Apport des architectures connexionnistes aux rétines électroniques

Bertrand Granado, Patrick Garda

Apport des architectures connexionnistes aux rétines électroniques

Bertrand Granado
Patrick Garda

LIS - Université Pierre et Marie Curie
case postale 203
4, Place Jussieu
75252 Paris Cedex 05
Téléphone : 01 44 27 75 07
Télécopie : 01 44 27 75 09
Courrier Electronique : granado@ufr924.jussieu.fr, garda@ufr924.jussieu.fr

RÉSUMÉ. *Dans cet article nous évaluons l'apport des architectures connexionnistes aux rétines électroniques en nous basant sur une méthode originale d'évaluation et de prédiction des performances que nous avons développée. Nous validons cette méthode sur une classe de réseaux connexionnistes, et sur une architecture existante. Nous proposons une architecture de rétine électronique basée sur une architecture connexionniste programmable et de ce fait adaptée à une large classe de situations.*

ABSTRACT. *In this article we study the impact of connexionnist architectures on electronic retina thanks to an original performance evaluation/prediction method. We validate this method with a specific connexionnist network class and an existing architecture. Finally we propose a versatile electronic retina based on a programmable connexionnist architecture.*

MOTS-CLÉS : *Rétines électroniques, Architectures connexionnistes, Evaluation de performances, Prédiction de performances, CNAPS, Perceptrons multi-couches*

KEYWORDS: *Electronic retina, connexionnist architectures, Performance evaluation, Performance prediction, CNAPS, Multi-layer perceptrons*

1. Introduction

Les rétines électroniques sont des circuits intégrés qui associent intimement un capteur d'image et une architecture parallèle de manière à constituer une réalisation compacte qui évite les transferts de l'un vers l'autre. Le qualificatif de "capteur intelligent" peut être attribué aux rétines électroniques du fait de leurs capacités de traitement des informations produites par le capteur d'image. Ces capacités de traitement sont limitées par la faible quantité des ressources matérielles qui sont attribuées à chaque processeur de l'architecture parallèle. Cette limitation conduit à la conception de deux types de rétines : les rétines dédiées, dont les fonctionnalités sont câblées à la conception et les rétines programmables, capables d'exécuter un jeu d'instructions réduit au traitement des données.

D'un autre coté, les chaînes de traitement qui sont exécutées par des rétines électroniques comportent le plus souvent deux phases. La première phase consiste en une suite de traitements d'images de bas niveau destinée à corriger les imperfections introduites par le capteur et à extraire les caractéristiques pertinentes de l'image. Ensuite, une seconde phase de moyen niveau exploite ces caractéristiques pour effectuer une mesure, prendre une décision ou reconnaître une forme représentée dans l'image produite par le capteur.

Pour rester compatible avec l'économie de matériel qu'exigent les rétines électroniques, il peut être envisagé de réaliser les deux phases de la chaîne de traitements avec la même classe d'algorithmes, et deux points de vue peuvent alors être adoptés. Une première approche consiste à effectuer des tâches de reconnaissance de formes avec des traitements d'images de bas niveau, avec des performances relativement limitées [GRR+88]. Une autre approche consiste à utiliser des réseaux connexionnistes pour effectuer l'ensemble de la chaîne de traitements. C'est celle qui est étudiée dans cet article, avec les perceptrons multi-couches comme exemple de réseaux connexionnistes.

Il faut observer que les réseaux de neurones biologiques ont été une source d'inspiration essentielle dans le domaine des rétines électroniques, à l'instigation des équipes de Carver Mead au Caltech [Mea89] et de Éric Vittoz au CSEM. Ils ont conduit à la conception de nombreux circuits VLSI analogiques originaux et performants. De nombreux circuits analogiques ont aussi été conçus et réalisés pour la simulation de réseaux de neurones formels en temps réel. Ce problème, né du volume important de calculs que nécessite la simulation de réseaux connexionnistes, a aussi été abordé par la conception et la réalisation d'architectures spécialisées numériques, appelées architectures connexionnistes dans cet article.

Il faut aussi remarquer que les architectures qui ont été spécifiquement conçues pour la simulation de réseaux connexionnistes ont le plus souvent eu comme objectif de réduire la latence des simulations, c'est-à-dire leur durée totale, plutôt que la cadence des simulations, c'est-à-dire le nombre de simulations qui peut être effectué par unité de temps [HP92]. Alors que c'est ce second point de vue qui a été adopté par la plupart des études de parallélisation de réseaux connexionnistes sur des machine parallèles [APMP93] a priori destinées au calcul scientifique, c'est le premier point de vue qui est adopté dans cet article, parce qu'il correspond aux contraintes des applica-

tions qui nécessitent des architectures connexionnistes ou des rétines électroniques.

Dans ce contexte, l'objet de cet article est de situer l'apport des architectures connexionnistes aux rétines électroniques. En effet, les études de rétines électroniques sont menées le plus souvent dans la perspective de l'exploitation ultérieure des progrès des technologies de réalisation de circuits intégrés, qui permettront la réalisation de rétines de taille significative à partir des concepts validés sur des démonstrateurs de petite taille construits dans des technologies disponibles actuellement. Cependant, cette perspective permet aussi d'envisager la conception de rétines électroniques basées sur des architectures connexionnistes, ce qui n'a jamais été fait à notre connaissance. Cette approche originale permet d'envisager des architectures parallèles programmables, dont les ressources matérielles permettent d'effectuer des traitements de plus haut niveau que ceux réalisés par les rétines électroniques classiques, quitte à ce que le degré de parallélisme soit plus petit [GRS91]. Ces architectures pourraient donc avoir des performances plus faibles, et par conséquent des latences de traitement plus longues, que des rétines électroniques.

Il apparaît donc que le point clé de cette étude est la comparaison des performances accessibles aux deux approches et la prédiction de leurs évolutions futures. Pour traiter ce point, nous avons élaboré une méthode quantitative d'évaluation et de prédiction des performances des architectures connexionnistes, que nous avons validée à l'aide d'un exemple de perceptron multi-couches simulé sur une machine parallèle possédant 128 processeurs.

Cet article est organisé en quatre parties. La première section décrit l'état de l'art des rétines électroniques et des architectures connexionnistes, et la seconde les réseaux connexionnistes qui nous servent de support. La troisième section décrit la méthode d'évaluation et de prédiction de performances mentionnée précédemment, et la dernière section sa validation et ses conséquences sur l'exploitation des architectures connexionnistes pour la conception des rétines électroniques.

2. Etat de l'art : rétines électroniques et architectures connexionnistes

Les rétines électroniques sont des circuits intégrés qui associent intimement des photocapteurs et des processeurs de manière à constituer une réalisation compacte qui évite les transferts des premiers vers les seconds. Les deux caractéristiques principales qui permettent de classer les rétines électroniques sont le style de conception, qui peut être analogique ou numérique, et la structure de l'imageur. Les rétines analogiques sont les réalisations les plus spectaculaires et les plus prometteuses à long terme pour des applications dédiées, elles sont présentées dans [Ber97]. Quelques rétines numériques ont aussi été réalisées, qui bénéficient d'une plus grande facilité de conception et d'une architecture programmable, c'est à elles que nous nous limiterons dans la suite de cette section.

Dans les rétines électroniques une distinction importante résulte de la structure de l'imageur que constitue la matrice des photocapteurs incluse dans la rétine. En effet cet imageur peut-être centralisé, c'est-à-dire qu'une région de la rétine lui est entièrement

dédiée, ou distribué, c'est-à-dire que chaque processeur inclut son propre photocapteur. Les rétines à imageur centralisé privilégient les qualités de celui-ci, et celles à imageur distribué privilégient l'interaction entre les photocapteurs et les processeurs.

Une rétine à imageur centralisé est composée d'un imageur, qui fournit des images analogiques échantillonnées spatialement et temporellement, d'une unité de traitement et d'une unité d'entrées-sorties. Lorsque l'unité de traitement est numérique, la rétine contient aussi une unité de codage analogique-numérique. Comme dans ce cas l'unité de traitement est programmable, la rétine inclut généralement une unité de décodage d'instructions.

Cette architecture est particulièrement modulaire, et elle permet la réalisation de démonstrateurs dans lesquels les unités sont réalisées dans des circuits différents, de manière à valider les concepts avec les technologies disponibles et à limiter les risques de la conception.

Un point clé est l'architecture du réseau de processeurs.Pour le traitement d'images bas niveau, les grilles fournissent les communications aux plus proches voisins adéquates pour l'exécution des algorithmes locaux. La dimension de la grille constitue alors un élément de différenciation des architectures.

Lorsque la dimension de la grille est inférieure à celle de l'image, il faut reconstituer le voisinage nécessaire au calcul des valeurs résultats dans des algorithmes locaux. Ceci est accompli par une unité de mise sous forme locale : elle est constituée d'une mémoire contenant autant de lignes successives de l'image d'entrée que le support du traitement local, et d'un module de génération d'adresse qui est chargé d'extraire de ces lignes le voisinage du pixel d'entrée nécessaire au calcul. Cette architecture est classique dans les machines pipe-line de traitement d'images à la volée en temps-réel, dans lesquelles les unités de traitement sont dédiées et n'exploitent pas de parallélisme de données [Man95].

Le choix d'un processeur ligne constitue un bon compromis entre la surface de l'unité de traitement et le gain en vitesse qu'elle apporte. Deux rétines incluant une ligne de processeurs ont été commercialisées par la société Integrated Vision Products, le plus connu étant le circuit MAPP2200, qui est une rétine numérique à imageur 2-D centralisé. Il contient un imageur 258x256, une ligne de 256 convertisseurs analogique-numérique sur 8 bits (maximum), et une ligne de 256 processeurs bit-série [Åst93].

Un petit nombre de rétines numériques programmables contenant une matrice 2-D de processeurs ont aussi été décrites. Dans ce cas l'imageur a une structure distribuée, contrairement aux sections précédentes. Un exemple est le circuit de rétine TCL 65x76 réalisé sur une surface de 50 mm2 dans une technologie CMOS 2 μm numérique [BZD92].

Par ailleurs, le domaine de machines connexionnistes est un domaine où la recherche est assez foisonnante. Deux axes apparaissent, un axe de développement de circuits spécifiques, généralement des circuits analogiques, dédiés à un unique réseau de neurones formels ou plus couramment, dédiés à un modéle connexionniste. C'est le cas de cicuits tel qu'ANNA [BSB$^+$91, BSB$^+$92, SBB$^+$92], dédié aux Perceptrons Multi-Couches, ou RÂ [KPG94], dédié à la machine de Boltzmann synchrone. Ce type de circuit reste assez limité et n'intégre pas un grand nombre de neurones.

Un autre axe de développement est celui des machines numériques programmables, ce type de machine permettant la simulation d'un grand nombre de réseaux de neurones formels et de tous les modèles connexionniste. Citons comme réalisation, SYNAPSE [Ram92], machine de type systolique développée par Siemens et CNAPS [Ham90, GTK+91] machine SIMD développée par Adaptive Solutions Inc., cette dernière machine étant la plus performante des machines connexionnistes numériques programmables actuellement : son architecture sera présentée dans la dernière partie de cet article.

3. Les perceptrons Multi-couches

Les Perceptrons Multi-Couches (PMC), sont des graphes orientés dont les sommets sont des neurones. Chaque neurone possède un état calculé en fonction d'une variable interne au neurone, son potentiel post-synaptique.
Les neurones d'un PMC sont reliés entre eux par des arcs valués appelés synapses, la valeur d'une synapse étant appelée poids synaptique.
L'ensemble E_i des neurones possédant une synapse vers le neurone i est appelé voisinage du neurone i.
Les règles de calcul des états des neurones sont les suivantes :

$$V_i = \sum_{j \in E_i} X_i W_{ij} \tag{1}$$

Dans l'équation 1 X_i est l'état du neurone i, W_{ij} est le poids synaptique de l'arc de j vers i et V_i est le potentiel post-synaptique du neurone i.

$$X_i = f(V_i) \tag{2a}$$

$$= m \frac{1 - e^{-\lambda V_i}}{1 + e^{\lambda V_i}} \tag{2b}$$

Dans l'équation 2a la fonction f est appelée fonction d'activation, en général c'est une fonction sigmoïdale, par exemple celle explicitée dans l'équation 2b.

Les PMC sont organisés en couches. Une couche est un ensemble ordonné de N neurones possédant les mêmes caractéristiques connexionnistes, elle peut être mono-dimensionnelle ou bi-dimensionnelle. Chaque neurone d'une couche est repéré par son rang i dans le cas d'une couche mono-dimensionnelle et par son rang (i, j) dans le cas d'une couche bi-dimensionnelle.
Pour représenter des PMC de grande taille, il est commode d'introduire la notion de connexions entre couches. Les couches étant interconnectées deux à deux, on appele couche initiale la couche qui possède les neurones d'où partent les arcs et couche terminale la couche où arrivent les arcs. Dans les PMC il y a trois types de couches :

- les couches d'entrée qui ne sont que des couches initiales,
- les couches de sortie qui ne sont que des couches terminales,

– les couches cachées, qui sont à la fois initiales et terminales.

Dans ce contexte les caractéristiques connexionnistes d'un neurone de rang i (resp. de rang (i,j)) sont le cardinal de E_i (resp. $E_{(i,j)}$) c'est-à-dire la taille du voisinage du neurone et la nature des interconnexions entre les neurones. Ces interconnexions peuvent être complètes ou locales.
Soit I une couche initiale et T une couche terminale.
Si pour tout neurone i (resp. (i,j)) appartenant à T, $|E_i| = |I|$ (resp. $E_{(i,j)} = |I|$) alors les deux couches I et T sont connectées à l'aide d'une connexion complète.
Dans les autres cas, I et T sont interconnectées à l'aide de connexions locales, ce qui nécessite que les deux couches aient la même dimension. Ces connexions locales sont invariantes par translation et elles sont définies à l'aide de deux paramètres, la localité L et le recouvrement R. La localité L est le cardinal des voisinages dans la couche I de tous les neurones de la couche T. Le recouvrement R est le nombre de synapses partagées par deux neurones de rang successif de la couche T. Dans le cas de couches bi-dimensionnelles de taille TailleX (son nombre de lignes) et TailleY (son nombre de colonnes), la localité et le recouvrement sont donnés pour les deux grandeurs dimensions. En général, elles sont identiques. Nous ne traiterons dans cet article que ce cas.

Un PMC, comme tout réseau de neurones formel, est un modèle de calcul non plus programmé mais entraîné, c'est-à-dire qu'il construit la fonction qu'il réalise par apprentissage à partir d'exemples. Il existe donc deux phases dans l'exploitation pratique des PMC, la première phase étant appelée phase d'apprentissage, ce qui correspond au calcul itératif de la valeur des poids synaptiques.

Dans cet article nous nous intéressons uniquement à la seconde phase, dite de reconnaissance, des PMC, parce que c'est celle qui est utilisée dans le cas de rétines électroniques. Dans cette phase, les états des neurones des couches d'entrée sont imposés, à partir de données produites par le capteur d'image dans le cas de rétine électroniques, et ces états sont propagés de couche en couche jusqu'aux neurones de sortie.

4. Méthodologie d'évaluation et de prédiction de performances

4.1. *Introduction*

L'objectif de cette méthodologie est d'évaluer, à l'aide d'un algorithme, la latence d'une simulation, c'est-à-dire sa durée totale, à partir d'une description du PMC sous la forme d'un graphe de couches et d'une description de l'architecture de la machine sur laquelle le PMC est simulé.

4.2. *Description de la méthode*

Notre méthode peut-être décrite par les quatre étapes suivantes :

1. Identification des opérations élémentaires caractérisant la simulation des PMC. Ces opérations élémentaires sont appelées primitives.

2. Détermination d'un schéma de parallélisation sur l'architecture cible.
3. Evaluation des temps de traitement des primitives.
4. Prédiction du temps de simulation d'une application mise en oeuvre sur l'architecture cible à l'aide d'un algorithme basé sur un parcours du graphe de couches décrivant le PMC.

L'étape 1 sera réalisée une fois pour toutes quelle que soit l'architecture de la machine cible.
Les étapes 2 à 3 s'effectuent une fois pour toutes pour chaque architecture cible, et elles offrent un modèle analytique de la simulation des PMC sur l'architecture de la machine cible.
L'algorithme de l'étape 4 doit être appliqué pour déterminer la durée de la simulation de chaque nouveau réseau.

Pour évaluer la phase de relaxation à laquelle nous nous sommes intéressés dans cette étude, nous disposons d'une unité de mesure qui est le CPS [DAR88]. Cette unité nous indique le nombre de connexions traitées par seconde, la forme générale d'une fonction de puissance en CPS est la suivante :

$$r = \quad f\frac{N_c}{N_t} \text{ CPS}$$

où f est la fréquence de la machine, N_c le nombre de connexions traitées et N_t le nombre de cycles machine nécessaire pour traiter ces connexions.

4.3. *Identification des opérations élémentaires caractérisant les perceptrons multi-couches*

Pour chaque sommet et chaque arc rencontré dans le graphe de couches, l'une des trois primitives suivantes est exécutée :

- Le calcul des potentiels post-synaptiques dans le cas de connexions locales,
- Le calcul des potentiels post-synaptiques dans le cas de connexions complètes,
- La mise à jour de l'état des neurones à partir des potentiels post-synaptiques.

4.4. *Détermination d'un schéma de parallélisation sur l 'architecture cible*

4.4.1. *Introduction*

Nous avons choisi comme classe d'architecture cible, les architectures SIMD avec anneau 1-D comme moyen de communication. Le but du schéma de parallélisation va être une minimisation de la latence. Pour cela notre placement tend à atteindre une efficacité maximale de l'architecture, en faisant travailler en parallèle le plus grand nombre possible de processeurs.

4.4.2. *Le placement*

Le placement effectué optimise les communications dans le cas de connexions locales. L'idée est de centrer le placement des neurones de la couche terminale par rapport au placement des neurones de la couche initiale, lorsque ces deux couches sont interconnectées à l'aide de connexions locales. Par neurone, on entend ici le couple état X_i (resp. $X_{(i,j)}$) et potentiel post-synaptique V_i (resp. $V_{i,j}$) qui caractérise le neurone i (resp. (i,j)).

Soient, par exemple, deux couches mono-dimensionnelles I et T interconnectées par

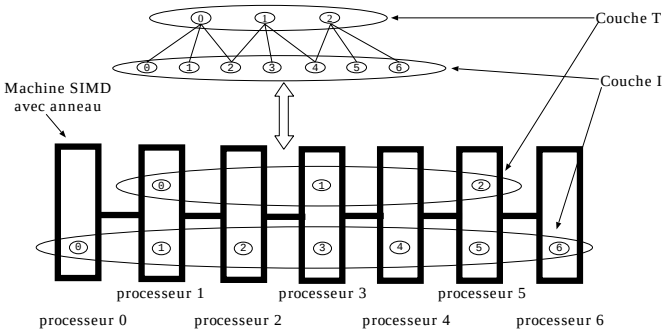

FIG. 1: Placement de deux couches T et I

des connexions locales. I a une taille de 7 neurones et T de 3 neurones, la localité L de la connexion est 3 et son recouvrement R est 1. Les neurones de la couche I sont placés consécutivement sur les processeurs de numéro 0 à 6 et ceux de la couche T sur les processeurs de numéro 1, 3 et 5, comme le montre la figure 1. Ainsi, tous les neurones de T sont centrés par rapport aux neurones de la couche I avec lesquels ils sont interconnectés.

Ce placement permet d'avoir pour tout neurone i (resp. (i,j)) une distance d'au plus $\lfloor \frac{L}{2} \rfloor$ (resp. $(TailleX+1)\lfloor \frac{L}{2} \rfloor$) pour les couches mono-dimensionnelles (resp. couches bi-dimensionnelles) avec le plus éloigné des neurones auquel il est connecté. Ceci minimise la distance de communication globale dans le PMC.

Ce placement permet aussi d'obtenir une distance de communication, pour tout couple de neurones $(i_T, (i(L-R)+k)_I)$, i_T appartenant à la couche terminale et $(i(L-R)+k)_I$ appartenant à la couche initiale, invariante par translation et donc ne dépendant que de k appartenant à l'intervalle $[0, L-1]$. Cette invariance réduit le nombre des instructions conditionnelles, qui sont hyper-séquentielles dans les machines SIMD [GRS91].

Le placement énoncé précédemment est naturellement récursif, puisque le placement des neurones d'une couche peut dépendre du placement des neurones d'une autre couche qui lui même peut dépendre du placement des neurones d'une troisième couche. Ceci peut conduire à de grandes distances de communication, surtout dans le cas de couches bi-dimensionnelles.

Pour éviter cette récursion, seuls les potentiels post-synaptiques sont placés suivant ce placement, les états quant à eux sont toujours placés de façon successive sur les processeurs en commençant toujours par le processeur de numéro 0.
Ainsi toute récursion est évitée, mais il faut effectuer une translation des potentiels post-synaptiques V_i (resp. $V_{(i,j)}$) vers les processeurs où sont placés les états des neurones X_i (resp. $X_{(i,j)}$) correspondants.

Le placement est réalisé à l'aide de deux fonctions qui chacune associent un numéro de processeur à un rang de neurone : une fonction $P\_état()$ pour les états des neurones et une fonction $P\_potentiel()$ pour les potentiels post-synaptiques. Dans les règles de calcul qui suivent, *couche* désigne la couche à placer, c'est aussi la couche terminale de la connexion quant il y en a, et *initiale* désigne la couche initiale de la connexion.

- Tous les états des neurones de rang (i), pour les couches mono-dimensionnelles, ou de rang (i,j), pour les couches bi-dimensionnelles, sont placés suivant la règle :

  - si la couche est mono-dimensionnelle
    P_état(couche,i)= $\left\lceil \frac{i}{\left\lceil \frac{TailleX(couche)}{P} \right\rceil} \right\rceil$
  - si la couche est bi-dimensionnelle
    P_état(couche,i,j)= $\left\lceil \frac{i+j.TailleX(couche)}{\left\lceil \frac{TailleX(couche).TailleY(couche)}{P} \right\rceil} \right\rceil$

- Tous les potentiels post-synaptiques de rang (i), pour les couches mono-dimensionnelles, ou de rang (i,j), pour les couches bi-dimensionnelles sont placés suivant les règles :
  - Si la connexion est complète et la couche mono-dimensionnelle
    P_potentiel(couche,i,j)= $\left\lceil \frac{i}{\left\lceil \frac{TailleX(couche)}{P} \right\rceil} \right\rceil$
  - Si la connexion est complète et la couche bi-dimensionnelle
    P_potentiel(couche,i,j)= $\left\lceil \frac{i+j.TailleX(couche)}{\left\lceil \frac{TailleX(couche).TailleY(couche)}{P} \right\rceil} \right\rceil$
  - Si la connexion est locale et les couches mono-dimensionnelles
    P_potentiel(couche,i)= $\left\lceil \frac{\left\lfloor \frac{l}{2} \right\rfloor + i(l-r)}{\left\lceil \frac{TailleX(initiale)}{P} \right\rceil} \right\rceil$
  - Si la connexion est locale et les couches bi-dimensionnelles
    P_potentiel(couche,i,j)= $\left\lceil \frac{\left\lfloor \frac{l}{2} \right\rfloor + i(l-r) + TailleX(initiale).(\left\lfloor \frac{l}{2} \right\rfloor + i(l-r))}{\left\lceil \frac{TailleX(initiale).TailleY(initiale)}{P} \right\rceil} \right\rceil$

4.4.3. *Algorithme*

L'algorithme de placement décrit dans la figure 2 consiste en un parcours en largeur du graphe de couches. A chaque couche rencontrée pour la première fois dans le parcours, les deux fonctions P_état() et P_potentiel() sont appliquées afin d'attribuer

à chaque état et à chaque potentiel post-synaptique de chaque neurone de la couche un numéro de processeur. Le parcours s'effectue tant que toutes les couches n'ont pas été visitées au moins une fois.

Début
Parcours en largeur du graphe représentant le PMC
liste sommet = vide
booléen tableau[Nombre de couches du PMC] = faux
couche initiale = première couche du réseau # couche d'entrée
couche terminale = vide
tableau[initiale]=vrai
ajouter(initiale,sommet)
P_ état$(initiale,\ i,\ j)=\lceil \frac{i+j.TailleX(terminale)}{\lceil \frac{TailleX(terminale).TailleY(terminale)}{P} \rceil} \rceil$
tant que sommet n'est pas vide **Faire**
  initiale = premier(sommet)
  retirer(initiale,sommet)
  Pour tous les successeurs de initiale **Faire**
    terminale=prochain successeur(initiale)
    Si Tableau[terminale]=faux **Alors**
      tableau[terminale]=vrai>
      ajouter(terminale,sommet)
      Pour tous les neurones de rang $(i,\ j)$ de la couche **Faire**
        P_ état$(terminale,\ i,\ j)=\lceil \frac{i+j.TailleX(terminale)}{\lceil \frac{TailleX(terminale).TailleY(terminale)}{P} \rceil} \rceil$
      Fin Faire
      Pour tous les potentiel post-synaptique de rang $(i,\ j)$ de la couche **Faire**
        Si Connexion(initiale,terminale)= complète **Alors**
          P_potentiel$(terminale,\ i,\ j)=\lceil \frac{i+j.TailleX(initiale)}{\lceil \frac{TailleX(initiale).TailleY(initiale)}{P} \rceil} \rceil$
        Si Connexion(initiale,terminale)= locale **Alors**
          Si TailleY(terminale)=1 **alors**
            P_ potentiel$(terminale,\ i)=\lceil \frac{\lfloor \frac{l}{2} \rfloor + i(l-r)}{\lceil \frac{TailleX(initiale)}{P} \rceil} \rceil$
          Fin Si
          Sinon alors
            P_ potentiel$(terminale,\ i,\ j)=\lceil \frac{\lfloor \frac{l}{2} \rfloor + i(l-r) + TailleX(initiale).(\lfloor \frac{l}{2} \rfloor + i(l-r))}{\lceil \frac{TailleX(initiale).TailleY(initiale)}{P} \rceil} \rceil$
          Fin Si
        Fin Si
      Fin Faire
    Fin si
  Fin Pour
Fin tant que
Fin

FIG. 2: Algorithme de placement

4.5. *Algorithme de Prédiction*

Le graphe de couches décrivant le PMC permet de définir l'ordonnancement des différentes primitives qui lui sont associées. Dans le cas des PMC, les états des neurones d'une couche ne peuvent pas être calculés avant d'avoir calculé tous les potentiels post-synaptiques de cette couche, et les potentiels post-synaptiques d'une couche ne peuvent pas être calculés avant de connaître les états des neurones qui interviennent dans le calcul de ces potentiels.
A chaque noeud du graphe de couche est associé un temps de traitement qui est la somme des temps nécessaires au calcul des potentiels post-synaptiques de la couche et au calcul des états des neurones.
La prédiction du temps de simulation du réseau s'effectue alors en deux étapes :

1. Dans la première étape on prédit le temps nécessaire à chaque noeud pour effectuer ses calculs, grâce aux fonctions de temps déterminées pour les primitives.

Un exemple de ces fonctions sera donné dans la section suivante.

2. Dans une seconde étape, on prédit le temps de simulation total. Ce temps de simulation est calculé en parcourant le graphe de couches dans un ordre déterminé d'après le placement effectué.

Dans le cas de la simulation des PMC sur une machine SIMD avec un traitement séquentiel des couches et parallèle de chaque couche, le parcours est séquentiel, le temps total est donc la somme de tous les temps calculés pour chaque noeud du graphe.

5. Validation de la méthode d'évaluation et de prédiction de performances

La méthode décrite dans la section précédente a été validée sur une classe de PMC utilisée pratiquement et une architecture connexionniste existante.

5.1. *Les PMC utilisés pour la validation*

Pour valider notre méthode, nous avons choisi de simuler les PMC à Champs d'Activation Locaux (PMC-CAL), qui sont des PMC combinant des couches 1-D et des couches 2-D. Dans ce type de PMC les couches 2-D forment les couches d'entrées et cachées, elles sont interconnectées entre elles par des connexions locales, et le plus souvent se partagent un même jeu de poids. Les couches 1-D forment quant à elles les couches de sortie et sont interconnectées par des connexions complètes à un sous ensemble des couches cachées. Ce schéma d'interconnexion permet une réduction considérable du nombre de connexions, comparé à un PMC où toutes les connexions sont de type complètes. En combinant des informations locales dans les couches cachées, et en regroupant ces informations à l'aide de connexions complètes dans les couches de sortie, les PMC-CAL obtiennent de très bons résultats en reconnaissance de formes visuelles. Ces réseaux ont été utilisés avec succès pour la reconnaissance de caractères. Parmi ce type de PMC nous avons simulé LeNet [LBD+90] qui est un PMC-CAL dédié à la reconnaissance de caractères manuscrits possédant 4365 neurones, 1920 connexions complètes et 96522 connexions locales.

5.2. *L'architecture cible: la machine CNAPS*

Structure générale de CNAPS CNAPS est une machine parallèle de type SIMD constituée d'un tableau mono-dimensionnel de processeurs appelés Processeurs Noeuds (PN), comme le montre la figure 3. L'ensemble des PN est contrôlé par un processeur externe, le séquenceur, qui génère les instructions exécutées simultanément par l'ensemble des processeurs. Le séquenceur délivre les instructions aux PN par l'intermédiaire d'un bus appelé PNCMD de trente deux bits de large.
La machine CNAPS est constituée de circuits intégrés comprenant 13,8 Millions de transistors sur une surface de 2,54x2,54 cm^2, ce qui permet d'obtenir 64 PN effectivement utilisables par circuit.

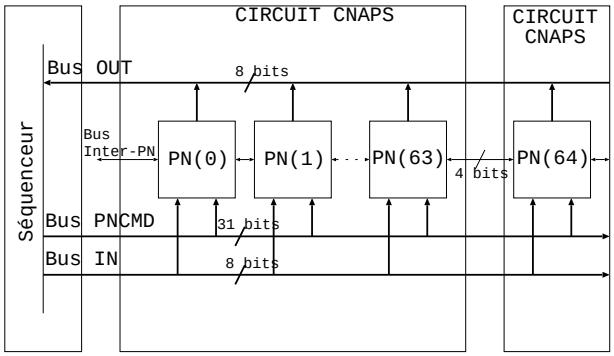

FIG. 3: Architecture de la machine

Les processeurs sont reliés entre eux par deux bus de communication, visibles sur la figure 3. Ces bus sont :

- un bus à diffusion composé de deux entités, une entité appelée IN de huit bits de large, qui permet d'effectuer la diffusion des données du séquenceur vers les PN, une entité appelée OUT de huit bits de large, qui permet d'effectuer le transfert des données d'un PN vers le séquenceur,
- un bus appelé Inter-PN de quatre bits de large, deux bits en entrée et deux bits en sortie, qui relie les processeurs en anneau, figure 4. Il permet d'effectuer des communications aux plus proches voisins.

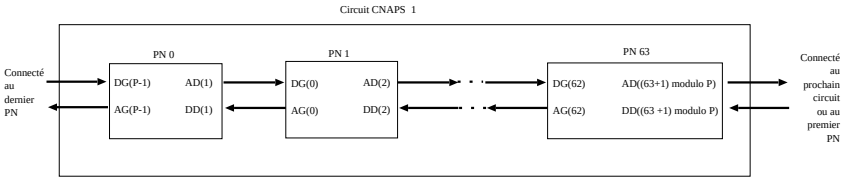

FIG. 4: Organisation des communications via le bus inter-PN

Le processeur élémentaire de CNAPS Chaque processeur possède, en interne, deux bus, A et B, de seize bits de large chacun, et sept unités fonctionnelles : un multiplieur neuf bits par seize bits, un additionneur trente deux bits, trente deux registres de seize bits, une unité de génération d'adresse mémoire, une unité logique, un port d'entrée et un port de sortie. La structure interne d'un processeur est présentée sur la figure 5. Les PN ne travaillent qu'en arithmétique entière ou en virgule-fixe, ils ne possèdent pas d'unité de calcul flottant.
CNAPS possède une mémoire répartie sur l'ensemble des PN : cette mémoire permet

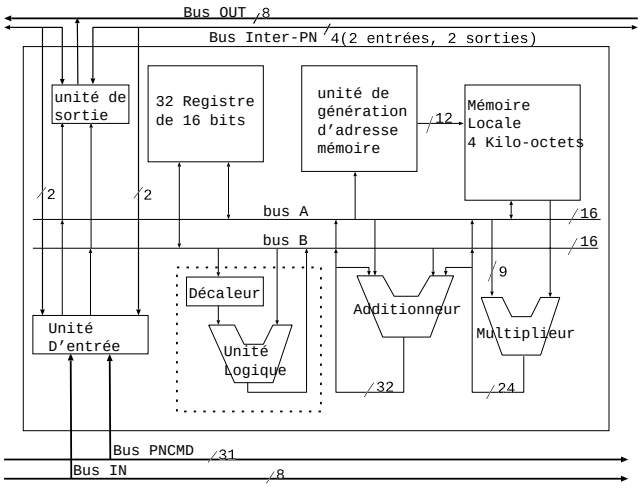

FIG. 5: Architecture interne d'un processeur

un accès rapide aux données, pour les processeurs, car c'est une mémoire statique et interne aux processeurs. Chaque processeur possède 4 kilo-Octets de mémoire, ce qui pour la machine à 128 processeurs que nous avons utilisée représente une mémoire de 512 kilo-Octets. De plus, il y a une une mémoire dynamique de 16 Méga-Octets disponible sur le séquenceur.

CNAPS : une machine programmable CNAPS est une machine programmable, on peut donc a priori simuler n'importe quel réseau connexionniste. Pour ce faire il existe deux niveaux de programmation, un niveau haut à l'aide du langage C-CNAPS, langage C dérivé de C* permettant la gestion du mode SIMD de la machine à haut niveau, et un bas niveau constitué d'un langage assembleur propre à CNAPS, CPL.
Nous avons utilisé le langage CPL, car bien que C-CNAPS soit un langage attrayant, le compilateur existant ne génère pas un code suffisamment optimisé pour atteindre des performances effectives suffisamment élevées.

5.3. *Evaluation de la machine CNAPS*

La machine CNAPS possédant 2 bus de communications, avec les trois fonctions introduites dans la section 4.3., cela nous donne en tout 5 primitives différentes, puisque le placement effectué limite les communications aux phases de calculs des potentiels post-synaptiques.

1. Calcul des potentiels post-synaptiques dans le cas de connexions complètes en utilisant le bus à diffusion.
2. Calcul des potentiels post-synaptiques dans le cas de connexions locales en utilisant le bus à diffusion.

3. Calcul des potentiels post-synaptiques dans le cas de connexions complètes en utilisant le bus Inter-PN.
4. Calcul des potentiels post-synaptiques dans le cas de connexions locales en utilisant le bus Inter-PN.
5. Calcul des états des neurones.

réseau d'interconnexion	temps en nombre de cycles
Potentiels Post-synaptiques pour les connexions complètes	
Bus IN et OUT	α + γ*NPPD + ε*NPPD*NPPA + θ*PD*NPPA*NPPD
Bus Inter-PN	α + ε*NPPD + θ*NPPD*P + ψ*NPPD*NPPA + ϕ*NPPD*NPPA*PD
Potentiels Post-synaptiques pour les connexions locales	
Bus IN et OUT	α + γ*NPPD + ε*NPPD*PD + θ*NPPA + ψ*NPPA*C_1*C_2
Bus Inter-PN	α + γ*NPPD + ε*DD + ω*DG + β*NPPA + θ*NPPA*C_2 + ψ*NPPA*C_1*C_2
Mise à jour de l'état des neurones	
	α + θ*NPPA

TAB. 1: Fonctions temps pour les primitives de base

Le modèle analytique de la durée de traitement sur CNAPS, pour ces 5 primitives, est décrit dans le tableau 1. Dans ce tableau les variables NPPA et NPPD indiquent le nombre de neurones par couches pour, respectivement, la couche initiale de la connexion et la couche terminale, PD indique le nombre de processeurs utilisés par la couche initiale de la connexion, C_1 indique la taille du voisinage de la connexion pour la première dimension et C_2 pour la seconde dimension, DG et DD indiquent respectivement la distance de communication à gauche et à droite, ces valeurs dépendant de C_1, de C_2 et de la taille de la couche initiale. Les paramètres α, γ, ε, θ, ψ, ω, β, ont été quant à eux estimés grâce à une analyse du code source et à des mesures expérimentales [GG96].
Ce modèle nous permet de prédire la durée de simulation d'un PMC-CAL, en effectuant un parcours de son graphe de couches et en accumulant les temps d'exécution de chaque primitive en chaque sommet et en chaque arc.

Pour vérifier la validité du modèle, nous l'avons utilisé pour prédire la durée de simulation de LeNet. En la comparant au temps réellement mesuré (tableau 2) nous constatons que l'erreur commise n'est que de 10%, ce qui est tout à fait acceptable.

Architecture	Temps de Simulation (ms)	Erreur
CNAPS 128 20 Mhz (mesuré)	2,57	
CNAPS 128 20 Mhz (prédit)	2,26	10%

TAB. 2: Temps de simulation de LeNet - Prédit et Mesuré

5.4. *Application de l'algorithme de prédiction*

Grâce à notre modèle analytique nous avons effectué des prédictions de temps de simulation sur plusieurs architectures virtuelles, obtenues par des modifications de l'architecture initiale de CNAPS :

Architecture	Temps de Simulation (ms)	Vitesse (MCPS)
CNAPS 64 20 Mhz	3,08	31,9
CNAPS 128 25 Mhz	1,83	53,7
CNAPS 512 20 Mhz	1,75	56,1
E-CNAPS 128 20 Mhz	1,92	51,2
E-CNAPS 512 25 Mhz	1,28	76,6
ANNA	1,20	82,0

TAB. 3: Prédiction du temps de simulation de LeNet avec différents changements dans l'architecture de CNAPS, comparaison avec le temps mesuré sur le cicuit ANNA

- CNAPS avec 64 PNs à la fréquence de 20 Mhz (diminution du nombre de PNs à fréquence constante),
- CNAPS avec 128 PNs à la fréquence de 25 Mhz (augmentation de la fréquence à nombre de PNs constant),
- CNAPS avec 512 PNs à la fréquence de 20 Mhz (augmentation du nombre de PNs à fréquence constante),
- E-CNAPS avec 128 PNs à la fréquence de 20 Mhz. (E-CNAPS diffère de CNAPS par un bus Inter-PN de 8 bits de large et par un registre de base d'adresse auto-décrémental),
- E-CNAPS avec 512 PNs à la fréquence de 25 Mhz (augmentation de la fréquence et du nombre de PNs de E-CNAPS).

Avec la première modification, une baisse de seulement 30% des performances est observée avec un gain de surface de 50%, ce qui permet d'envisager une implémentation de l'ensemble rétine et architecture connexionniste dans un seul circuit en gardant des temps de simulation satisfaisants. Ce point sera développé dans la section suivante.
Les prédictions effectuées, tableau 3, montrent un gain sensible de performance de l'ordre de 30% pour chacune de trois modifications suivantes. La dernière modification, qui est une combinaison des trois précédentes, montre un gain de l'ordre de 100%. Elle conduit à une latence de simulation très voisine de celle obtenue avec ANNA, un circuit analogique dédié à la simulation des PMC-CAL comme LeNet.

5.5. *Conclusion*

Grâce à la méthode de prédiction proposée et appliquée à un cas spécifique, la simulation des PMC-CAL sur CNAPS, il nous a été possible de prévoir les temps de simulation de LeNet sur une architecture dérivée d'une amélioration de CNAPS et de montrer que les performances de cette architecture sont très voisines de celles obtenues avec un circuit analogique spécifique tel qu'ANNA.

6. Rétine Electronique basée sur CNAPS

A partir des éléments précédents nous pouvons proposer une architecture de rétine connexionniste programmable, qui associe un capteur d'image centralisé, une électronique de conversion analogique-numérique et une architecture connexionniste, comme

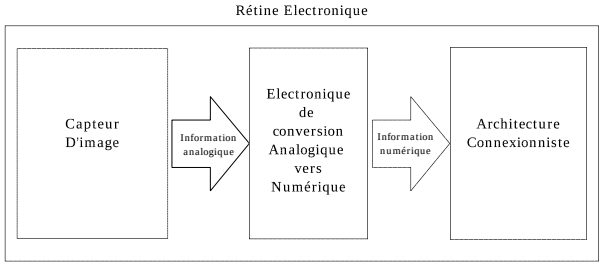

FIG. 6: Principe d'une architecture de rétine connexionniste programmable

cela est représenté sur la figure 7.

Nous pouvons alors prédire les performances de cette rétine à partir des performances de ses composantes, qui peuvent elles-mêmes être estimées à partir de l'état de l'art pour ce qui concerne les fonctions électroniques, et à partir des résultats établis dans les sections précédentes pour ce qui concerne l'architecture connexionniste.

Les images de caractères exploitées par les PMC-CAL sont généralement de petite taille, typiquement 20x30. Pour cette raison nous faisons notre évaluation avec un imageur de taille 64x64 et un tableau de 64 processeurs.

Les performances de ce tableau de 64 processeurs, identiques à ceux de CNAPS et fonctionnant à la même fréquence, ont été établis dans la section précédente. Elles conduisent à une durée de reconnaissance de 3,08 ms pour LeNet.

Etudions maitenant la faisabilité de l'imageur 64x64. Il se trouve que la conception et la réalisation d'imageurs dans des technologies standards CMOS est un sujet d'une grande actualité dans le domaine de l'électronique. En particulier, des circuits dont la seule fonction est celle d'imageurs 2-D ont été réalisés préalablement ou parallèlement à la conception de rétines numériques à imageurs centralisés :

- à l'Université d'Edimbourg, dans une technologie CMOS numérique (SLP DLM) 2μm, un imageur 128x128 à sortie analogique, et une caméra vidéo à sortie analogique au format CCIR avec un imageur 312x287 [RDWL90]
- à l'Université de Linköping, dans une technologie CMOS numérique (SLP DLM) 1,2 μm, un imageur 256x256 à sortie numérique [JISF92]
- à l'Université de Pavie, dans une technologie CMOS numérique (SLP TLM) 1,2 μm, un imageur 256x256 à sortie numérique [SMST95]

Le transfert des pixels de l'imageur vers l'architecture connexionniste impose nécessairement une unité de conversion analogique-numérique dont la vitesse est un élément essentiel de la durée totale de traitement.

Cette butée a été étudiée depuis plusieurs années sur le plan électronique, et deux voies ont été explorées pour la dépasser. L'une de ces voies, basée sur une architecture parallèle comprenant un tableau de convertisseurs analogique-numérique, a été validée en particulier par les équipes de Svensson à Linköping et de Maloberti à Pavie, qui ont

montré expérimentalement la faisabilité de différents degrés de parallélisme du tableau de convertisseurs : 256 convertisseurs algorithmiques [Åst93], et 32 convertisseurs algorithmiques [SMST95]. Une autre voie, moins avancée, est basée sur l'exploitation d'un unique convertisseur analogique-numérique rapide comme celui de [Kli96].
Ce convertisseur peut-être associé à un imageur dont la vitesse de transfert des pixels est rapide, comme par exemple l'imageur de taille 128x128 réalisé dans une technologie CMOS 1,2 μm par [AW96]. Dans ce cas, la durée de la sortie d'une image 128x128 est de 1,7 ms, et celle d'une image 64x64 est de 0,44 ms.

Pour inclure CNAPS dans une rétine électronique, il est nécessaire de modifier son architecture afin de pouvoir exploiter les convertisseurs A/N. En effet, les canaux d'E/S prévus dans CNAPS permettent au maximum un débit de 20 Mo/s pour chaque puce, la machine utilisée n'ayant d'ailleurs qu'un seul canal d'E/S pour tout le tableau de PN, à travers un bus VME. Néanmoins, l'expérience de l'équipe de Linköping montre la faisabilité de l'interconnexion d'un capteur d'image et d'un tableau de processeurs par l'intermédiaire d'une électronique de lecture et de conversion analogique-numérique parallèle.

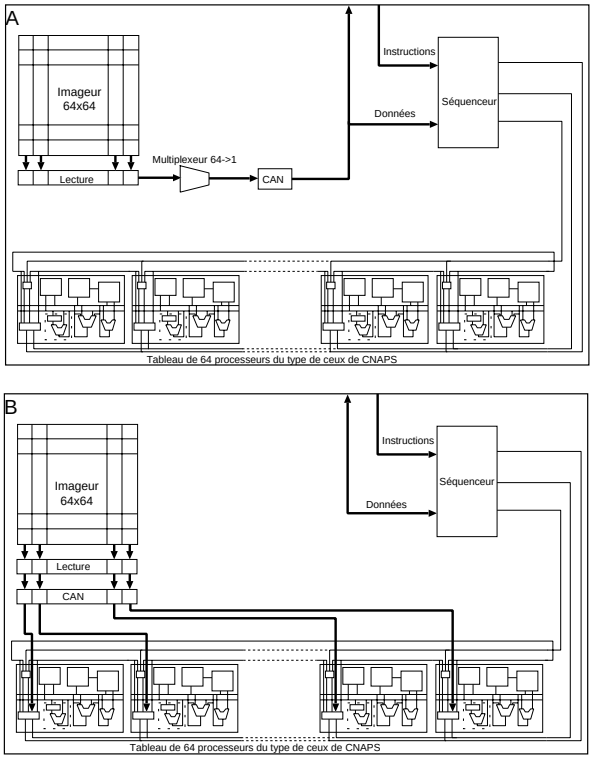

FIG. 7: Schéma bloc des Architectures proposées - A : transfert de l'information séquentiel - B : transfert de l'information parallèle

A partir de ces deux architectures de conversion, nous proposons deux architectures de rétine électronique connexionniste programmable basée sur CNAPS, figure 7. Dans le cas A le transfert s'effectue de manière séquentielle via un multiplexeur et un convertisseur unique, dans le cas B les 64 CAN algorithmiques ont un accès direct aux 64 processeurs donc ils permettent un transfert parallèle.

A partir des éléments précédents nous pouvons déterminer les durées totales accessibles aux deux architectures.

Architecture	Temps de traitement (ms)
A	3,52
B	3,08

TAB. 4: Prédiction du temps de traitement d'une image par l'architecture de rétine électronique connexionniste proposée

Pour démontrer la faisabilité de ces architectures, il reste à traiter de nombreux problèmes d'électronique (couplage analogique-numérique, rapport signal/bruit versus vitesse, consommation, surface) qui sont hors du sujet de cet article consacré aux problèmes d'architecture.

Il reste à situer ces propositions architecturales par rapport aux circuits voisins par leurs fonctionnalités ou leur architecture.

Dans le domaine des circuits connexionnistes, le circuit ANNA constitue une référence puisqu'il a été conçu spécialement pour exécuter des PMC-CAL dont LeNet. Les caractéristiques du circuit sont une puissance crête de 10 GCPS et vitesse de reconnaissance de 1000 caractères/s. Cependant sa mise en oeuvre a demandé le développement de deux cartes VME spécifiques :
la première [SBB+92] a permis des vitesses de 100 caractères/s et la seconde [SG96] a permis 1000 caractères/s soit une vitesse effective de 140 MCPS avec des poids codés sur 6 bits et des états sur 3 bits.

Une autre référence dans le domaine des circuits analogiques est le circuit I1000 de la société SYNAPTICS [PA96]. Il a été conçu spécialement pour la reconnaissance optique des caractères E13B. Ces caractères, qui sont initialement prévus pour une lecture magnétique, sont utilisés pour inscrire des informations numériques dans la ligne inférieure des chèques bancaires. Le circuit contient une rétine 18x24, qui produit 20000 images analogiques par seconde. La sortie de la rétine est exploitée par un classifieur linéaire qui produit 42 indices de confiance, ce qui correspond à trois positions verticales différentes de chacun des 14 caractères de la police E13B. Les 42 indices sont classés par un circuit de type "Winner Take All" analogique [LRMM89], qui choisit le caractère correspondant à l'indice le plus élevé. Les résultats obtenus par ces composants analogiques se sont révélés insuffisants pour certains types de chèques et de ce fait un PMC de type TDNN (Time Delay Neural Network) a été utilisé pour retraiter les sorties du WTA : ce réseau est programmé sur un micro-processeur associé au I1000. Les résultats obtenus en validation sont de 99,995 % de bonne classification dans un prototype de lecteur de chèque. Le circuit consomme 10 mW, et il occupe 0,5x0,5 cm^2 dans une technologie CMOS 1,6 μm.

Cependant, les difficultés de mise au point, l'impossibilité de faire évoluer les fonctionnalités du I1000 ou de l'utiliser pour d'autres applications ont conduit F.Faggin, fondateur de Synaptics, à remettre en cause l'approche analogique dans le contexte de la reconnaissance de formes [Fag94]. De plus, il faut observer que les débouchés commerciaux des rétines analogiques se situent en réalité au niveau des capteurs et de l'instrumentation, c'est-à-dire de capteurs pratiquement dépourvus d'intelligence. Les deux exemples les plus significatifs sont le "TouchPad" de Synaptics et le circuit conçu pour les souris de Logitech par le CSEM [AvSB+96].
Au contraire, la rétine électronique basée sur CNAPS que nous avons proposée peut-être programmée, donc elle peut voir ses fonctionnalités évoluer en fonction des applications.

Une dernière rétine très voisine de celle que nous avons proposée est le circuit MAPP220 de la société IVP (cf section 2.). Les principales limitations rencontrées par les utilisateurs de ce circuit ont porté sur la difficulté de programmation des processeurs bit-série et sur la petite taille de la mémoire des PE. Au contraire l'architecture que nous avons proposée possède une mémoire beaucoup plus grande et sa programmation est plus aisée que celle des processeurs bit-série. Néanmoins cette programmation reste ardue et la taille de la mémoire, quoique relativement importante, risque d'être limitée pour des applications de traitement d'images complexes.

Au total, il apparaît que la rétine proposée surmonte chacune des limitations rencontrées sur les systèmes antérieurs tels qu'ANNA, I1000 et MAPP2200.

7. Conclusion

Dans cet article, nous avons étudié l'apport des architectures connexionnistes aux rétines électroniques dans la perspective de leur intégration dans des technologies de circuits intégrés futures.
Nous avons proposé une méthode d'évaluation et de prédiction des performances originale permettant la comparaison quantitative d'architectures connexionnistes. Nous l'avons validée à partir d'exemples de réseaux connexionnistes sur une architecture existante, CNAPS, et nous avons montré que les performances accessibles sont comparables à celles d'architectures connexionnistes analogiques à technologie équivalente.

Nous avons proposé une architecture de rétine électronique basée sur cette architecture connexionniste. Elle présente l'avantage d'être programmable, et de pouvoir être adaptée à une large classe de situations, tout en ayant des performances élevées.

Les perspectives de ces travaux se situent sur deux plans, architecture et électronique. La méthodologie d'évaluation de performances que nous avons élaborée peut-être appliquée à d'autres classes d'architectures. Par ailleurs, l'architecture connexionniste proposée peut être utilisée pour la simulation d'autres modèles de réseaux connexionnistes que les PMC, comme les RBF (Radial Basis Function), et aussi pour le traitement d'images de bas niveau [Por95].

Remerciements

Nous tenons à remercier l'Institut d'Electronique Fondamentale et le Réseau Doctoral en Architecture de Machines et des Systèmes Informatique, pour l'aide apportée dans la réalisation de ces travaux, et Luc Gaborit, étudiant de l'UPMC, pour sa collaboration.

Références

[APMP93] Arnulfo Azcarraga, Hélène Paugam-Moisy, and Didier Puzena, *An incremental neural classifier on a mimd computer*, Research Report 93-40, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, Unité de Recherche Associée au CNRS 1398, December 1993.

[AvSB+ 96] Xavier Arreguit, F.André van Schaik, François V. Bauduin, Marc Bidiville, and Eric Raeber, *A cmos motion detector system for poiting devices*, IEEE Journal of Solid-State Circuits **31** (1996), no. 12, 1916–1921.

[AW96] Chye Huat Aw and Bruce A. Wooley, *A 128x128-pixel standard-cmos image sensor with electronic shutter*, IEEE Journal of Solid-State Circuits **31** (1996), no. 12, 1992–1930.

[Ber97] Thierry Bernard, *Rétines artificielles : quelle intelligence au sein du pixel ?*, Calculateurs Parallèles (1997).

[BSB+ 91] B. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L. J ackel, *An analog neural network processor with programmable topology*, IEEE Journal of Solid-State Circuits **26** (1991), no. 12, 2017–2025.

[BSB+ 92] B. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L. J ackel, *Hardware requirements for neural network pattern classifiers*, IEEE Micro **12** (1992), no. 1, 32–40.

[BZD92] Thierry Bernard, Bertrand Zavidovique, and Francis Devos, *The ncp retina: an imager, a halftoner and a micro-gained array processor on the same chip*, European Solid-State Circuits Conference, September 1992, pp. 159–162.

[DAR88] DARPA (ed.), *Darpa neural network srudy*, AFCEA International Press, November 1988.

[Fag94] F. Faggin, *Commercialization of neural networks*, 1994, Invited Talk in Micro-Neuro'94.

[GG96] Bertrand Granado and Patrick Garda, *Evaluation of the two differents interconnection networks of the cnaps neurocomputer*, Proceedings of ICANN'96, Juillet 1996.

[GRR+ 88] P. Garda, A. Reichart, H. Rodriguez, F. Devos, and B. Zavidovique, *Une rétine électronique automate cellulaire*, Traitement du Signal, vol. 5, 1988, pp. 435–449.

[GRS91] Cécile Germain-Renaud and Jean-Paul Sansonnet, *Les ordinateurs massivement parallèles*, Armand Colin, 1991.

[GTK+ 91] Matthew Griffin, Gary Tahara, Kurt Knorpp, Ray Pinkham, Bob Riley, Dan Hamerstrom, and Eric Means, *An 11 million transistor digital neural network execution engine*, Proceedings of IEEE International Solid-State Circuits Conference, 1991.

[Ham90] Dan Hammerstrom, *A vlsi architecture for high-performance, low-cost, on-chip learning*, Proceedings of Internationnal Join Conference on Neural Network, 1990, pp. 537 – 544.

[HP92] John L. Hennessy and David A. Patterson, *Architecture des ordinateurs : Une approche quantitative*, Paris : McGraw-Hill, 1992.

[JISF92] Christer Jansson, Per Ingelhag, Christer Svensson, and Robert Forchheimer, *An adressable 256x256 photodiode image sensor array with an 8-bit digital output*, European Solid-State Circuits Conference, 1992, pp. 151–154.

[Kli96] Geoffroy Klisnick, *Etude et réalisation en technologie cmos de circuits d'acquisition de signaux analogiques*, Ph.D. thesis, Université Pierre et Marie Curie - Paris, 1996.

[KPG94] Jacques-Olivier Klein, Hubert Pujol, and Patrick Garda, *Simulation de la machine de boltzmann en temps réel*, Traitement du Signal (1994).

[LBD+90] Y. LeCun, B. Boser, J.S. Denker, D.henderson, R.E. Howard, W. hubbard, and L.J. Jackel, *Handwritten digit recognition with a back-propagation network*, Neural Information Process and System, 1990, pp. 396–404.

[LRMM89] J. Lazarro, S. Ryckebush, M. Mahowad, and C. Mead, *Winner-take-all networks of 0(n) complexity*, Advances in Neural Information Processing Systems (Morgan Kaufmann, ed.), vol. 1, D. Touretzky, 1989.

[Man95] Manual, *Series 150/40 products systems*, Tech. report, Imaging Technology Inc., 1995.

[Mea89] Carver Mead, *Analog vlsi and neural systems*, Addison-Wesley, 1989.

[PA96] John C. Platt and Timothy P. Allen, *A neural network classifier for the i1000 ocr chip*, Advances in Neural Information Processing Systems (Morgan Kaufmann, ed.), vol. 8, D. Touretzky, 1996.

[Por95] Thomas Pornin, *Traitement d'images de bas niveau sur un calculateur parallèle*, Tech. report, Institut d'Electronique Fondamentale - Orsay, 1995.

[Ram92] Ulrich Ramacher, *Synapse - a neurocomputer that synthetizes neural algorithm on a parallel systolic engine*, Journal of Parallel and Distributed Computing (1992), no. 12, 306 – 318.

[RDWL90] D. Renshaw, P.B. Denyer, G. Wang, and M. Lu, *Asic vision*, Custom Integrated Circuits Conference, IEEE Press, May 1990, pp. 7.3.1–7.3.4.

[SBB+92] Eduard Säckinger, Bernhard Boser, Jane Bromley, Yann LeCun, and Lawrence D. Jackel, *Application of the ANNA neural network chip to high-speed character recognition*, IEEE Transaction on Neural Networks **3** (1992), no. 2.

[SG96] Eduard Säckinger and Hans-Peter Graf, *A board system for high-speed image analysis and neural networks*, IEEE Transaction on Neural Networks **7** (1996), no. 1.

[SMST95] A. Simoni, F. Maloberti, A. Sartori, and G. Torelli, *A 256x256-pixel optical sensor architecture with 32 algorithmics a/d converters*, Proceedings of Twenty-first European Solid-State Circuits Conference (Lille (France)), September 1995, pp. 338–341.

[Åst93] Anders Åström, *Smart image sensors*, Ph.D. thesis, Linköping University - Department of Electrical Engineering, Linköping, Swenden, 1993.