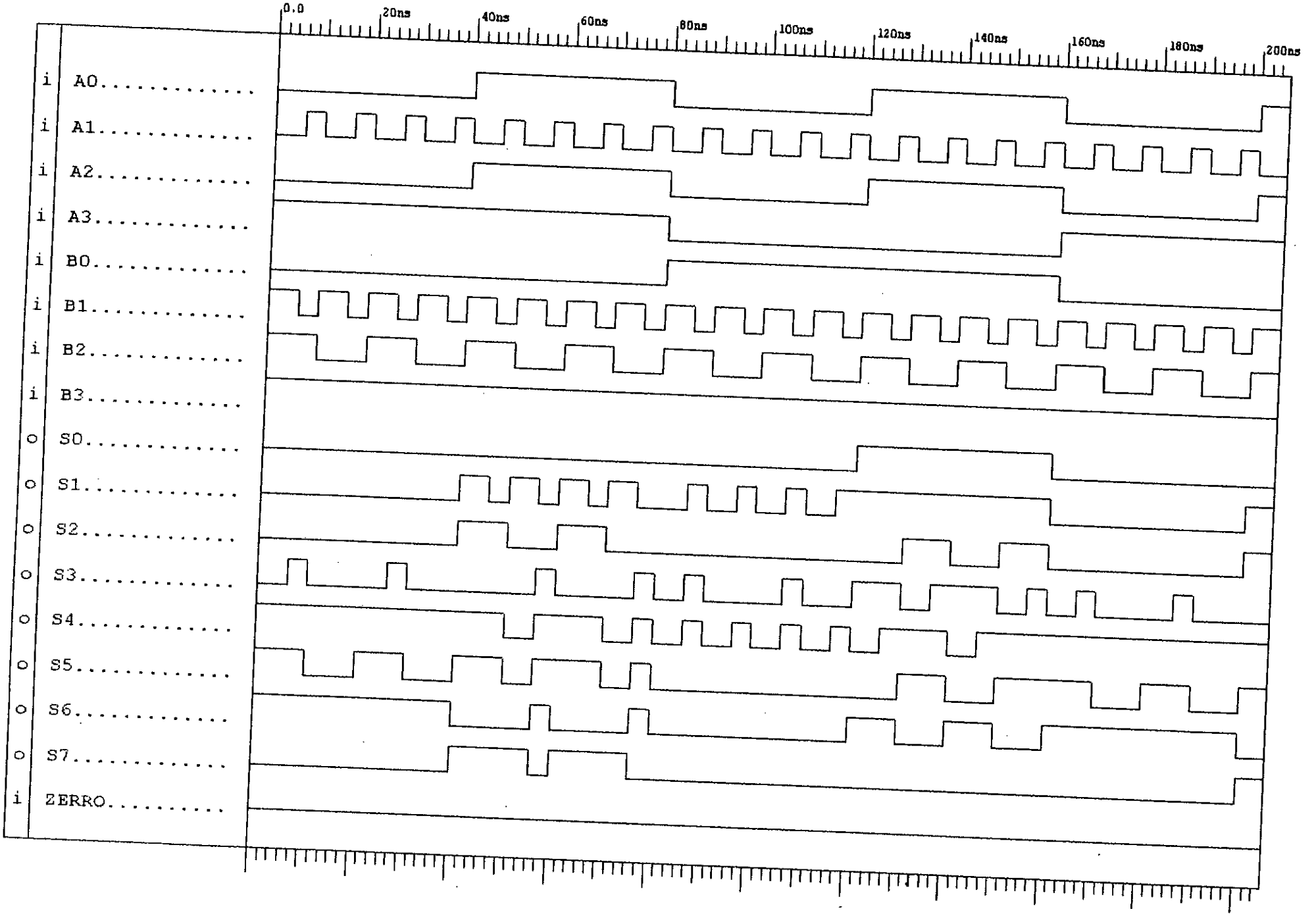

0.0
10ns
20ns
30ns
40ns
50ns
60ns
70ns
80ns
90ns
100ns
i B.....
i A.....
i CIN.....
o COUT.....
o S.....

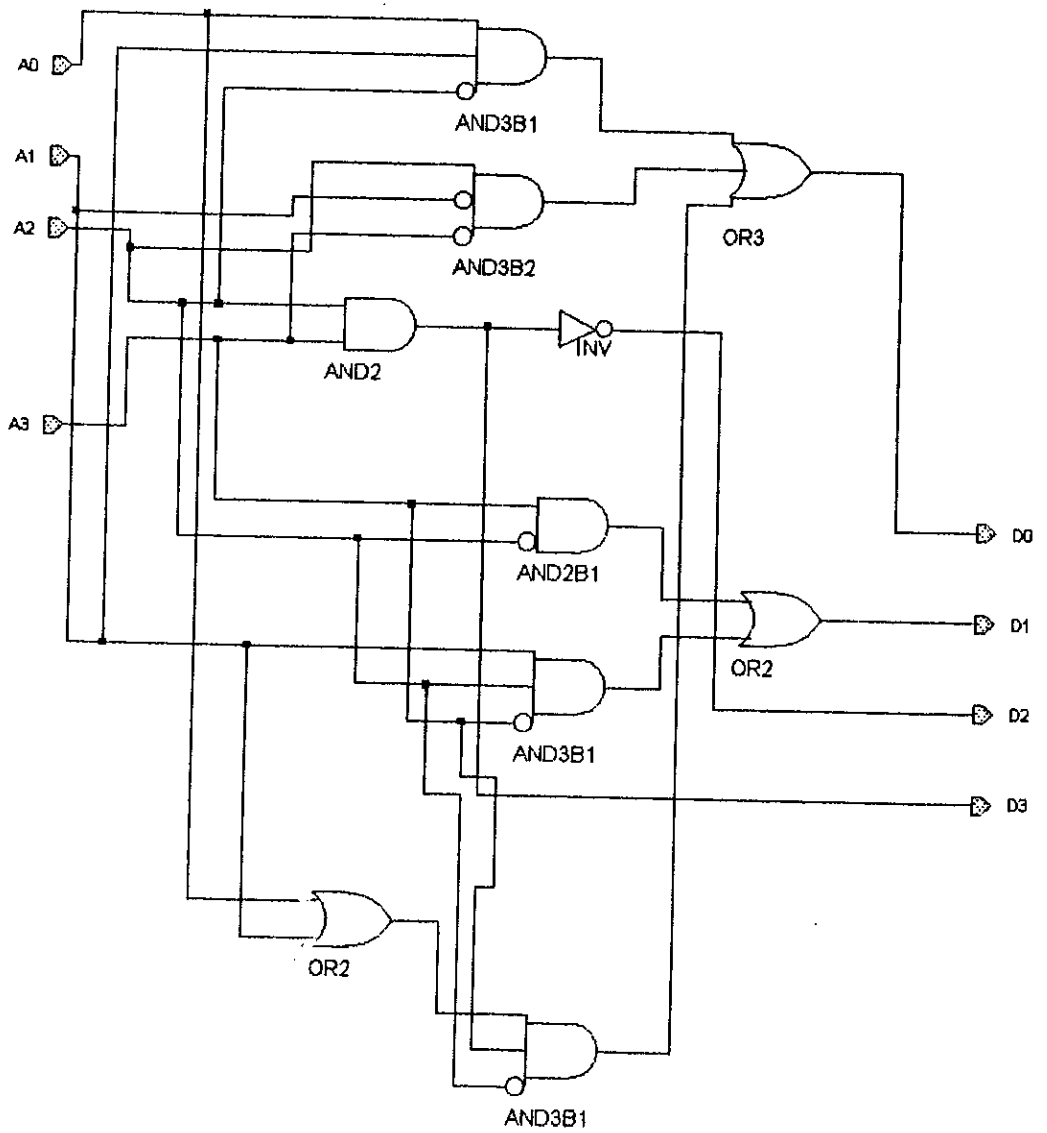


Xilinx Corporation	diag de multiplieur:
2100 Logic Drive	belaifa salah salim: diag3
San Jose, CA 95124	Date: 14/11/2000

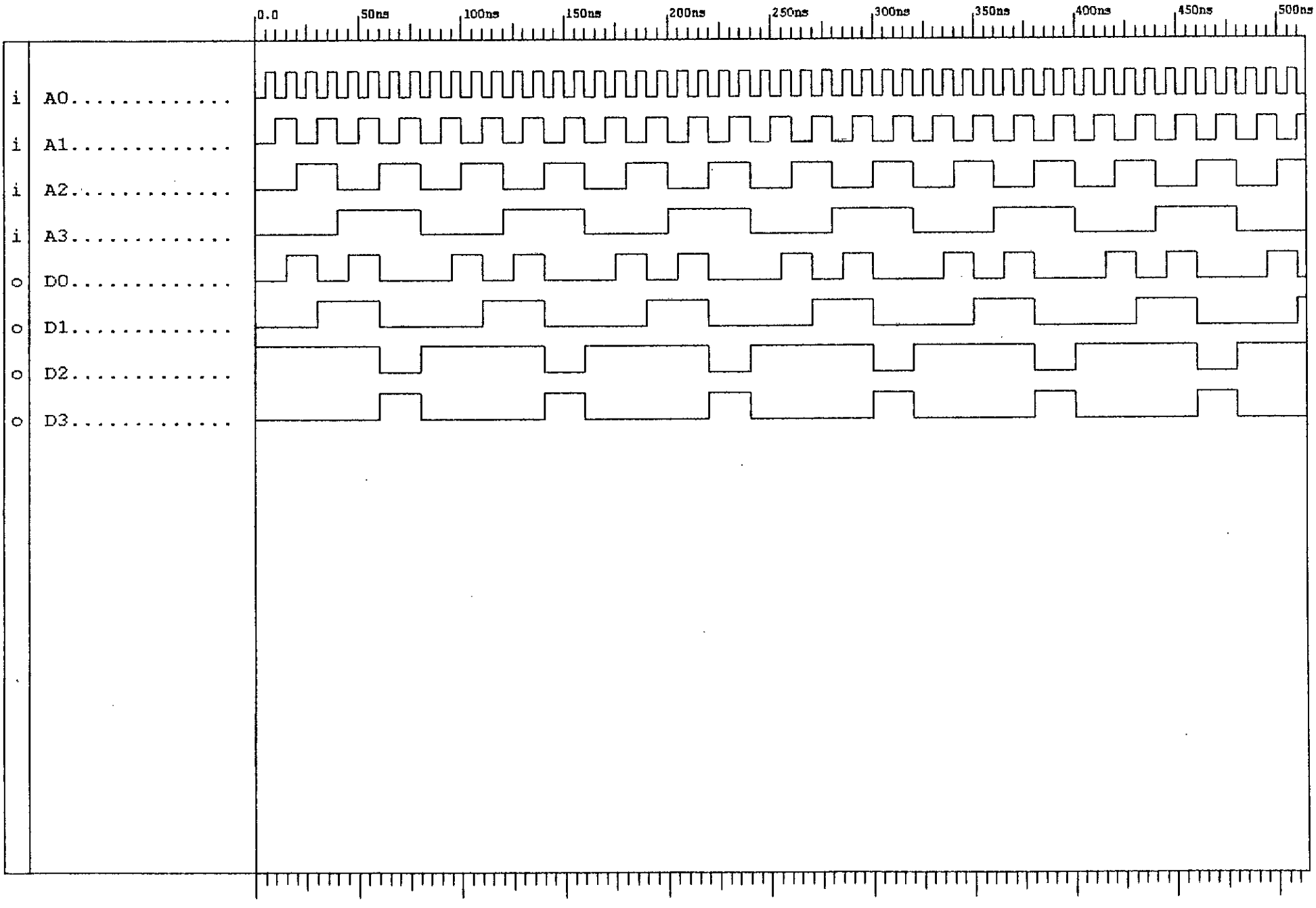


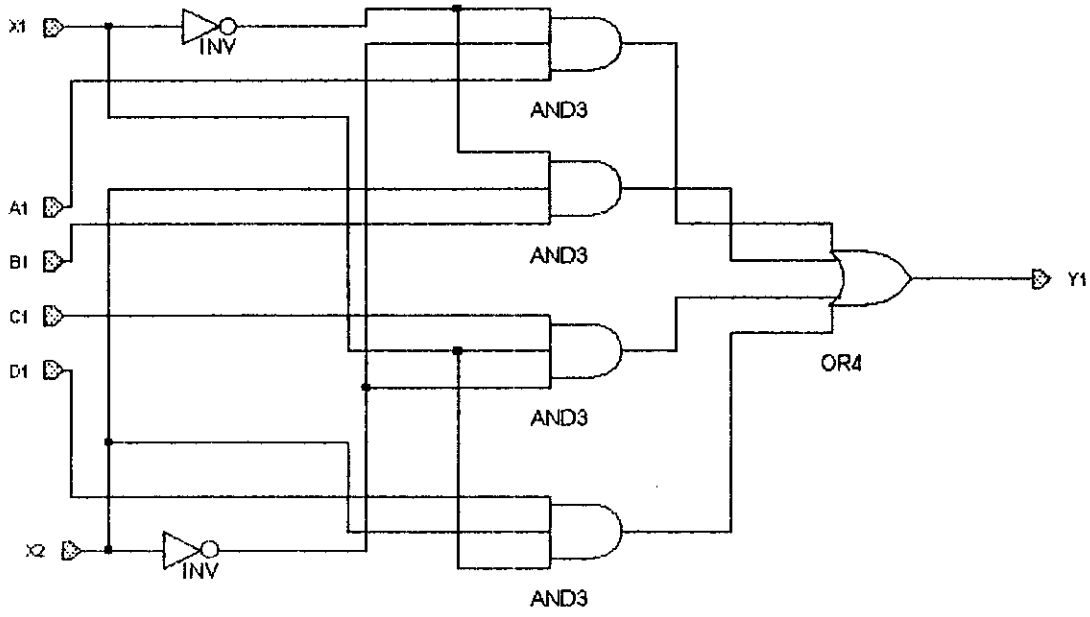
Xilinx Corporation	multiplieur 4 bits:
2100 Logic Drive	belaifa salah salim: multip4
San Jose, CA 95124	Date: 14/11/2000


0.0
20ns
40ns
60ns
80ns
100ns
120ns
140ns
160ns
180ns
200ns
i A0.....
i A1.....
i A2.....
i A3.....
i B0.....
i B1.....
i B2.....
i B3.....
o S0.....
o S1.....
o S2.....
o S3.....
o S4.....
o S5.....
o S6.....
o S7.....
i ZERRO.....

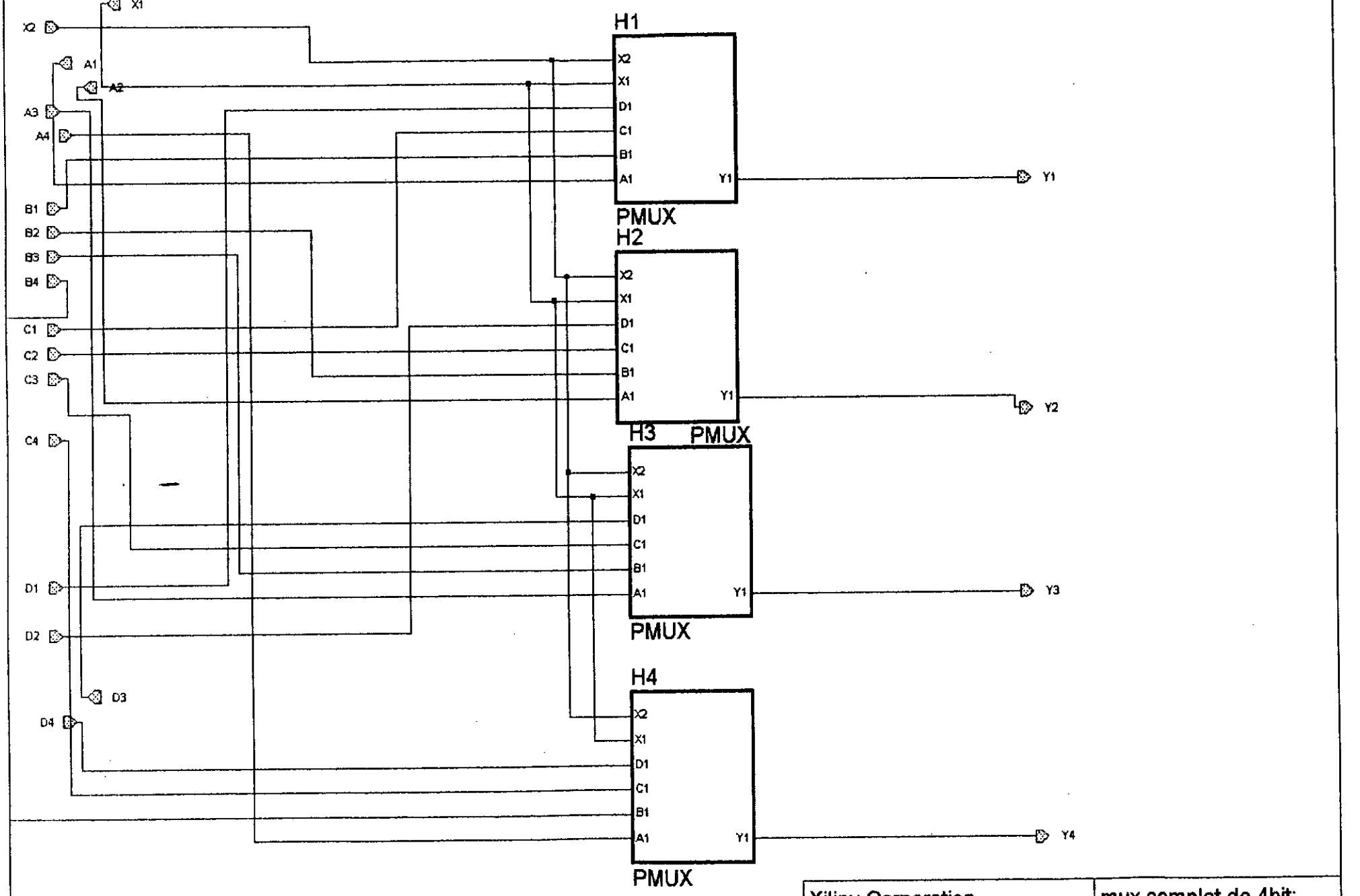


Xilinx Corporation	tabe de look-up: lut
2100 Logic Drive	belaifa salah salim: lut
San Jose, CA 95124	Date: 14/11/2000

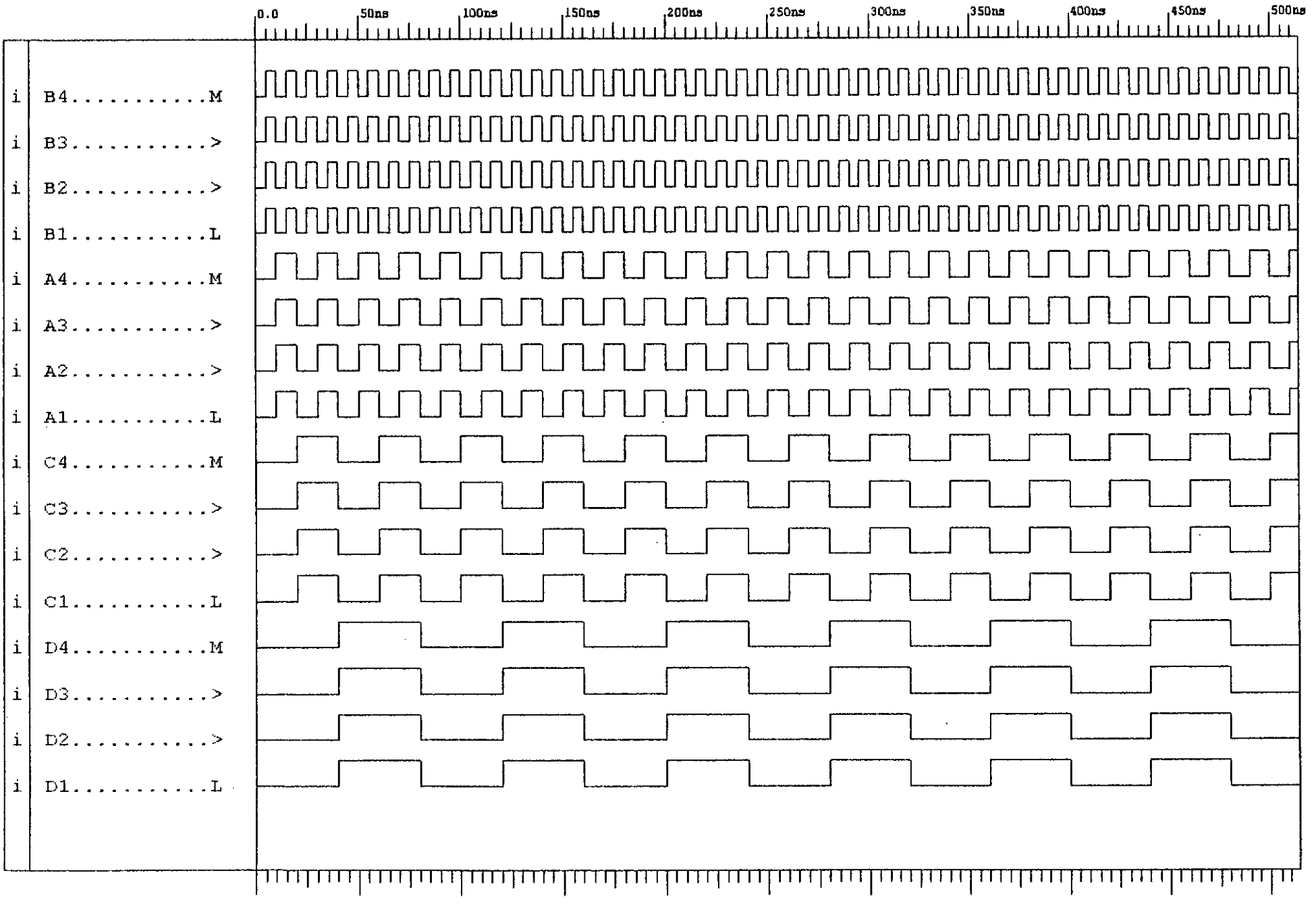

0.0
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
450ns
500ns
i A0.....
i A1.....
i A2.....
i A3.....
o D0.....
o D1.....
o D2.....
o D3.....

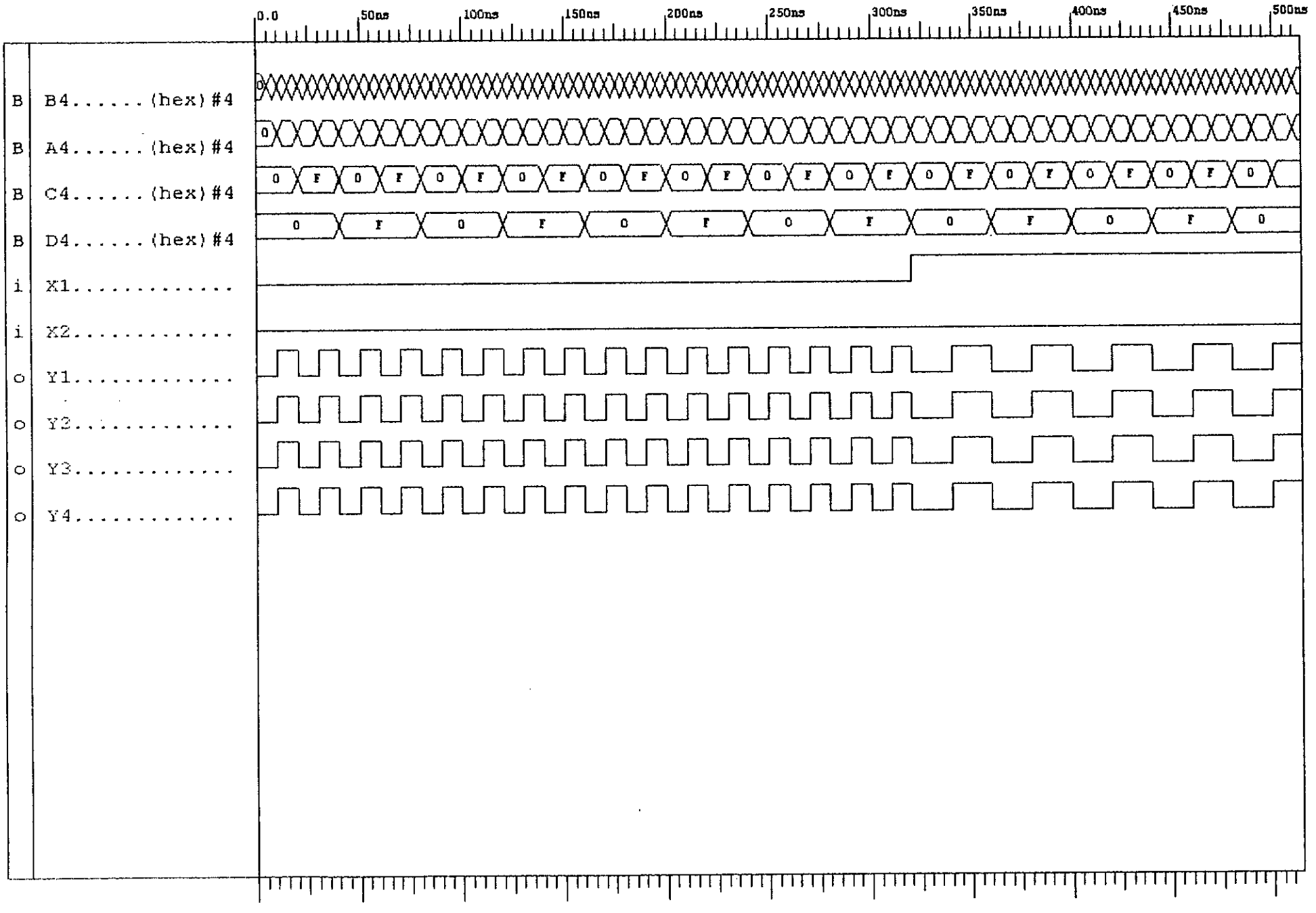


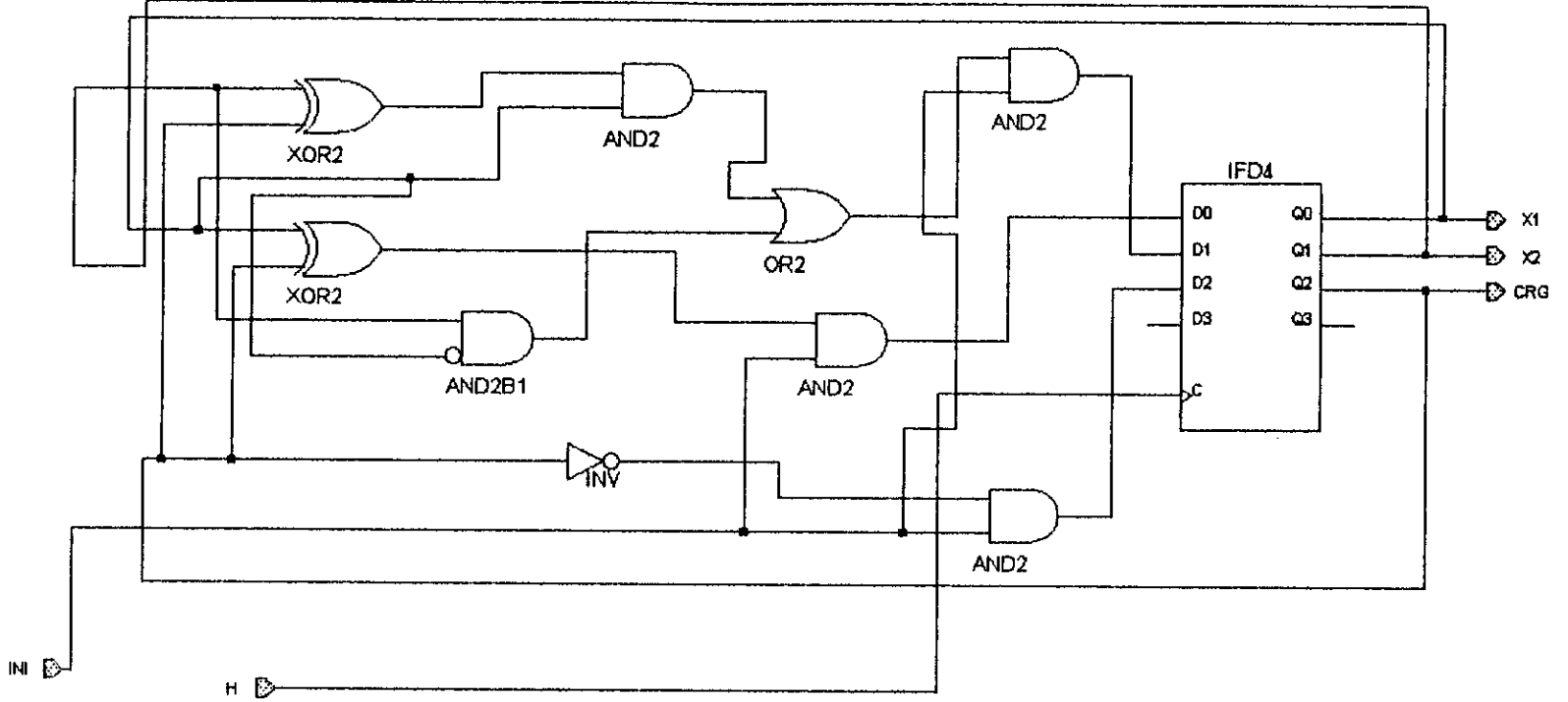
Xilinx Corporation	bloc du muliplexeur:
2100 Logic Drive	belaifa salah salim: pmux
San Jose, CA 95124	Date: 14/11/2000



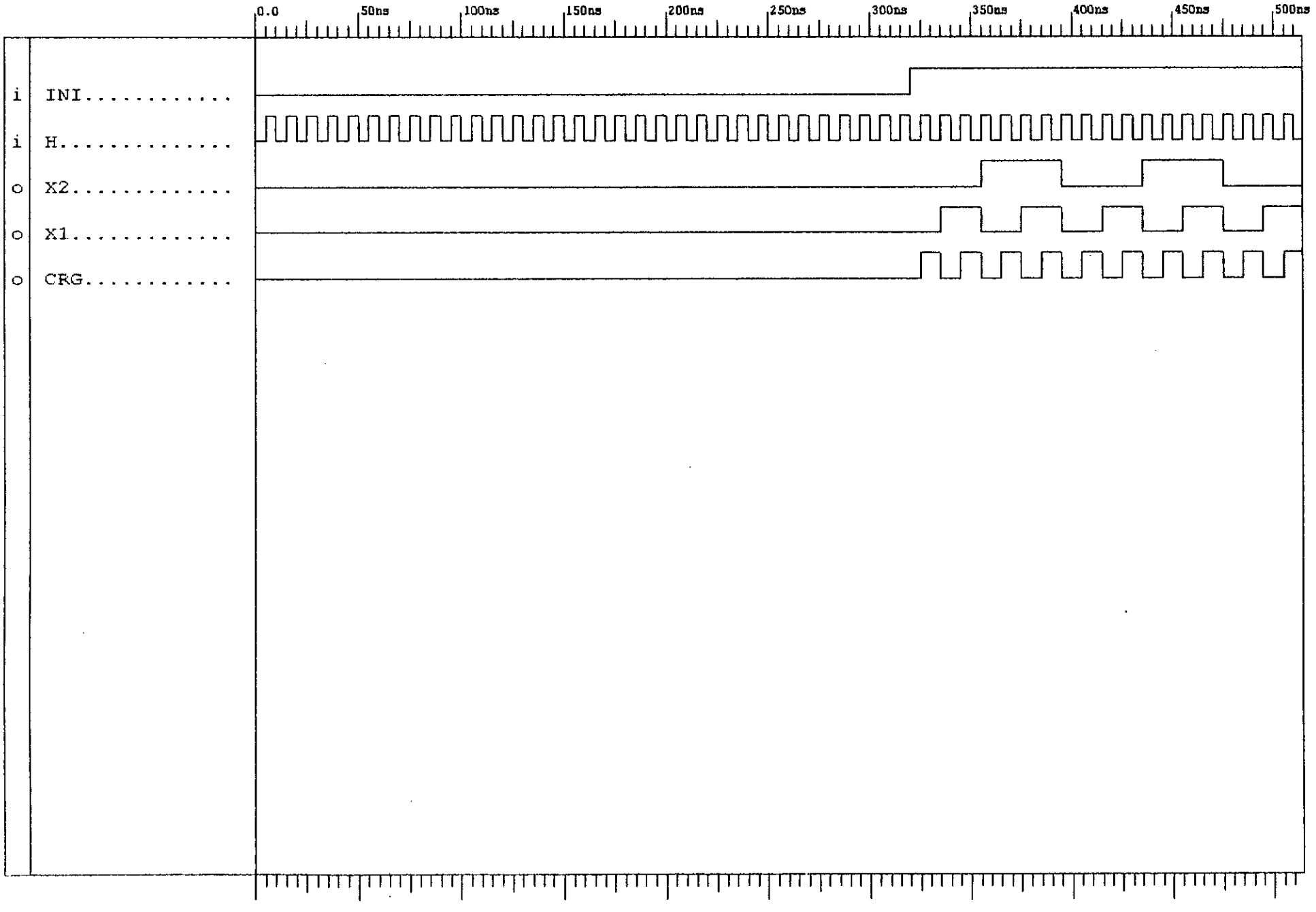
Xilinx Corporation	mux complet de 4bit:
2100 Logic Drive	belaifa salah salim: mux
San Jose, CA 95124	Date: 14/11/2000

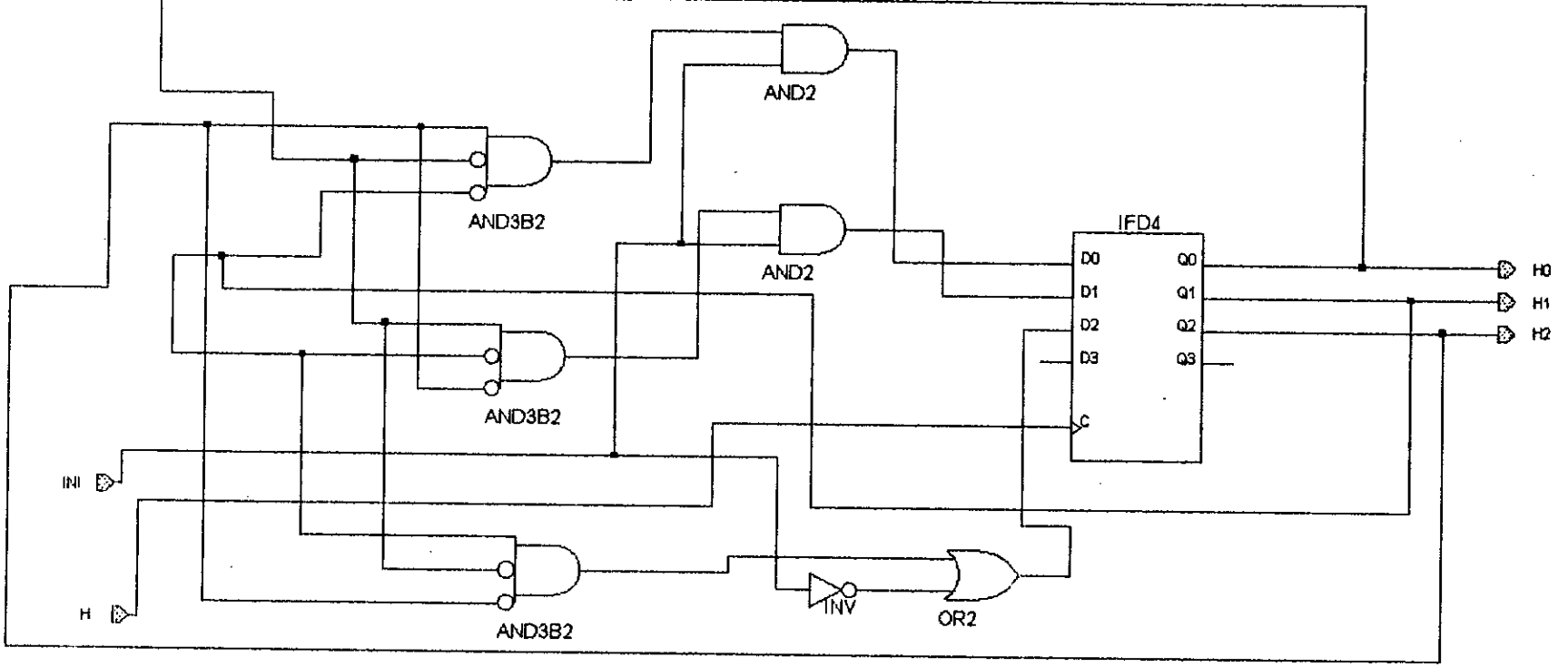

0.0
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
450ns
500ns
i B4.....M
i B3.....>
i B2.....>
i B1.....L
i A4.....M
i A3.....>
i A2.....>
i A1.....L
i C4.....M
i C3.....>
i C2.....>
i C1.....L
i D4.....M
i D3.....>
i D2.....>
i D1.....L


0.0
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
450ns
500ns
B B4.....(hex)#4
B A4.....(hex)#4
B C4.....(hex)#4
B D4.....(hex)#4
i X1.....
i X2.....
o Y1.....
o Y2.....
o Y3.....
o Y4.....

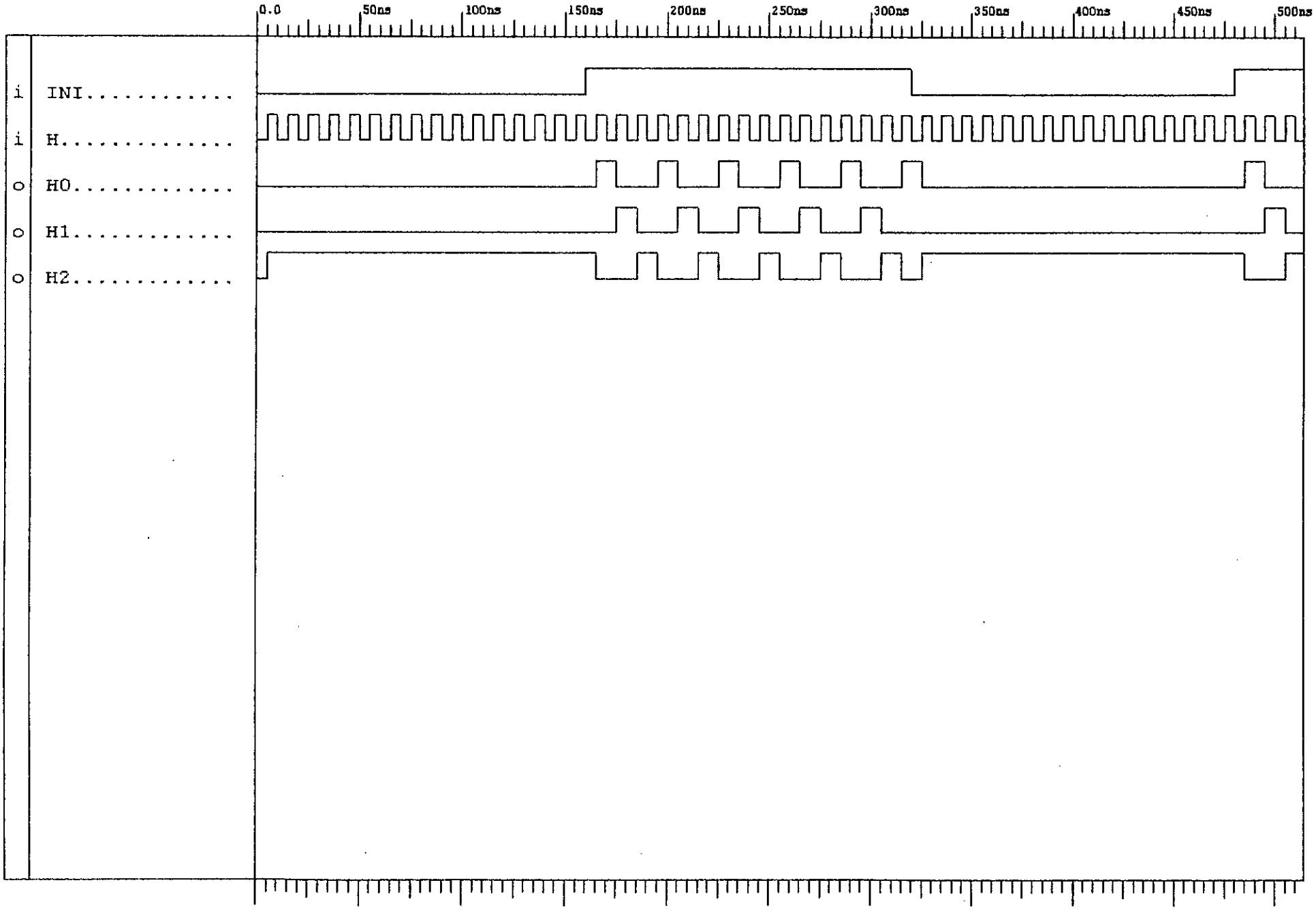


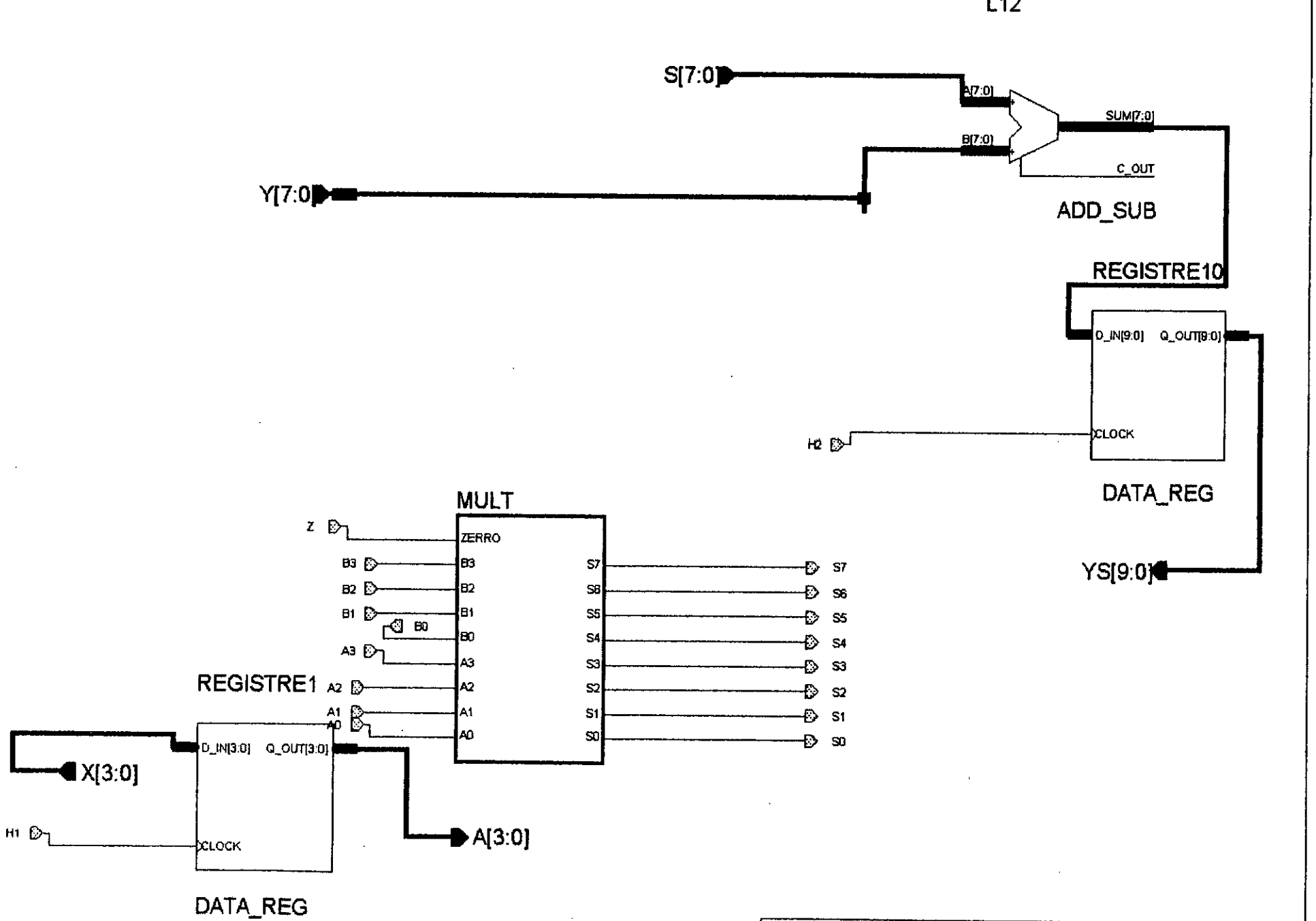
Xilinx Corporation	commande du neurone:
2100 Logic Drive	belaifa salah salim: comoneurone
San Jose, CA 95124	Date: 14/11/2000


0.0
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
450ns
500ns
i INI.....
i H.....
o X2.....
o X1.....
o CRG.....

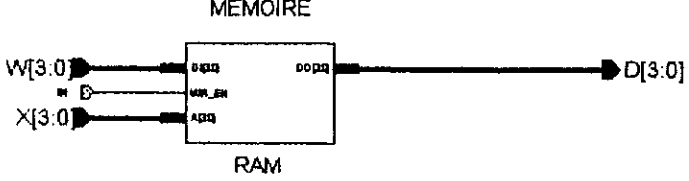


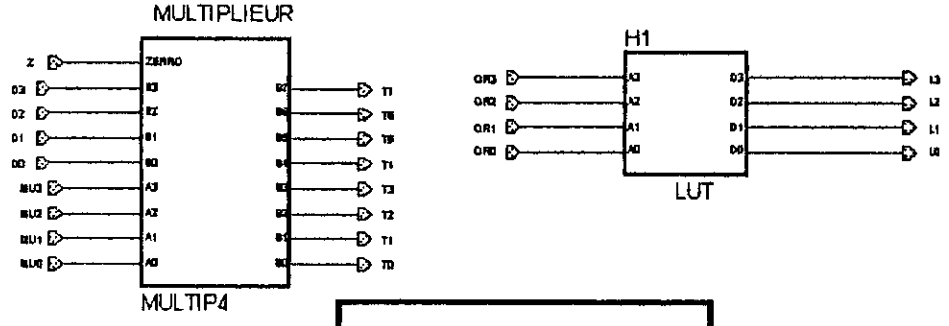
Xilinx Corporation	unité de commande:
2100 Logic Drive	belaifa salah salim: commande du filtre1
San Jose, CA 95124	Date: 14/11/100

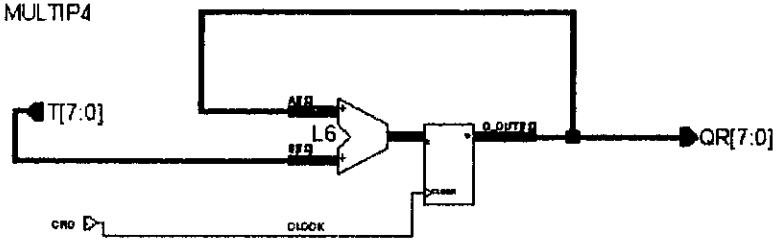

0.0
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
450ns
500ns
i INI.....
i H.....
o H0.....
o H1.....
o H2.....



Xilinx Corporation	schéma d'un MAC:
2100 Logic Drive	belaifa salah salim: mac1
San Jose, CA 95124	Date: 14/11/2000

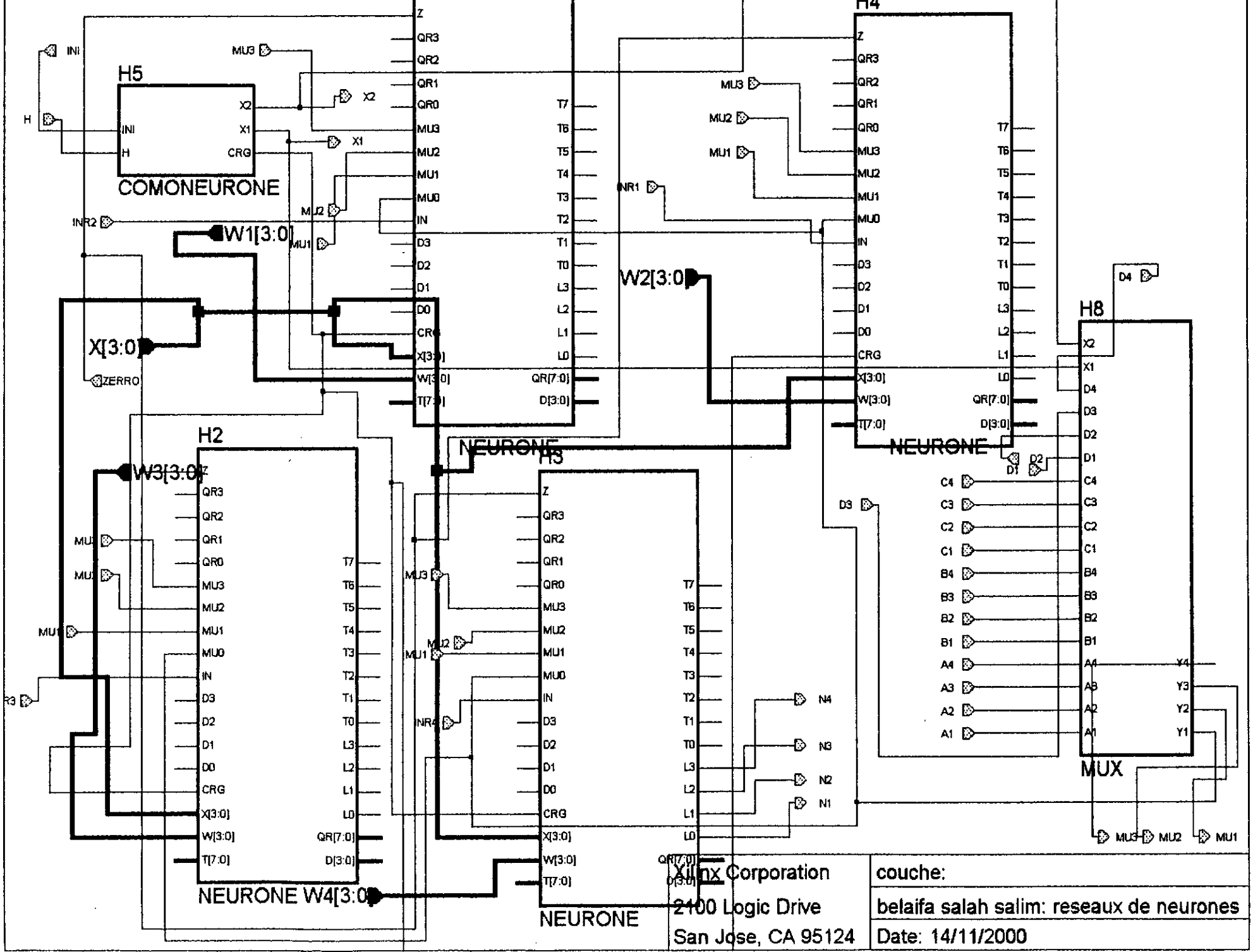


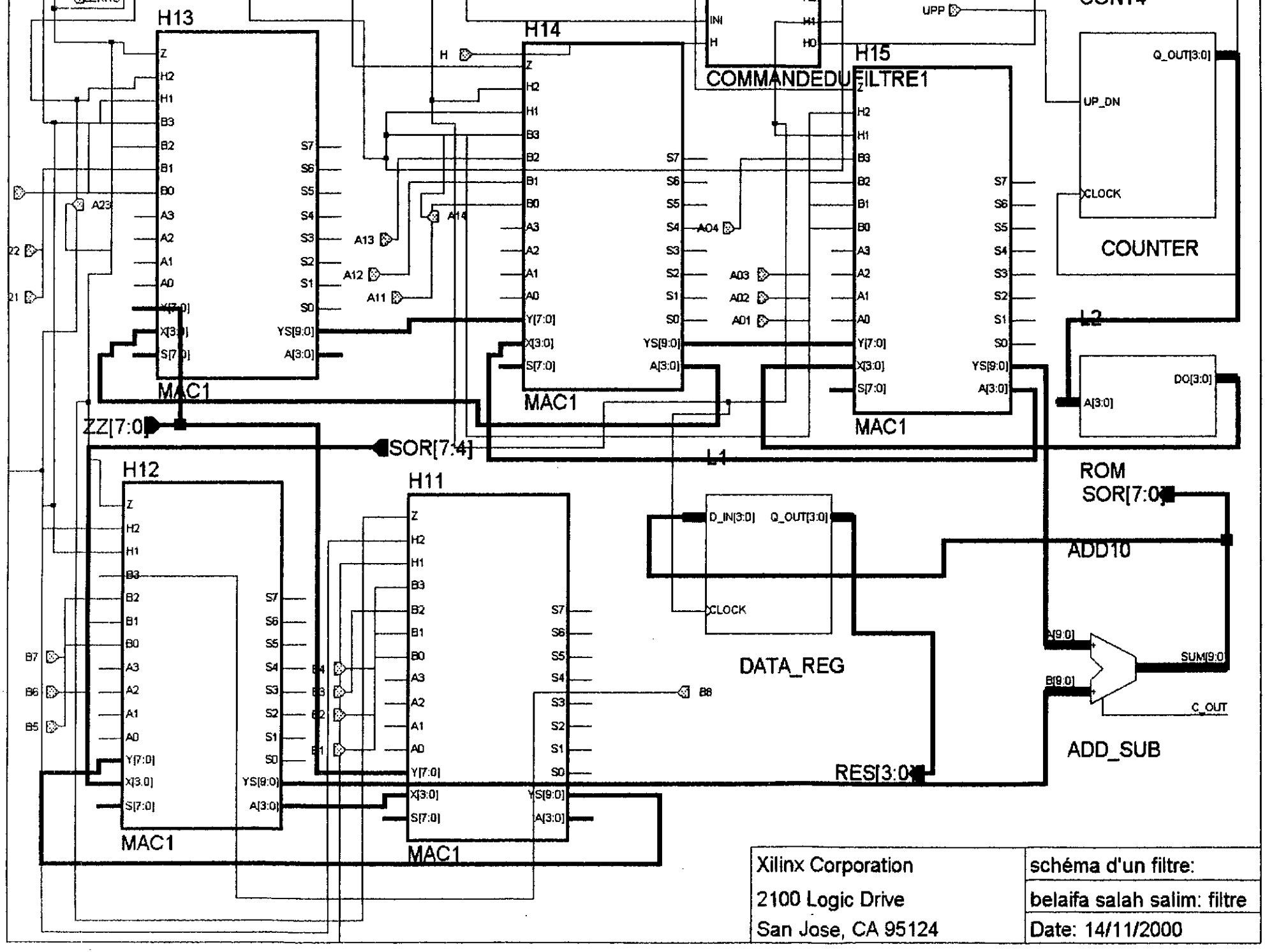


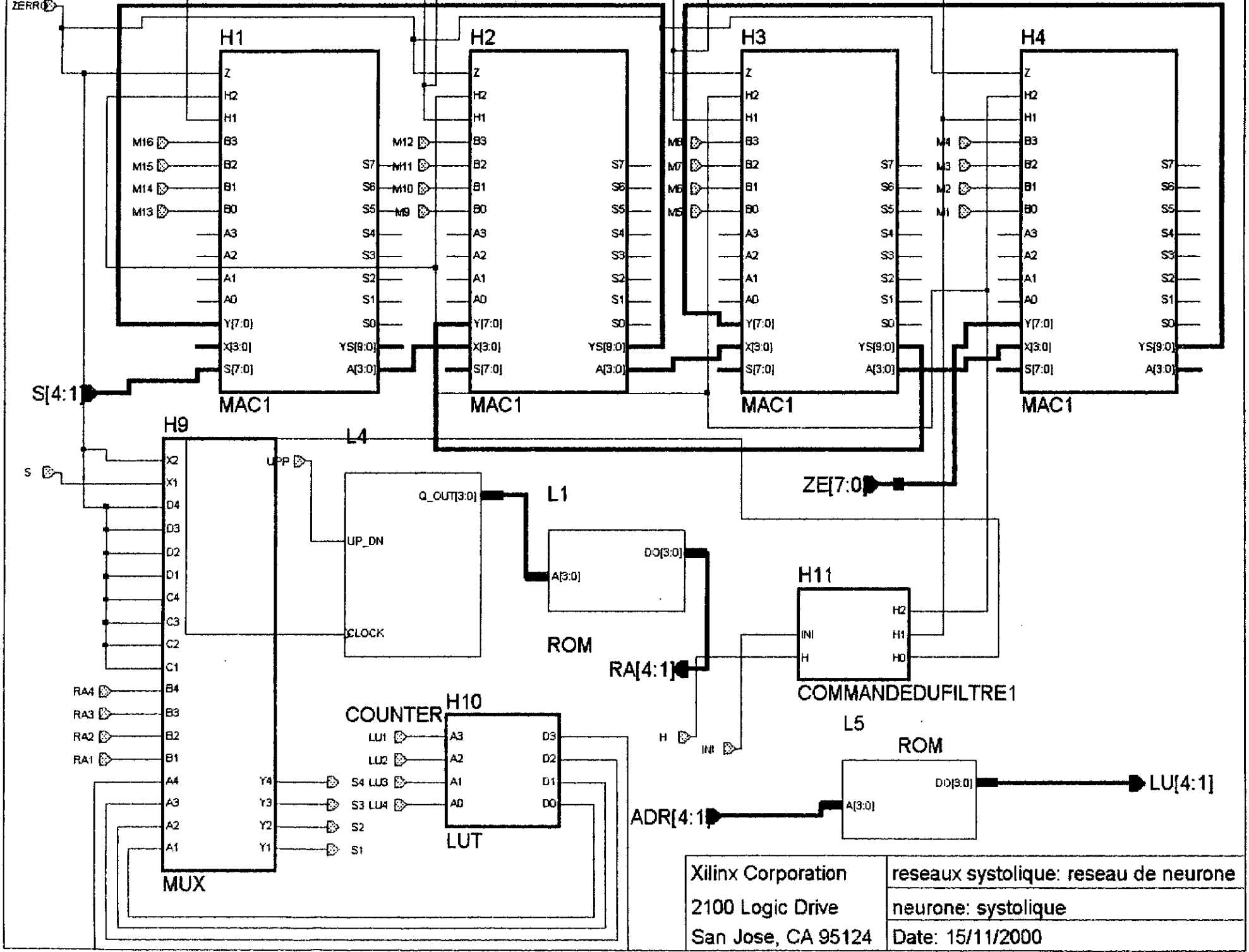


ADD_SUB

Xilinx Corporation	schéma d'un neurone: neurone
2100 Logic Drive	belaifa salah salim: NEURONE2
San Jose, CA 95124	Date: 14/11/2000


H5
COMONEURONE
INI
H
MU3
X2
X1
MU2
MU1
MU0
IN
INR2
W1[3:0]
X[3:0]
ZERRO
Z
QR3
QR2
QR1
QR0
D3
D2
D1
D0
CRG
W[3:0]
T[7:0]
T7
T6
T5
T4
T3
T2
T1
T0
L3
L2
L1
L0
QR[7:0]
D[3:0]
H4
NEURONE
INR1
W2[3:0]
H8
MUX
D4
C4
C3
C2
C1
B4
B3
B2
B1
A4
A3
A2
A1
Y4
Y3
Y2
Y1
H2
NEURONE W4[3:0]
W3[3:0]
H3
NEURONE
INR4
N4
N3
N2
N1
Xilinx Corporation
2100 Logic Drive
San Jose, CA 95124
couche:
belaifa salah salim: reseaux de neurones
Date: 14/11/2000


H13
H14
H15
COMMANDEDUFILTRE1
MAC1
MAC1
MAC1
ZZ[7:0]
SOR[7:4]
L1
L2
H12
H11
COUNTER
Q_OUT[3:0]
UP_DN
CLOCK
ROM
SOR[7:0]
DO[3:0]
A[3:0]
ADD10
D_IN[3:0]
Q_OUT[3:0]
CLOCK
DATA_REG
SUM[9:0]
C_OUT
ADD_SUB
RES[3:0]
Xilinx Corporation
2100 Logic Drive
San Jose, CA 95124
schéma d'un filtre:
belaifa salah salim: filtre
Date: 14/11/2000


ZERR
H1
H2
H3
H4
Z
H2
H1
B3
B2
B1
B0
A3
A2
A1
A0
Y[7:0]
X[3:0]
S[7:0]
S7
S6
S5
S4
S3
S2
S1
S0
YS[9:0]
A[3:0]
M16
M15
M14
M13
M12
M11
M10
M9
M8
M7
M6
M5
M4
M3
M2
M1
MAC1
S[4:1]
H9
L4
S
X2
X1
D4
D3
D2
D1
C4
C3
C2
C1
UPP
UP_DN
CLOCK
Q_OUT[3:0]
L1
A[3:0]
DO[3:0]
ROM
ZE[7:0]
H11
INI
H
H2
H1
H0
COMMANDEDUFILTRE1
RA[4:1]
RA4
RA3
RA2
RA1
B4
B3
B2
B1
A4
A3
A2
A1
Y4
Y3
Y2
Y1
S4 LU3
S3 LU4
S2
S1
MUX
COUNTER
H10
LU1
LU2
A3
A2
A1
A0
D3
D2
D1
D0
LUT
L5
ROM
ADR[4:1]
LU[4:1]
Xilinx Corporation
2100 Logic Drive
San Jose, CA 95124
reseaux systolique: reseau de neurone
neurone: systolique
Date: 15/11/2000

ملخص

يهدف هذا العمل الى انجاز هندسات لشبكات الأعصاب السيستوليكية والمتسلسلة. ولهذا الغرض فقد قسمنا عملنا الى جزئين، جزء التعلم ودوره، حساب الأثقال لوضعهم بعد ذلك في ذاكرة ROM لإستعماله في جزء المعرفة. يمكننا استعمال نفس وحدة التحكم مهما تكن أبعاد المرشح وشبكات الأعصاب. يبقى للمصمم إختيار الهندسة الملائمة لإحتياجاته.

ABSTRACT

This work consists on presenting digital neural networks architecture, which are séquence and systolique architectures. The hardware implementation content two stages. The first stage of learning is used to calculate synapses in order to put it in ROMs, in the seconde stage of reognition or imprementation we use those synapses. The implementation of filter and neural networks with systolique architeture show that the unity of control is the same whatever the dimension of this filter end this neural networks. In fact, it is up to the designer to choose adequate architecture for his model in order to satisfy the needed performances.

RESUME

Ce travail consiste à implémenter les architectures séquentielle et systolique des réseaux de neurones, pour réaliser cette implémentation on a divisé notre étude en deux grandes étapes. L'étape d'apprentissage sert à calculer les synapses afin de les stocker dans des ROMs pour l'étape de reconnaissance ou l'étape d'implémentation des architectures. Les implémentations hardwares du filtre récursif et du réseau de neurones avec l'architecture systolique, montrent que l'unité de commande est universelle c'est à dire elle est la même quelque soit la dimension de ce filtre de ce réseau de neurones.

En effet, il reste au concepteur de choisir l'architecture adéquate à son modèle afin de satisfaire les performances demandées.

Mots clés :
RNA, RPG, FPGA, Architecture séquentielle, Architecture systolique.

ملخص

يهدف هذا العمل الى انجاز هندسات لشبكات الأعصاب السيستوليكية والمتسلسلة. ولهذا الغرض فقد قسمنا عملنـــا الى جزئين، جزء التعلم ودوره، حساب الأثقال لوضعهم بعد ذلك في ذاكرة ROM لإستعماله في جزء المعرفة. يمكننا استعمال نفس وحدة التحكم مهما تكن أبعاد المرشح وشبكات الأعصاب. يبقى للمصمم إختيار الهندسة الملائمـــة لإحتياجاته.

ABSTRACT

This work consists on presenting digital neural networks architecture, which are séquence and systolique architectures. The hardware implementation content two stages. The first stage of learning is used to calculate synapses in order to put it in ROMs, in the seconde stage of reognition or impremention we use those synapses. The implementation of filter and neural networks with systolique architeture show that the unity of control is the same whatever the dimension of this filter end this neural networks. In fact, it is up to the designer to choose adequate architecture for his model in order to satisfy the needed performances.

RESUME

Ce travail consiste à implémenter les architectures séquentielle et systolique des réseaux de neurones, pour réaliser cette implémentation on a divisé notre étude en deux grandes étapes. L'étape d'apprentissage sert à calculer les synapses afin de les stocker dans des ROMs pour l'étape de reconnaissance ou l'étape d'implémentation des architectures. Les implémentations hardwares du filtre récursif et du réseau de neurones avec l'architecture systolique, montrent que l'unité de commande est universelle c'est à dire elle est la même quelque soit la dimension de ce filtre de ce réseau de neurones.

En effet, il reste au concepteur de choisir l'architecture adéquate à son modèle afin de satisfaire les performances demandées.

Mots clés :
RNA, RPG, FPGA, Architecture séquentielle, Architecture systolique.