Architecture matérielle pour implantation de réseaux de neurones en temps réel

Sonia KHATCHADOURIAN[1], Narayanan RAMANAN[1], Jean-Christophe PRÉVOTET[2], Lounis KESSAL[1]

[1]Laboratoire ETIS/ENSEA CNRS UMR 8051
6 av. du ponceau, 95014 Cergy-Pontoise, France

[2]Laboratoire IETR/INSA CNRS UMR 6164
20 av. des buttes de Coësmes , 35043 Rennes, France

sonia.khatchadourian@ensea.fr, narayanan.ramanan@ensea.fr,
jean-christophe.prevotet@insa-rennes.fr, lounis.kessal@ensea.fr

Résumé – Le travail effectué se situe dans le cadre du calcul de réseaux neuronaux dans des domaines d'applications à très fortes contraintes temporelles. L'article propose une architecture originale tirant un bénéfice maximum des ressources embarquées dans les circuits reconfigurables actuels. Un réseau de neurones de type Perceptron Multicouches peut être calculé en un temps de l'ordre de la microseconde, et ce, en occupant un nombre très limité de ressources logiques.

Abstract – The study presented is in the field of the neural networks computation for applications of strong time constraints. In this communication, we propose an original architecture which strives to reduce the amount of logic to be utilized. A neural network of multilayer perceptron type can be computed in an execution time of the microsecond order while occupying few logic ressources.

1 Introduction

Depuis leur étude au milieu du siècle dernier, de nombreuses applications ont tiré profit des propriétés des réseaux de neurones dans des domaines variés tels que la reconnaissance de formes, le contrôle non destructif, la classification, la robotique, etc. La plupart de ces applications n'a pas de contraintes de latence forte pour s'exécuter et un simple processeur peut généralement convenir parfaitement à l'exécution de l'algorithme neuronal. Il existe cependant des applications à latence réduite, nécessitant un calcul ultra performant. Dans ce cas, il est généralement envisagé de s'orienter vers des solutions matérielles de type ASIC (Application Specific Integrated Cicuit). Aujourd'hui, la technologie reconfigurable basée sur des circuits comme les FPGAs (Field Programmable Gate Array) a atteint un niveau de performances tel, qu'elle constitue une alternative intéressante aux ASICs. En effet, ces circuits reconfigurables permettent de tirer profit du parallélisme intrinsèque des réseaux neuronaux [1] et ouvrent des perspectives d'utilisation nouvelles, dans des domaines, jusque là, privés d'une telle puissance de calcul [2]. Dans le cadre de ce travail, nous avons cherché à élaborer une architecture neuronale permettant d'exécuter des réseaux de neurones de type perceptron multicouche à l'échelle de la microseconde. L'autre contrainte majeure était de réduire au maximum l'utilisation de ressources logiques, cette architecture étant éventuellement amenée à ne constituer qu'une partie d'un circuit global.

2 Structure du réseau de neurones

Dans cet article, nous considérerons une structure très classique de réseau neuronal [3] appelée PMC (Perceptron Multicouches) décrit en figure 1.

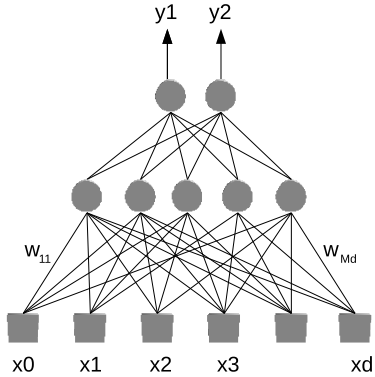

FIGURE 1 – Structure d'un perceptron à deux couches

Ce dernier consiste en une succession de couches constituées d'unités neuronales, lesquelles possèdent une fonction d'activation, linéaire ou non, et différentiable. Chaque neurone, à l'intérieur d'une couche, reçoit des signaux provenant de la couche précédente, effectue un calcul, et transmet le résultat à la couche suivante, si elle est présente. Il n'existe pas d'interconnexions entre les neurones situés à l'intérieur d'une même

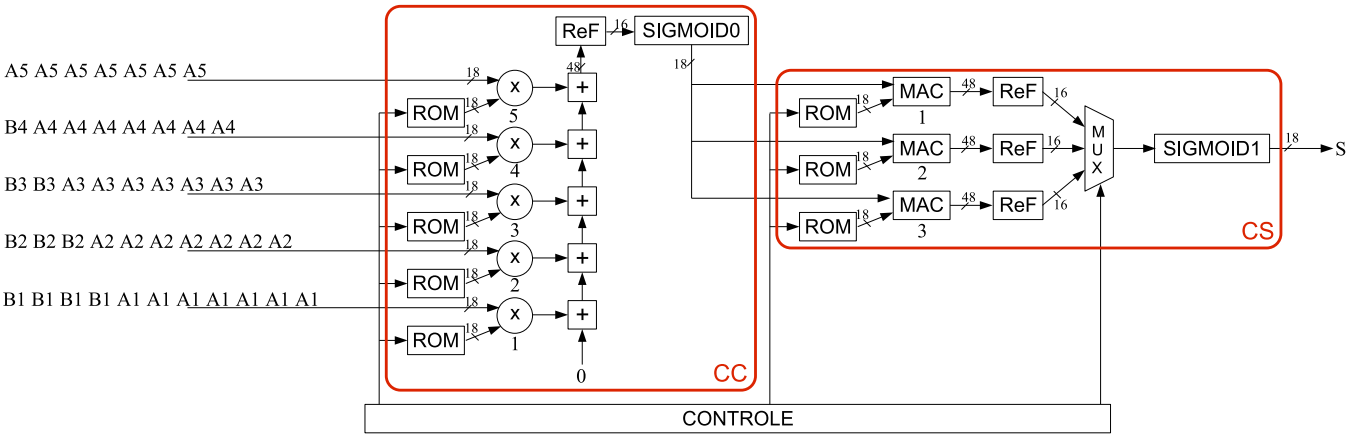

FIGURE 2 – Architecture matérielle implantant un réseau de neurones à 5 entrées, 7 neurones cachés et 3 sorties.

couche : les activations des différents neurones sont seulement propagées de la couche d'entrée vers la couche de sortie à travers tous les neurones constitutifs du réseau. La couche d'entrée collecte les variables d'entrée tandis que la couche de sortie produit les résultats.

Un PMC est une application de R^N dans R^M basée sur la composition de fonctions non-linéaires, représentées par les neurones de la couche cachée. La sortie du réseau est donnée par l'équation 1.

$$y_j^{(L)} = f(v_j^{(L)}) = f(\sum_{i=1}^{N_{L-1}} W_{ij}^L x_i^{L-1}) \qquad (1)$$

Dans l'équation 1, $W_{ij}^{(L)}$ est le poids entre le neurone i de la couche $(L-1)$ et le neurone j de la couche (L), et $x_i^{(L-1)}$ est la sortie du neurone i dans la couche $(L-1)$. f est la fonction d'activation différentiable et non linéaire de type sigmoïde.

3 Architecture matérielle

Dans cet article nous tirons parti de la nature intrinsèque des réseaux de neurones : leur parallélisme en fait d'excellents candidats pour une implantation matérielle. L'architecture proposée dans cet article est flexible. Tout en optimisant le nombre de ressources matérielles nécessaires, elle peut être adaptée à différentes applications ayant besoin de différentes configurations d'un PMC.

3.1 Description

Un exemple d'implantation est décrit dans la figure 2 : un réseau de neurones de $I = 5$ entrées, $H = 7$ neurones sur la couche cachée et $O = 3$ sorties. Les données entrantes sont envoyées en série au réseau de neurones et arrivent par paquets A, B, etc. Les paquets contiennent les I éléments de chaque entrée. L'architecture se décompose en deux niveaux : le calcul de couche cachée (CC) et celui de la couche de sortie (CS).

Dans cet exemple, les cinq entrées (codées sur 18 bits) arrivent séquentiellement et sont fournies directement aux cinq multiplieurs (18x18) disposés en parallèle. A_j correspond à la $j^{ième}$ entrée de l'ensemble présent alors que B_j correspond à la $j^{ième}$ entrée de l'ensemble suivant. Les entrées sont synchronisées sur l'horloge et sont maintenues pendant H cycles d'horloge. Les multiplieurs permettent de réaliser le produit des entrées avec leurs poids respectifs stockés dans des mémoires ROM (Read Only Memory). Chacune de ces mémoires contient l'ensemble de poids reliant une entrée à tous les neurones de la couche suivante. A chaque cycle d'horloge, une accumulation est réalisée et la somme transite à travers tous les additionneurs. La sortie du dernier additionneur est transmise à un opérateur de remise en forme (ReF) afin de transformer le signal initialement codé sur 48 bits en un signal sur 16 bits. Ce signal adresse ensuite une mémoire tabulant la fonction sigmoïde d'activation (SIGMOID0) en 65536 valeurs de 18 bits.

Les résultats de la couche cachée sont transmis à la couche de sortie pour faire les multiplications-accumulations (MAC) correspondantes. Chaque unité MAC calcule la somme pondérée associée à un neurone de sortie. Dans l'étage CS, chaque mémoire ROM contient l'ensemble des poids reliant une sortie aux différents neurones de la couche cachée. Après l'opération MAC, les données sont à nouveau mises en forme sur 16 bits afin d'adresser la seconde mémoire sigmoïde d'activation. Le flot de données est multiplexé de manière à réduire l'utilisation des mémoires. L'étage de contrôle permet le séquencement des différentes opérations dans l'architecture. Il va permettre la synchronisation des multiplications dans la couche cachée par le biais de la commande de lecture dans la ROM. Le procédé est le même en ce qui concerne les unités MAC de la couche de sortie. Enfin, c'est aussi l'étage de contrôle qui commande le multiplexage des données en sortie du réseau de neurones.

Le temps d'exécution du réseau de neurones, à partir de l'occurrence de la première entrée, est égal à $I + H + c + O$ cycles d'horloge, c étant une constante dépendant de la latence des différentes opérations (dans notre cas $c = 6$). Cette constante peut être revue en fonction des différentes performances à atteindre, notamment la fréquence maximale d'utilisation.

TABLE 1 – Performances et ressources de quelques circuits implantés

Taille du réseau Ni*Nh*No	Ressources logiques occupées/ disponibles	Blocs DSP dédiés utilisés/ disponibles	Blocs mémoire utilisés/ disponibles	Temps de traitement
20x20x3	477/49152 (1%)	23/96	151/240	408 ns
20x255x3	477/49152 (1%)	23/96	151 /240	2,36 μs
50x255x3	477/49152 (1%)	53/96	181/240	2,62 μs
50x255x8	593/49152 (1%)	58/96	186/240	2,65 μs

3.2 Avantages de cette architecture

Un avantage majeur de cette architecture réside dans le fait que de simples compteurs permettent de gérer le flot de données à travers les couches du réseau. Le contrôle devient alors très simple à mettre en œuvre et très peu coûteux en ressources. Ces compteurs sont limités par les paramètres du réseau de neurones. La gestion des ROMs se fait sur chaque coup d'horloge et l'incrémentation est limitée par le nombre de neurones sur la couche cachée H. Ceci est vrai pour les ROMs associées aussi bien à la couche cachée que la couche de sortie. La gestion du multiplexeur se fait aussi sur chaque coups d'horloge et son incrémentation est limitée par le nombre de sorties O.

Dans l'architecture proposée, le nombre de multiplieurs requis pour calculer l'ensemble du réseau est fixé à $I + O$, où I est le nombre d'entrées et O est le nombre de sorties. Il est donc indépendant du nombre de neurones sur la couche cachée. L'architecture est entièrement scalable. L'ajout d'un neurone sur la couche cachée ne demande qu'une incrémentation de la valeur de compteur. L'addition d'une entrée ne requiert que l'ajout d'un multiplieur, d'une mémoire et d'un additionneur. L'ajout d'une sortie nécessite une nouvelle ROM ainsi qu'une unité MAC supplémentaire.

D'autre part, il est à noter l'utilisation d'une unique mémoire permettant de tabuler les valeurs de la fonction d'activation pour l'ensemble des neurones de la couche cachée. Ceci permet une grande liberté dans le choix de la fonction d'activation du PMC indépendamment de l'architecture. L'ajout du multiplexeur en sortie du réseau de neurones permet l'économie de mémoire dédiée aux fonctions d'activation. Cette astuce impose la sérialisation des sorties, elles arrivent ainsi sur le même bus. Bien entendu, cette solution n'est applicable que dabs le cas où le nombre de sorties est inférieur ou égal au nombre de neurones sur la couche cachée. Si ce n'est pas le cas, la latence sur laquelle les sorties des MACs de la couche de sortie n'est pas suffisante pour la sérialisation des sorties du réseau de neurones.

4 Performances obtenues

L'architecture a entièrement été simulée et implantée dans un circuit FPGA de la famille Xilinx Virtex4 (xc4vlx100). Ce type de circuit reconfigurable est doté de ressources dédiées telles que les blocs mémoire ou les blocs DSP permettant l'exécution d'opérations comme les MAC très efficacement. Les temps de traitement pour différentes configurations de réseau sont donnés dans le tableau 1. La fréquence du fonctionnement retenue a été fixée à 120 MHz.

4.1 Ressources

Aujourd'hui, la plupart des circuits reconfigurables sont dotés de ressources dédiées supplémentaires permettant d'accélérer les traitements tels que les multiplications-accumulations. Ils mettent en œuvre également de nombreux blocs mémoire. L'architecture proposée a été conçue afin d'utiliser massivement ce type de ressources. Ceci est d'autant plus intéressant, que les ressources logiques économisées peuvent alors être utilisées pour implanter d'autres types d'algorithmes comme des prétraitements des données présentées au réseau, des opérations de normalisation, de post-traitement par exemple. Les ressources (logiques puis dédiées) requises sont présentées dans le tableau 1.

Au regard de ce tableau, il est important de noter que quelque soit la configuration, l'occupation logique est très faible. Il est également clair que les ressources dédiées sont très massivement utilisées (blocs mémoire + blocs DSP). Les performances obtenues sont donc très liées à la quantité de ces ressources. On constate que les multiplications et additions associées aux entrées sont exécutées par DSPs. De la même façon, les MACs associées aux neurones de sortie sont aussi exécutées par des DSPs. C'est ainsi que l'occupation des DSPs du FPGA est de $I + O$. De la même façon, les poids associés aux liaisons entre chaque entrée et les neurones de la couche cachée sont stockés dans les ROMs associées à chaque entrée. Il en est de même pour les poids associés aux liaisons entre les neurones de la couche cachée et les neurones de sortie. En cas de dépassement des ressources dédiées, il est également possible de réaliser les opérations MAC à l'aide des cellules logiques restantes, au détriment de l'occupation du circuit. Ceci est aussi vrai pour l'implantation des mémoires ROM à partir des cellules logiques.

Le nombre de neurones sur la couche cachée n'influence pas l'occupation logique.

4.2 Temps d'exécution

Les paramètres du réseaux de neurones comme le nombre d'entrées, de neurones sur la couche cachée ou de sorties ont un impact non négligeable sur le temps de traitement. En effet, le nombre d'entrées influent sur le nombre d'opérations à exécuter du fait de la façon pipelinée dont les entrées sont traitées. D'autre part, chaque entrée est reproduite H fois sur la couche cachée afin d'être traitée dans son intégralité ce qui jus-

tifie de la dépendance du nombre de neurones sur la couche cachée sur le temps de traitement. Ensuite les H résultats de la couche cachée sont transmis à la couche de sortie pour finaliser le calcul. Les sorties sont calculées en parallèle et elles sont maintenues sur H cycles d'horloge. Le fait que le temps de traitement soit aussi dépendant du nombre de sortie est dû au multiplexeur placé avant la dernière fonction d'activation. D'une façon générale, le temps requis à l'architecture proposée pour obtenir toutes les sorties après l'incidence de la première entrée peut être évalué de la façon suivante : $I + H + c + O$ cycles d'horloge, c étant une constante dépendant de la latence des différentes opérations.

5 Conclusion

Dans cet article, nous avons présenté une architecture matérielle permettant d'implanter des réseaux de neurones de type perceptron multicouches. Le circuit présenté exploite de manière optimale la technologie des circuits reconfigurables actuels. L'architecture présentée tire un profit maximum des ressources intrinsèques disponibles dans de nombreux FPGAs actuels. Elle possède une occupation des ressources logiques très faible, ce qui permet d'utiliser ce type d'algorithmes en combinaison avec d'autres types de traitements [4]. D'autre part, nous avons confirmé la faisabilité du portage d'applications de calcul temps-réel complexes sur des circuits reconfigurables actuels.

Références

[1] A.R. Omondi and J.C. Rajapakse. *Fpga Implementations of Neural Networks*. Springer, 2006.

[2] J.-C. Prevotet, B. Denby, P. Garda, B. Granado and C. Kiesling. *Moving NN Triggers to Level-1 at LHC Rates*. Nuclear Inst. and Methods in Physics Research, Elsevier, 2003.

[3] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[4] S. Khatchadourian and J.C Prévotet and L. Kessal. *A Neural Solution for the Level 2 Trigger in Gamma Ray Astronomy*. ACAT, Proceedings of Science, 2007.