# Architecture dédiée pour le traitement temps réel des signaux physiologiques : spécification algorithmique et méthodologie d'implantation sur circuits reconfigurables

Boubaker Mohamed[1], Akil Mohamed[2],Ben Khalifa Khaled[1], Bedoui Mohamed Hedi[1]

[1] Equipe TIM, Laboratoire de Biophysique, Faculté de Médecine de Monastir, Monastir 5019, Tunisie

[2] IGM, Unité Mixte de Recherche CNRS-UMLV-ESIEE UMR 8049
Laboratoire A2SI, Groupe ESIEE, 2 BLD Blaise Pascal, BP 99
93162 Noisy le Grand Cedex, France

**Résumé -** Nous traitons dans cet article l'optimisation de l'implantation sur circuit FPGA d'un réseau de neurones à partir d'une spécification algorithmique sous la forme d'un Graphe Factorisé et Conditionné de Dépendances de Données (GFCDD). Nous proposons une approche d'optimisation automatique de l'implantation d'un réseau de neurone LVQ (Learning Vector Quantification) en exploitant au mieux l'aspect motif répétitif dans l'algorithme neuronal par une méthodologie « Adéquation Algorithme Architecture ». Nous avons pu générer une implantation de ce réseau de neurone LVQ pour des couches de sortie de dimension variable en minimisant le temps de conception.. Dans la phase d'optimisation, nous avons minimisé la consommation des ressources matérielles tout en respectant les contraintes temporelles liées au contexte de l'application exemple choisie : la décision en temps réel sur l'état de vigilance par traitement des signaux physiologiques

**Abstract -**This paper presents an optimization of an FPGA circuit implementation of a neuron network used for physiological vigilance application. It is based on an unified model of Factorized and Conditioned Data Dependence (GFCDD) graphs as well to specify the application algorithm, as to deduce the optimized implementation onto reconfigurable hardware, in terms of graph transformations. An automatic and optimized implementation of the neurons network LVQ (Learning Vector Quantification) has been carried out. The repetitive property of the neuronal algorithm has been exploited, as much as possible, by means of the methodology "Algorithm Architecture Adequation ". The implementation has been carried out with the minimum variable dimension output layer by minimizing the design time, in order to automate the design step. The proposed approach allows also, using the minimum area of the materiel resources in real time condition of the alertness detection based on physiological signals.

## 1. Introduction

Plusieurs méthodes connexionnistes à apprentissage supervisé et non supervisé ont été utilisées pour quantifier l'état de vigilance à partir du traitement des spectres des dérivations du signal électroencéphalographique (EEG) collectées au niveau du cortex du sujet [1-2]. Pour notre part, nous avons utilisé les cartes auto-organisatrices de Kohonen en phase de décision LVQ (Learning Vector Quantization) comme outil de classification de l'éveil et du sommeil [3]. L'approche proposée dans ce papier s'inscrit dans la continuité de ces travaux avec comme objectif de proposer un système totalement embarqué de détection en temps réel de la baisse de vigilance. Dans ce cadre, nous nous sommes intéressés à la problématique de l'implantation de ce modèle neuronal sur un circuit programmable de type FPGA. Différents types d'implantations du LVQ sur circuits FPGA ont été réalisés. Ces implantations du LVQ exigent la mise en œuvre d'une méthodologie associée à un flot de conception au niveau système, pour spécifier l'algorithme et explorer les différentes implantations possibles et ce pour différentes cibles de circuits FPGA.

Dans le cadre de ce travail et afin d'assurer l'implantation d'un modèle LVQ avec les méthodologies exigées pour un fonctionnement embarqué et en temps réel, nous avons adopté une démarche basée sur la méthode 3A : Adéquation Algorithme Architecture (méthodologie formelle développée à l'INRIA dans le cadre du projet AOSTRE). Cette méthodologie est basée sur un formalisme d'hypergraphes pour modéliser l'algorithme, l'architecture et l'implantation de l'algorithme sur l'architecture. Une implantation optimisée est obtenue par transformation de graphes [4]. Le flot de conception retenu est basé sur l'extension de méthodologie AAA, aux circuits et le logiciel associé SynDEx-IC [5] (flot développé à l'ESIEE en collaboration avec l'INRIA). Cette extension de AAA permet de générer des circuits synthétisables optimisés à partir de la même spécification algorithmique. L'architecture du circuit est obtenue par transformation de graphes, laquelle permet une génération automatique du VHDL synthétisable. Ces transformations comportent une phase d'optimisation. Il s'agit à l'aide d'heuristique de rechercher automatiquement une solution qui permet d'implanter l'algorithme sous des contraintes de latences et de surfaces [5].

Le parallélisme intrinsèque des réseaux de neurones artificiels et particulièrement du LVQ a motivé le choix de ce flot unifié de conception pour explorer et implanter cet algorithme sur un support de type FPGA tout en cherchant à optimiser la consommation des ressources et à respecter la contrainte temps réel.

Cet article est organisé comme suit, après une description succincte du contexte de notre application et de l'algorithme LVQ, nous décrivons l'extension de la de la méthodologie AAA aux circuits; ainsi que le logiciel de CAO au niveau système qui lui est associé : le logiciel SynDEx-IC. Nous décrivons l'approche que nous avons développée, pour spécifier l'algorithme de notre application, exploiter l'heuristique d'optimisation et générer automatiquement l'implantation matérielle. Nous présentons les résultats obtenus, ainsi qu'une conclusion et les perspectives de ce travail.

## 2. Contexte de l'utilisation du LVQ pour la détection en temps réel de l'hypovigilance

La décision par modèle neuronal sur l'état de vigilance (veille ou sommeil) d'un sujet se base sur l'analyse des données spectrales d'une dérivation EEG enregistrée chez ce volontaire. L'analyse de cette dérivation se fait sur des époques de 4s. Pour chaque époque 23 puissances spectrales relatives sont calculées. Le vecteur X= $(x_1, x_2, ..... x_{23})$, est le vecteur d'entrée du réseau LVQ en phase d'apprentissage et de décision. L'architecture du réseau LVQ utilisé comporte 23 neurones sur la couche d'entrée et une matrice de (5x5) 25 neurones sur la couche de sortie. Pour chaque neurone un poids lui est attribué après la phase d'apprentissage qui est réalisée préalablement. Dans le cadre de cet article, nous présentons l'implantation du LVQ en phase de décision. Le principe consiste en deux étapes :

- un calcul des distances $d_j$ (j allant de 1 à 25) entre le vecteur à classer et les poids des différents neurones de la carte.

$$d_j = \sum_{i=1}^{i=23} (x_i - w_{ji})^2$$

X = $(x_1, x_2, ..... x_{23})$ est le vecteur d'entrée du réseau et W est une matrice de 23 x 25 éléments avec $\mathbf{W_{ij}}$ = ( $w_{1j}$, $w_{2j}$, .... $w_{23j}$), j allant de 1 à 25.

- Une détermination, par comparaison, de la distance $d_j$ minimale afin de conclure sur le neurone gagnant (figure 1).

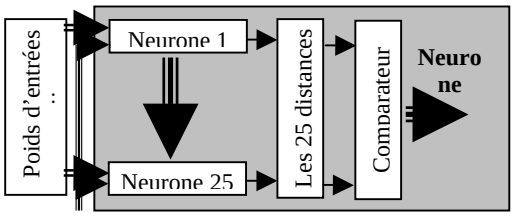


FIG. 1:architecture du LVQ

## 3. Méthodologie de conception pour circuit reconfigurable

L'extension de la méthodologie AAA aux circuits est fondée sur une approche globale formalisant l'algorithme et l'architecture. Le graphe flot de données modélisant l'algorithme au niveau comportemental est transformé progressivement jusqu'à ce qu'il corresponde au graphe matériel de l'architecture, lequel comprend le graphe du chemin de contrôle et le graphe du chemin des données. Le résultat de ces transformations correspond à l'implantation matérielle optimisée. L'adéquation revient à choisir parmi toutes les implantations possibles d'un algorithme sur une architecture, l'implantation dont les performances, déduites des caractéristiques des composants de l'architecture (la version actuelle de SynDEx-IC supporte une architecture mono-composant), respectant les contraintes temps réel, tout en minimisant la consommation des ressources, ce qui se traduit par un problème d'optimisation. La figure 2 décrit les étapes du processus d'implantation AAA appliquées aux circuits.

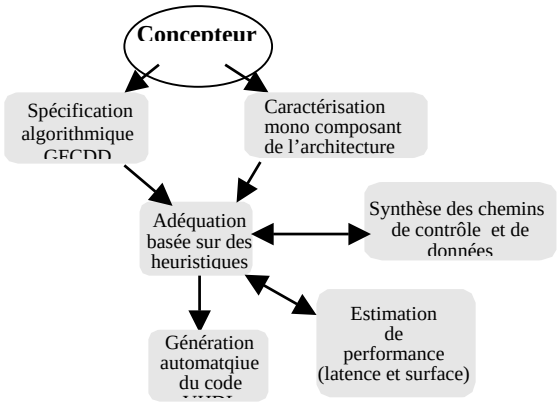


FIG. 2 : processus d'implantation AAA

### 3.1 Spécification algorithmique

La spécification de l'algorithme est sous la forme d'un graphe de dépendances de données, comprend les parties régulières (motifs répétitifs périodiques) et non régulières. Cette spécification doit être indépendante de toutes contraintes liées à l'implantation matérielle et nécessite la mise en oeuvre d'un processus de décomposition de l'algorithme en opérations implantables (addition, soustraction et multiplication) [5]. Pour réduire la taille de la spécification et mettre en évidence ces parties régulières, nous utilisons un processus de factorisation, qui fait apparaître des sommets spécifiques (sommets frontières de factorisation) : *F* (partition d'un tableau en autant d'éléments que de répétitions du motif), *J* (composition d'un tableau à partir des résultats de chaque répétition du motif), *D* (diffusion d'une donnée à toutes les répétitions du motif) et *I* (dépendance de donnée inter itération du motif). Ce processus de factorisation spécifie les différentes manières de factoriser les données et les opérations en traversant une frontière de factorisation. Une opération, située du côté d'une frontière qui présente

un groupe d'opérations identiques opérant sur un groupe factorisé de données différentes, sera réalisée au moyen d'un seul opérateur ou de plusieurs opérateurs en parallèle utilisés itérativement autant de fois qu'il existe d'opérations dans le groupe factorisé.

Pour le modèle LVQ de notre application, nous avons à calculer pour chaque neurone j la distance dj donnée par la formule :

$$d_j = \sum_{i=1}^{i=23} (x_i - w_{ji})^2$$

j varie de 1 à 25 le nombre de neurone sur la couche de sortie et $X_i$ le vecteur à l'entrée à 23 composantes et W est une matrice de 23 x 25 éléments. Notre graphe comporte deux frontières de factorisation *F11* et *F22*, la première (marquée par son degré de répétition) calcule la distance élémentaire entre les *23* entrées et un neurone de la couche de sortie. La deuxième génère à travers le sommet *J* les *25* distances pour tous les neurones de la couche de sortie (figure 3). J est un tableau qui permet de présenter les 25 distances calculées à l'entrée du calcul de la distance minimale, la spécification algorithmique du calcul de la distance minimal, contient un sommet conditionnement. D est le vecteur X à 23 composantes qui sera présenté après chaque calcul d'un $d_j$. La frontière *F22* comprend une dépendance I qui consiste à cumuler les résultats des différences au carré.

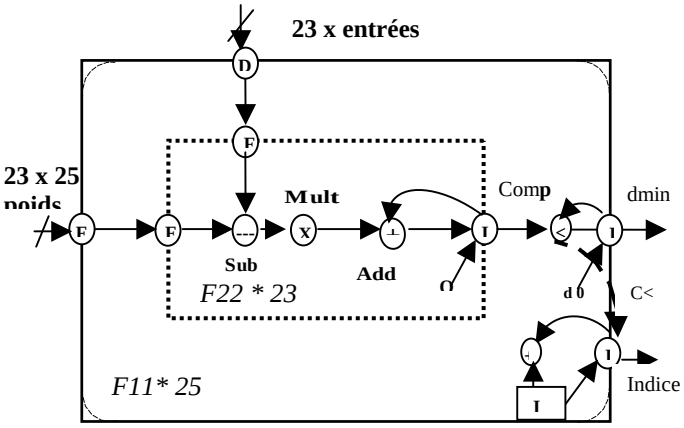


FIG. 3 : graphe factorisé de dépendances des données d'un réseau de neurones LVQ

## 3.2 Adéquation Algorithme Architecture

L'implantation séquentielle répétitive directe de notre graphe factorisé des dépendances de données GFDD sous sa forme factorisée entraîne une utilisation minimale de la surface en dépit d'un temps de calcul élevé. Dès lors, si la latence de cette implantation directe ne respecte pas les contraintes temps réel de l'application, nous procédons à des défactorisations des frontières de factorisation du graphe initial (déroulage de boucles). Défactoriser une frontière consiste à remplacer cette frontière par plusieurs frontières représentant le même motif de répétition, mais dont la somme des répétitions est égale à la répétition de la frontière initiale. Ces nouvelles frontières s'exécutent en parallèle d'où une réduction du temps de calcul, mais en contrepartie la surface d'implantation est augmentée (figure 4). Parmi toutes les transformations possibles par défactorisation, on éliminera celles qui ne respectent pas les contraintes temps réel.

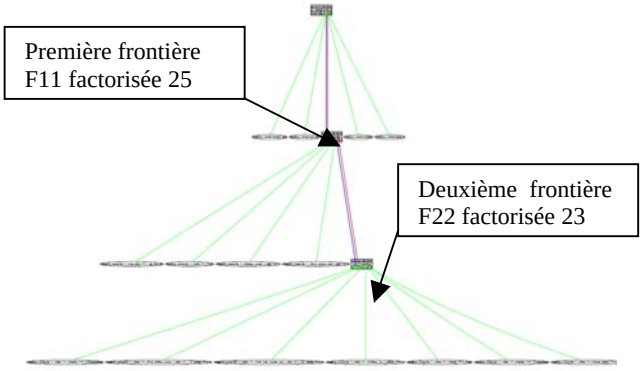


FIG. 4 : graphe de voisinage (relation entre les différentes frontières) : graphe flot de donnée

# 4. Résultats de l'implantation

Dans la figure 3 nous présentons la spécification algorithmique sous forme de GFCDD des principaux éléments constituants un réseau de neurone à 23 entrées et 25 sorties avec les calculs des distances euclidiennes et de la distance minimale. Nous avons testé l'implantation du réseau de neurones LVQ sur le circuit programmable de la famille XILINX Virtex II Pro xc2vp7 sous différentes contraintes de temps. Le tableau 1 présente les résultats obtenus par l'heuristique d'optimisation (l'heuristique détermine les facteurs de défactorisation optimaux de chacune des frontières composant le chemin critique) sous certaines contraintes. Par exemple pour une contrainte de latence supérieure à *12.32 µs* l'heuristique opte pour une implantation factorisée de la frontière *F11* (figure 4) par un facteur de *25* (Calcul séquentiel, 1 neurone s'exécutant 25 fois dont la répétition est égale au facteur de répétition de frontière *F11*). Ces résultats présentés en termes, de surface pour les ressources matérielles (nombre de CLBs : Control Logic Blocs) et de latence (en µs) montre que les solutions les plus défactorisées permettent de réduire la latence mais augmentent en contrepartie la consommation en ressources. De ce fait, selon les contraintes imposées, les latences des solutions obtenues évoluent ainsi par palier ce qui est rapporté par le tableau 1 et illustré par la figure 5.

TAB. 1 : Résultats d'implantation sur VIRTEX II Pro

| Contrainte application µs | Latence réelle µs | Surface CLB | Neurone implanté |
|---|---|---|---|
| 10.5 | 10.32 | 3930 | Calcul séquentiel : 1 neurone factorisé 25 fois |
| 3 | 2.082 | 6505 | Calcul séquentiel : 5 neurones factorisé 5 fois |
| 0.426 | 0.426 | 7942 | 25 neurones factorisé 1 fois |

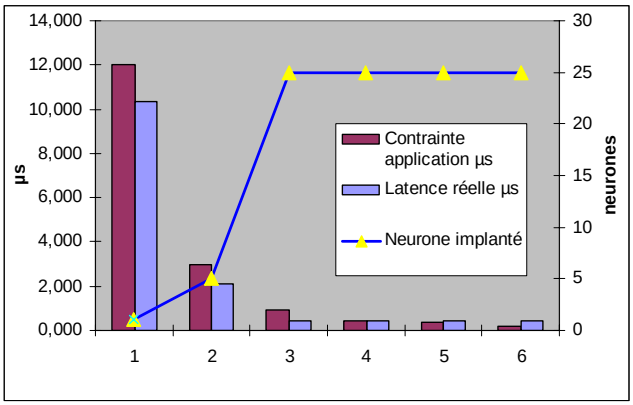


FIG. 5 : variation de la contrainte temporelle en fonction du nombre de neurones factorisés

Dans un deuxième temps, nous avons fixé la latence à 10,5 µs et nous avons effectué les synthèses en variant le nombre de neurones sur la couche de sortie (3x3, 6x6, 5x5 et 6x6 neurones). Le tableau 2 rapporte les résultats obtenus. En termes de ressources et de temps de synthèse.

Tab. 2 : Résultats d'implantation sur VIRTEX II Pro en variant le nombre de neurones de la couche de sortie

| **Latence (µs)** | **Dimension de la couche de sortie** | **Ressources (CLB)** |
|---|---|---|
| 10,5 | 9 (3x3) | 2303 |
| 10,5 | 16(4x4) | 3600 |
| 10,5 | 25(5x5) | 3930 |
| 10,5 | 36 (6x6) | 4905 |

# 5. Conclusions

Nous avons adopté la méthodologie AAA et le logiciel SynDEx-IC, pour l'optimisation de l'implantation sur circuit FPGA d'un réseau de neurones à partir d'une spécification algorithmique sous forme d'un graphe factorisé et conditionné de dépendances de données. Nous avons exploité au mieux l'aspect motif répétitif dans l'algorithme neuronal pour une meilleure factorisation. Cette approche nous a permis d'effectuer la synthèse de l'implantation d'un réseau de neurones LVQ en imposant des contraintes de latence et de ressources et en variant le nombre de neurones sur la couche de sortie. Dans le cas où la cible est un FPGA Xilinx Vertex IIPro avec 4928CLB de ressources, la meilleure adéquation de l'implantation d'un LVQ à 25 neurones sur la couche de sortie, est une architecture totalement factorisée occupant 3930 CLB (80% de ressources disponibles) avec une latence de 10.32 µs à une fréquence de 80 MHz. Sur la même cible et dans les mêmes conditions de latence et de fréquence, nous avons pu implémenter un LVQ à 36 neurones sur la couche de sortie en occupant 4905 CLB (99.6%). Dans le cas du support XCVP20 de la même famille, nous avons atteint comme latence 0.213 µs en implémentant d'une façon totalement défactorisée le LVQ à 25 neurones sur la couche de sortie en occupant 7942 CLB (82%) à une vitesse de 80 MHz.

Nous soulignons que l'application développée sous l'environnement SynDEx-IC nous a permis le choix de la meilleure architecture d'implémentation du LVQ destinée à un système embarqué sous contraintes de temps et de ressources tenant compte de la cible et de générer automatiquement le code VHDL pour des configurations différentes de l'algorithme neuronal LVQ, ce qui permet d'explorer automatiquement (prototypage rapide) différentes implantations.

Nous travaillons actuellement sur la validation de la phase de décision par comparaison avec les résultats obtenus en soft sur notre corpus de travail. De même nous adoptons la même démarche pour l'implantation de l'algorithme LVQ dans ses phases d'apprentissage et de décision. Nous étudions la possibilité d'intégrer d'autres règles dans l'environnement SynDEx-IC qui tiennent compte des spécificités de l'implantation des algorithmes connexionnistes par exemple : des combinaisons d'opérateurs de base, exploiter dans la même architecture des arithmétiques différentes parallèle et sérielle. D'autre part, nous souhaitons tenir compte de la consommation dans notre démarche d'optimisation des performances de l'implantation du réseau neuronal.

# Références

**[1]** Vuckovic A, Radivojevic V, Chen AC, Popovic D. Automatic recognition of alertness and drowsiness from EEG by an artificial neural network, Med Eng & Phys. 2002 Jun; 24(5), p.349-60.

**[2]** K. Ben Khalifa F. Alexandre, N. Kerkeni and M.H. Bedoui. Artificial neural networks to extract knowledge from eeg. In The IASTED International Conference on Biomedical Engineering - BioMED2005, Innsbruck, Austria, Feb 2005

**[3]** Ben Khalifa K, Traitement Matériel et Logiciel des signaux physiologiques : Application à l'étude de la vigilance. Université de Monastir, Faculté des Sciences de Monastir, Thèse de doctorat en Physique. Monastir: Avril 2006.

**[4]** T. Grandpierre, Y. Sorel. From Algorithm and Architecture Specifiations to Automatic Generation of Distributed Real-Time Executives: a Seamless Flow Graphs Transformations. IN Procs. IEEE Formal Methods and Models for Codesign Conference (MEMCODE'2003), pages 123-133, Mont Saint-Michel, France, June 24-26, 2003. IEEE Computer Society.

**[5]** L.Kaouane, M.Akil, Y.Sorel, T.grandpierre. A methodology to implement real-time applications onto reconfigurable circuits, Special issue on Engineering of Configurable Systems of the Journal of Supercomputing, Kluwer Academic Publisher, Vol.30, No.3, Dec. 2004 .