Méthodologie de Conception d'une Implémentation Matérielle d'un algorithme par l'Analyse en Composantes Indépendantes pour la Séparation de Signaux

Mouloud OUNAS[1] , Salim CHITROUB[2] et Rachida TOUHAMI[3]

[1,3]Laboratoire d'Instrumentation, Département Instrumentation
[2]Laboratoire de Communication Parlée et de Traitement des Signaux, Département Télécommunications
Faculté d'Electronique et d'Informatique
Université des Sciences et de la Technologie Houari Boumediène
BP. 32, El – Alia, Bab – Ezzouar, Alger, 16111, ALGERIE.
Emails: [1]ounas_mouloud@yahoo.fr [2]salim_chitroub@yahoo.fr [3]rtouhami@yahoo.fr

Résumé – Dans cet article on a proposé une méthodologie de conception d'une implémentations matériels afin d'optimisé un algorithme d'apprentissage des réseaux de neurones en utilisant l'analyse en composantes indépendantes (ACI) pour la séparation des signaux. L'objectif de la conception est de permettre la réalisation d'une architecture d'implémentation qui consomme mois de ressources matériels dans les circuits FPGA. L'utilisation du critère de la règle d'arrêt sur l'ajustement des poids synaptiques nous a permis l'optimisation d'une architecture matérielle tour en préservant des performances minimales de la séparation de signaux. L'optimisation de l'algorithme par l'analyse en composantes indépendantes nous a permit d'obtenir une architecture matérielle moins complexe avec un ajustement efficace des poids synaptiques.

Abstract – In this paper, we proposed a hardware circuit design methodology that optimises the learning algorithm of a neural network by independent component analysis (ICA) for blind signal separation. The aim of the proposed approach is to favour the achievement of an implementation architecture of FPGA circuits that consumes less hardware resources. The use of the stopped rule criterion on the updated weight adjustments allows the optimisation of the hardware architecture while keeping optimal performance of blind signal separation. The optimisation of the algorithm of independent component analysis yielded obtaining a less complex hardware architecture with an efficient adjustment of the updated weights.

1. Introduction

La Technique d'analyse en composantes indépendantes (ACI) est utilisée pour recouvrir des signaux supposés inconnues a priori et mutuellement statistiquement indépendants [1]. La séparation des signaux s'effectue à partir de signaux de mélanges connus, issus de plusieurs sources de capteurs. L'algorithme de l'ACI exige un nombre important d'opérations arithmétiques durant son traitement d'où la nécessité de disposer d'un opérateur mathématique très performant capable de supporter la puissance de calculs exigée par ces opérations. Dans cette optique, l'algorithme de séparation des signaux effectue l'apprentissage par les réseaux de neurones artificiels [2] en ajustant ses poids synaptiques par l'ACI en vue de minimiser la distribution gaussienne dans les signaux de mélanges à l'aide les propriétés statistiques du signal [3].

Actuellement, les chercheurs visent à contrôler la complexité des calculs de l'algorithme par l'ACI afin d'exécuter des applications pour le traitement de la séparation de signaux. Ils ont alors envisagé la réalisation d'applications par l'ACI, qui utilisent des implémentations logiciels et matérielles grâce a l'avancée technologique dans la fabrication des circuits intégrés VLSI en microélectronique. Pour le traitement de l'algorithme en temps réel, l'implémentation logicielle présente une limitation en vitesse pour le traitement des calculs car il agit sur une architecture de type Von Neumann supportant des opérations séquentielles dans le traitement des calculs. La solution envisagée pour le traitement en temps réel de l'algorithme est l'implémentation dans la technologie VLSI qui permet la fabrication des circuits intégrés ASICs susceptibles de réaliser les calculs avec de très grandes performances. Néanmoins, cette implémentation a pour effet d'augmenter les temps et les coûts de conception. Une solution alternative est offerte par les circuits FPGA programmables qui permettent de diminuer les temps de conception et de produire des prototypes a moindre coût de fabrication.

Le nombre de ressources présentes dans un circuit FPGA étant limitées, des efforts doivent être consentis dans le but de minimiser ces ressources à travers une méthodologie de conception simple proposée dans cet article, et qui sera appliquée aux architectures matérielles.

Cette méthodologie est basée sur l'illustration d'un compromis qui réalise une optimisation d'une architecture matérielle tout en préservant les performances minimales de séparation des signaux. Toujours dans l'optique de fournir une architecture matérielle simple, nous avons utilisé la représentation des nombres à point fixe [6],[7],[8] et de l'approximation par la fonction segmentation [9], de la fonction d'activation sigmoïde non linéaire des réseaux de neurone. La conception de notre architecture matérielle a été réalisée en langage VHDL [10] pour décrire et simuler des circuits matériel d'architectures très complexes. A cet effet, les constructeurs de circuit FPGA ont introduits une plate forme de développement en VHDL, offrant des outils de synthèse, de placement et de routage des architectures matériels dans les circuits FPGA.

2. Mise en œuvre de l'algorithme de l'analyse en composantes indépendantes

L'algorithme (ACI) suivant est appliqué à la séparation instantanée de signaux. Soit m le nombre de signaux xi de mélanges issus des combinaisons linéaires de n sources de signaux s_j supposés mutuellement statistiquement indépendants. L'expression du signal x_i est donnée par l'équation :

$$x_i = \sum_{j=1}^{n} A_{ij} S_j \text{ , (i=1, 2,...,m)} \quad (1)$$

Où : A_{ij} est le coefficient de mélange inconnu de la j^{eme} source au i^{eme} signal de mélange pour (m≥n). Par simplification, on pose (m=n). La fonction de densité de probabilité est de distribution non gaussienne pour au moins (n-1) sources de signaux s_j. Le coefficient de l'ajustement des poids synaptiques w_{ij} estimé par l'algorithme d'apprentissage, est utilisé pour le calcul de l'entrée synaptique u_i du réseau de neurone dont l'expression est donnée par l'équation:

$$u_j = \sum_{i=1}^{n} w_{ij} x_j \quad (2)$$

La fonction d'activation non linéaire f est utilisée pour le calcul de sortie y_j du réseaux de neurone dont l'expression est donnée par l'équation :

$$y_j = f(u_j) \quad (3)$$

Le calcul de l'ajustement des poids synaptiques w_{ij} utilise l'approche de la descente du gradient naturel [1] dont l'expression est donnée par l'équation:

$$\Delta W = \mu\left[W^{-T} + \varphi(u)x^T\right] \quad (4)$$

Où ; φ désigne la fonction non linéaire d'apprentissage, μ est le facteur d'apprentissage, w est la matrice qui contient les coefficients des poids synaptiques, x est le vecteur de signaux de mélange, et u est le vecteur des entrées synaptiques u_i .

AMARI [1] et CARDOSO [4] ont proposé une modification de la relation du calcul d'apprentissage de l'ajustement des poids synaptiques en évitant le calcul complexe de la matrice (W^{-T}) utilisé dans l'équation (4). La nouvelle expression de l'équation d'apprentissage des poids synaptiques est donnée par l'équation:

$$\Delta W = \mu\left[I + \varphi(u)u^T\right]W \quad (5)$$

Où ; I désigne la matrice unité, d'ou le calcul du coefficient d'ajustement des poids synaptiques wij est donné pour (i=1,2,…,n) et (j=1,2,…,n) selon :

$$W_{ij}^{+} = W_{ij} + \mu\left[1 + \sum_{k=1}^{n} \varphi(u_k)u_k\right]W_{ij} \quad (6)$$

Bell et Sejnowski [5] ont proposé une approche qui maximise le transfert de l'information entre l'entrée et la sortie du réseaux de neurone, d'ou l'expression de la fonction non linéaire d'apprentissage est donnée par l'équation (7) :

$$\varphi(u_i) = \frac{f''(u_i)}{f'(u_i)} \quad (7)$$

Où : f désigne la fonction d'activation des réseaux de neurone dont on assigne l'expression de la fonction sigmoide :

$$f(u_i) = \frac{1}{1 + e^{-u_i}} \quad (8)$$

D'où l'expression de φ(ui) devient :

$$\varphi(u_i) = (1 - 2f(u_i)) \quad (9)$$

3. Mise en œuvre du calcul d'implémentation de l'algorithme

3.1 Calcul de l'ajustement des poids synaptiques :

Dans cette étape, l'ajustement des poids w_{ij} se détermine à travers les calculs dans les boucles d'itérations (ite) jusqu'à la convergence. Le signal ε permet l'illustration de l'information sur l'arrêt de la convergence des poids synaptiques selon :

$$\varepsilon = \sum_{i=1}^{n} \varphi(u_i)u_i \quad (10)$$

Avec ; $\varepsilon < \varepsilon_{max}$, et ε_{max} = Cte. Où ; ε_{max} désigne le nombre maximal de l'information sur le critère d'arrêt de l'ajustement des valeurs correctes des poids synaptiques w^{+}_{ij} dont le calcul devient, pendant l'ajustement des poids:

Pour (i=1,2,…,n) et (j=1,2,…,n),

$$W_{ite\,ij} = W_{(ite-1)ij} + \mu(1 + \varepsilon)W_{(ite-1)ij} \quad (11)$$

et après l'ajustement des poids, l'équation (11) devient :

$$W_{c_k ij} = W_{(c_k-1)ij} + \mu(1+\varepsilon)W_{(c_k-1)ij} \quad (12)$$

Où : $(1 \le ite \le c_k)$ et $(1 \le k \le M)$; M désigne le maximum des échantillons des signaux d'entrées ; c_k indique le nombre des itérations nécessaire pour la convergence de l'ajustement des poids synaptiques de chaque valeur de l'échantillon k à l'entrée. Les nouvelles valeurs de l'ajustement des poids synaptiques normalisées sont donnes par l'expression de l'équation :

$$W_{c_k ij} = \frac{W_{c_{k-1} ij} + \mu(1+\varepsilon)W_{c_{k-1} ij}}{W_{c_k ij\,\max}} \quad (13)$$

Où : $W_{c_k ij\,\max}$ est la valeur maximale des poids synaptiques ajusté avant normalisation.

3.2 Calcul de l'accumulation des poids synaptiques ajustés :

Dans cette étape, l'accumulation des poids synaptiques ajustés W_{cij} se calcule à travers les boucles d'itérations pour chaque valeur de l'échantillon k à l'entrée jusqu'à (k=M). L'expression de W_{cij} est donné par l'équation suivante :

$$W_{cij} = \sum_{k=1}^{M} W_{c_k ij} \quad (14)$$

3.3 Calcul de l'ajustement des poids synaptiques finaux :

L'ajustement des poids synaptiques finaux est déterminé par l'expression de l'équation :

$$Wf_{ij} = \frac{W_{cij}}{M} \quad (15)$$

4. Mise en œuvre de l'adéquation algorithme et architecture

L'architecture d'implémentation est divisée en deux parties, une partie est destinée au bloc de traitement et l'autre partie pour le bloc de contrôle. Pour le bloc de traitement, nous avons implémenté une architecture pour le traitement d'un seul neurone afin de minimiser le nombre de ressources matériel présentes dans l'architecture du circuit d'implémentation. Après la mise en œuvre de l'étape de calculs d'implémentations, trois unités d'architectures principales pour le traitement du calcul de l'algorithme ont été présentées. Une unité d'implémentation pour le bloc de sortie du neurone, une unité d'implémentation pour le bloc de l'ajustement des poids synaptiques, et une unité d'implémentation pour le bloc de l'ajustement des poids synaptiques finaux. L'ensemble des trois unités de blocs d'implémentation sont interconnecté à travers un bloc de mémoire RAM par l'intermédiaire des signaux bidirectionnels et unidirectionnel pour stocker les échantillons des signaux d'entrée, les valeurs des poids initiales, et les valeurs de l'ajustement des poids synaptiques. La Figure 1 montre la structure du schéma d'implémentation d'un seul neurone.

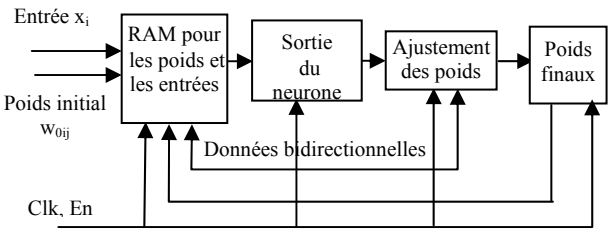

FIG. 1-Schéma d'implémentation d'un seul neurone

Les signaux de contrôle d'horloge (Clk) et de validation En, générés par le bloc de contrôle, effectuent la commande de validation des signaux horloge dans les différents blocs d'implémentation du bloc de traitement. L'architecture d'implémentation du bloc de contrôle a été réalisée sur la base d'une méthodologie de traitement des machines d'états.

4.1 Bloc d'implémentation de sortie du neurone

Le bloc d'implémentation de sortie du neurone effectue le calcul de l'entrée synaptique, et le calcul de la fonction d'activation non linéaire sigmoïde qui représente la valeur du signal de sortie du neurone. La Figure 2 illustre le schéma d'implémentation du bloc de sortie du neurone.

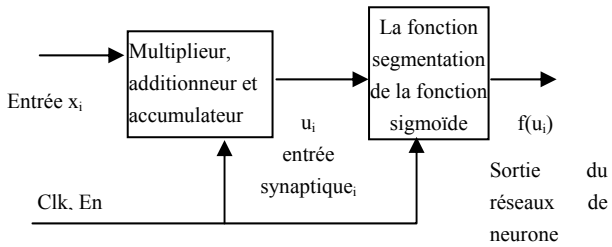

FIG. 2 -Schéma d'implémentation de sortie du neurone

4.2 Bloc d'implémentation de l'ajustement des poids synaptiques

Le bloc d'implémentation de l'ajustement des poids synaptiques effectue le calcul de la fonction non linéaire d'apprentissage du réseau de neurone, le calcul de la valeur du signal ε qui représente le test du critère d'arrêt permettant l'ajustement des poids synaptiques, et le calcul de la valeur de l'ajustement des poids synaptiques. La Figure 3 illustre le schéma d'implémentation du bloc de l'ajustement des poids synaptiques.

4.3 Bloc d'implémentation de l'ajustement des poids synaptiques finaux

Le bloc d'implémentation de l'ajustement des poids synaptiques finaux réalise une implémentation pour le calcul de l'accumulation des poids synaptiques ajustés, et

le calcul de la valeur des poids synaptiques finaux. La Figure 4 illustre le schéma d'implémentation du bloc de l'ajustement des poids synaptiques finaux.

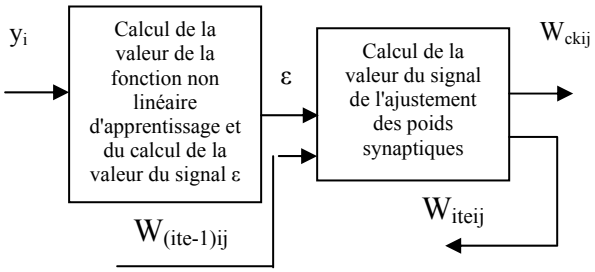

Fig.3. Schéma d'implémentation du bloc de l'ajustement des poids synaptiques

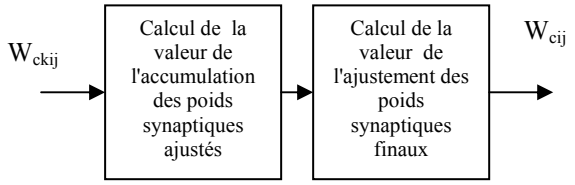

Fig.4. Schéma d'implémentation du bloc de l'ajustement des poids synaptiques finaux

5. Effet de l'architecture d'implémentation sur le circuit matériel

Nous avons utilisé la plate forme de développement ISE 6.1 pour la conception de l'implémentation matérielle sur le circuit FPGA de la compagnie Xilinx. L'implémentation du bloc de traitement a été décrite en VHDL dans un niveau structurel. Par contre, l'implémentation du bloc de contrôle a été décrite en VHDL dans un niveau comportemental. La valeur 1000 est le nombre des échantillons des signaux d'entrée qui ont été utilisés pour le traitement des calculs de l'algorithme dans le cas de 2 signaux d'entrées et 2 signaux de sorties. L'implémentation a été simulée grâce a l'outil ModelSim 5.8 . Le tableau 1 montre le résumé des résultats de simulation de la synthèse du circuit Xilinx Virtex II XC2V8000. La valeur N=8 bits (dimension des nombres binaires dans la représentation a point fixe) a été utilisée pour fournir une précision minimale sur les calculs des nombres du traitement de l'algorithme.

Unité	coût (slice)	coût (flip flops)	coût (Input LUT)	coût (IOB)	Temps (cycles)	Fréquence Max (MHz)
Sortie du neurone	111	185	73	54	12	186.376
Poids synaptique	113	199	41	43	11	185.580

TAB.1- Résultat de simulation de la synthèse du circuit Xilinx Virtex II XC2V8000

6. Conclusion

Le contrôle par le critère d'arrêt de l'ajustement des poids synaptiques nous a permis de mettre en évidence une méthodologie de conception dans les calculs d'implémentation de l'algorithme. Ainsi, des calculs d'optimisation ont été effectués dans le traitement de l'algorithme. L'optimisation effectuée nous a permis de mettre en œuvre une adéquation algorithme et architecture pour une architecture d'implémentation de complexité très réduite pour la séparation de signaux. La méthodologie employée pour l'implémentation matérielle de l'algorithme par l'analyse en composantes indépendantes a fourni des résultats de simulation avec des valeurs de performances permettant la réduction de la consommation des ressources dans les circuits FPGA. Ceci nous permettra de traiter un grand nombre d'échantillons d'entrées dans le circuit FPGA. D'où l'augmentation de la fréquence d'échantillonnage des signaux d'entrées pour des applications de séparation de signaux de traitement en temps réel.

Références

[1] Andrej Cichocki and Shun-Ichi Amari. *Adaptive Blind Signal and Image Processing, Learning Algorithms and Applications*. John Wiley, 2002.

[2] Yuhen Hu and Jenq-Nenghwang. *Handbook of Neural Network Signal Processing*. CRC Press, 2001.

[3] Aapo Hyvarinen, Juha Karhunen and Erkki Oja. *Independent Component Analysis*. John Wiley, 2001.

[4] J.F. Cardoso. *Blind Signal Separation; Statistical Principles*. Pro. of the IEEE , vol.86, n° 10, pp.2009-2025, 1998.

[5] J.Bell and J. sejnowski. *An Information – Maximization Approach to Blind Separation and Blind Deconvolution* .neural computation , vol.7, pp.1129-1159, 1995.

[6] A.B.Lim , R.C. Rajapakse and A.R. Omondi. *Comparative Study of Implementing ICNNs on FPGAs*. Proceedings International Joint Conference on Neural Networks , pp.177-182, 2001.

[7] Zhong Feng Li and Qiuhua. Lin. *FPGA Implementation of infomax BSS Algorithm with Fixed Point Number Representation*. International Conference on Neural Networks and Brains (ICNN&B'05) , vol.2, pp.889-892, 2005.

[8] C.Charoensak, and F.Sattar. *A Single- Chip FPGA Design for Real Time ICA Based Blind Source Separation Algorithm*. in Proc.IEEE International Symposium on Circuits and Systems, pp. 5822-5825, May 2005.

[9] H. Amin , K.M. Curtis and B.R. Hayes-Gill. *Piecewise Linear Approximation Applied to Nonlinear Function of a Neural Network*. IEE Proc. Crcuits Divices syst , 144 (6), pp.313-317, 1997.

[10] Volnei. A.Pedroni. *Circuit Design With VHDL*. MIT Press , London, England, 2004.