Implantation basse-consommation de cartes auto-organisatrices pour la détection de l'hypovigilance

Low-power implementation of self-organizing maps for alertness detection

K. Ben Khalifa[1] B. Girau[2] F. Alexandre[2] M.H. Bedoui[1] R. Raytchev[1]

[1] Laboratoire de Biophysique, Faculté de Médecine de Monastir, Tunisie
[2] LORIA INRIA-Lorraine, Nancy, France

khaled.benkhalifa@issatso.rnu.tn

Résumé

Plusieurs études ont déjà été menées pour tenter de discriminer, à l'aide de réseaux de neurones artificiels, les différents états de vigilance d'un sujet humain à partir de signaux électroencéphalographiques (EEG). Nous avons mené une étude plus large et exhaustive sur les modèles neuronaux utilisés, sur leurs caractéristiques et sur leurs performances. Nous y avons associé des médecins, notamment lors de la mise au point fine de nos modèles. Enfin, et surtout, notre étude a été orientée de manière à pouvoir obtenir un système léger, utilisable sans entrave par un sujet humain, grâce à une limitation des besoins de calcul et de mémoire. Cet article résume ces travaux et présente leur implantation basse-puissance sur circuit FPGA, dans la perspective d'un système électronique portable. Cette implantation est basée sur une architecture totalement parallèle à base d'opérateurs sériels. Les performances en surface d'implantation, vitesse, et surtout consommation satisfont largement les attentes de ce travail.

Mots Clef

hypovigilance, EEG, réseaux de neurones, SOM, FPGA, basse-puissance

Abstract

Several studies have been carried out, using artificial neural networks, to discriminate vigilance states in humans from electroencephalographic (EEG) signals. Our study is wider and more exhaustive, taking into account various neuronal models, and precisely studying their features and their performances. Physicians have been associated to the project, especially when tuning our models. Above all, our work has been oriented in such a way to get a light, easy to wear system: computation and memory requirements have been limited. In this paper, these works are summarized, and their low-power implementation on a FPGA is described, looking forward to a portable electronic system. It is based on a full-parallel architecture that uses serial operators. Performances in terms of area, speed and especially power consumption are highly satisfactory.

Keywords

alertness, neural networks, Kohonen, FPGA, low-power

1 Introduction

Nous présentons dans cet article les travaux de mise en application sur circuit numérique programmable embarqué des résultats d'une démarche menée antérieurement pour dégager les paramètres électroencéphalographiques susceptibles de caractériser et de classifier les différents états de vigilance chez des sujets en situation réelle et en tenant compte des artefacts.

Nous utilisons, entre autres, des algorithmes de classification automatique à apprentissage non supervisé qui se fondent sur les cartes auto-organisatrices [1] de Kohonen. Les résultats de ces travaux sont présentés dans [1]. Le but est d'aboutir à un système portable de détection de l'hypovigilance à partir d'un nombre minimal de dérivations d'EEG en vue d'une exploitation en ambulatoire et dans les conditions réelles de tous les jours. En effet, la conception d'un tel système pose un problème multidisciplinaire en mettant en œuvre diverses facettes du savoir humain : la physiologie, la psychologie, ainsi que l'informatique et l'électronique. De plus, les contraintes d'utilisation envisagées impliquent la nécessité d'aboutir à une solution très basse-puissance, ce qui constitue le critère essentiel pour les différents choix technologiques opérés. Il faut noter que les travaux qui sont menés jusqu'à maintenant [6, 12], traitent esssentiellement la détection elle-même sans présenter les travaux d'implantation sur des systemes programmables.

L'approche developpée dans ce papier concerne l'implantation des réseaux de neurones artificiels (RNA), notamment dans notre application les cartes auto-organisatrices de Kohonen en phase de décision, sur un circuit programmable de type FPGA en vue d'une éventuelle exploitation

1. Self-organizing map, SOM

dans un système embarqué. Il faut noter que les paramètres du réseau en question (les poids des liaisons entrées/sorties) sont précalculés à l'exterieur du circuit programmable, sur un PC.

Différents types d'implantations de SOM sur circuits intégrés existent déjà. On peut les répartir principalement en deux catégories :

- Des implantations analogiques des SOM sur circuits intégrés dédiés ont été réalisées (par exemple [8]). Ces implantations sont particulièrement bien adaptées aux fonctionnalités simples des neurones et permettent d'atteindre par la suite un rapport entre vitesse de calcul, densité et consommation hors de portée des implantations numériques. En revanche, ces technologies présentent des limites techniques telles que leur manque de précision, de robustesse ainsi que leur sensibilité à la technologie utilisée.
- Des implantations de cartes auto-organisatrices de Kohonen sur circuits numeriques ASIC (Application Specific Integrated Circuit) ont également été réalisées (neuroproceseurs). Actuellemtn, ces composants constituent la catégorie de circuits VLSI la plus utilisée pour l'intégration d'algorithmes neuromimétiques. On peut citer par exemple l'implantation sur la plateforme expérimentale MANTRA I de [11] formée par une grille de 40x40 éléments de calcul (Processing Element PE). Cependant la configuration de ce système est trop complexe pour des utilisateurs non connaisseurs et il est de plus impossible d'intégrer cette plateforme dans un système embarquable.

Les défauts cités pour ces deux types d'implantations peuvent être contournés par l'utilisation de circuits reprogrammables, comme par exemple les FPGA (Field Programmable Gate Arrays). Les avantages de ces circuits sont directement liés à leur architecture programmable. En effet, ces circuits sont constitués d'une matrice de cellules préconçues (appelées CLB : Configurable Logic Blocks) dont les interconnexions et les cellules de base sont programmables. Cette disponibilité du matériel dans la puce permet au concepteur d'imaginer des architectures parallèles d'ANN. Dans [3] l'auteur a réussi à implanter une SOM en phase de décision à 8 entrées (codées sur 8 bits) et 4096 neurones sur la couche de sortie (grâce à des calculs séquentialisés). L'architecture proposée nécessite beaucoup de lignes d'interconnexions parallèles entre les différents co-processeurs, ce qui rend l'exploitation de ce réseau non modulaire pour d'autres applications. Speckmann dans [10] a implanté sur FPGA des unités de calcul (Processing Unit: PU) qui permettent d'extraire la distance euclidienne d'un vecteur d'entrée (codé sur 16 bits) pour seulement un neurone. L'architecture est entièrement parallèle et ne permet pas de calculer la distance minimale (en effet elle se fait à l'extérieur sur un PC qui s'interface avec le FPGA via des mémoires FIFO). Dans [9], les auteurs ont implantés une architecture dynamique reconfigurable de SOM (9 entrées codés sur 8 bits et 250x250 neurones sur la couche de sortie) sur une carte RAPTOR2000 architecturées autour de 4 FPGA Xilinx de la série Virtex-E et un CPLD de la famille Xilinx aussi. Comme dans [3] l'architecture proposée est entierement paralléle donc elle va nécessiter beaucoup de lignes d'interconnexions parallèles entre les différents co-processeurs (4 FPGA).

Nous présentons dans cet article une implantation sérielle d'un modèle de réseaux de neurones de type Kohonen sur un circuit FPGA de la famille Virtex de Xilinx. Ce travail est constitué de trois parties principales. La première consiste à traiter les signaux électroencéphalographiques par application de réseaux de neurones à apprentissage non supervisé et supervisé. La deuxième concerne l'étude de la précision en virgule fixe du calcul des opérations arithmétiques (soustraction et carré) qui donne les résultats les plus proches de ceux trouvés dans [1] en virgule flottante. Enfin, les résultats obtenus aboutissent à l'implantation des cartes de Kohonen en phase de décision sur le FPGA en exploitant les poids d'apprentissage extraits dans la première partie.

L'article débute par une description du traitement spectral effectué sur le signal EEG (sous-section 2.1). La présentation des méthodes connexionnistes qui sont exploitées dans notre application (SOM et LVQ de Kohonen) ainsi que des résultats trouvés suite à l'application de ces derniers est detaillée dans la sous-section 2.2. La section 3 décrit les choix technologiques utilisés pour l'implantation neuronale du Kohonen : dans la sous-section 3.1 nous allons décrire l'environnement de l'implantation, le choix de l'arithmétique série sera detaillé dans la sous-section 3.2 et enfin l'architecture des SOM de Kohonen implantée sera detaillée dans la sous section 3.4. Les résultats obtenus sont presentés dans la section 4.

2 Traitement et interprétation du signal EEG

2.1 Prétraitement du signal EEG

Comme premier traitement spectral, une transformée de Fourier Rapide à Court Terme (TFRCT) est appliquée à la dérivation EEG pariéto-occipitale droite (P4-O2) sur des portions de 4 secondes, avec une fenêtre de pondération de type Hamming et une résolution de 512 points.

Par la suite un découpage en bandes de 1 Hz du spectre du signal EEG est effectué. En effet 23 bandes de 1 Hz (allant de 1 à 24 Hz, normalisées par rapport à la puissance spectrale totale) sont extraites :

$PPS_i = \frac{PS(i \text{ à } (i+1)Hz)}{PST}$*100, i allant de 1 à 23 Hz.

Où PPSi représente le Pourcentage de la Puissance Spectrale de la bande i Hz correspondante, PST la Puissance Spectrale Totale, et PS la Puissance Spectrale.

Après ce codage initial par bandes du signal EEG, un traitement connexionniste par les cartes auto-organisatrices de Kohonen est appliqué. Les entrées du réseau de neurones en question sont les Pourcentages de la Puissance Spectrale par bandes (PPS_i) du même signal.

2.2 Méthodes connexionnistes

Le connexionnisme est l'étude des réseaux de neurones artificiels. Il peut en particulier s'intéresser aux propriétés statistiques des traitements effectués par des réseaux relativement simples et réguliers, ou encore, en se fondant sur la connaissance actuelle des mécanismes du fonctionnement du cerveau, modéliser, par la mise au point de réseaux d'inspiration biologique, différents aspects de la cognition. Dans tous les cas, les réseaux étudiés ont des propriétés de traitement de l'information communes : la mise en œuvre d'un grand nombre de cellules de base, "les neurones", travaillant en parallèle et massivement interconnectés, leur donne des capacités d'apprentissage et de prise de décision (généralisation à partir des données d'apprentissage).
Dans ce travail deux algorithmes d'extraction automatique de catégories à l'aide de modèles auto-organisés (apprentissage non supervisé ou apprentissage supervisé) ont été exploités. Ils sont présentés ci-dessous.

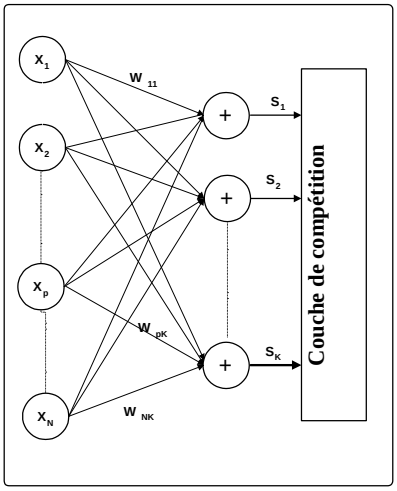

FIG. 1 – *SOM de Kohonen*

les cartes auto-organisatrices de Kohonen. Le but est d'utiliser la capacité des cartes auto-organisatrices de Kohonen à séparer, d'une façon non supervisée, des états par ailleurs déjà qualifiés par l'expert, pour analyser aussi bien la répartition de ces états sur l'espace de sortie que les associations qui peuvent se dégager entre ces états. Le modèle SOM modélise le mécanisme de l'auto-organisation spatiale des perceptions opérée par le cortex sous forme d'un processus de classification topographique. Selon ce processus, les données d'entrées, représentables dans le cas général sous forme de vecteurs à N dimensions, sont ramenées à des classes qui s'auto-organisent selon une structure bidimensionnelle de nœuds sur laquelle les relations de voisinage sont prédéfinies. Le processus de classification topographique du modèle SOM combine donc une étape de classification avec une étape de projection des données. Pour la réalisation de ce modèle connexionniste, deux couches de neurones sont utilisées : la première représente les entrées, la seconde les sorties (les classes). Les deux couches sont entièrement connectées. L'algorithme d'apprentissage du modèle SOM est présenté de manière détaillée dans [7]. Il est de type compétitif, non supervisé. Il comprend principalement deux étapes : (1) Sélection d'un nœud gagnant - (2) Mise à jour du profil du nœud gagnant et de ceux des nœuds appartenant à son voisinage.

Sélection du nœud gagnant :
Soit $x(t) = \{x_1(t),x_2(t),.....,x_N(t)\}$ le vecteur d'entrée (i.e. la donnée) sélectionné au temps t, et soit

$$W_k(t) = \{W_{k1}(t),W_{k2}(t),.....,W_{kN}(t)\}$$

le vecteur de poids associé au nœud k au temps t. Une mesure de distance est choisie (par exemple, la distance euclidienne) et la distance la plus faible $\|x(t) - W_k(t)\|$ permet de définir le nœud gagnant c, soit : $\|x(t) - W_c(t)\| = \min_k \|x(t) - W_k(t)\|$

Apprentissage non supervisé et sélection du voisinage :
Après la sélection du nœud gagnant c, le vecteur de poids associé à ce nœud ainsi que les vecteurs de poids associés aux nœuds se trouvant dans un voisinage donné du nœud c (c'est-à-dire les sorties situées dans un périmètre défini autour du nœud c) sont ajustés de manière à ce que leur profil se rapproche de celui de la donnée d'entrée. Cet ajustement de poids qui caractérise l'apprentissage non supervisé du modèle peut être décrit par l'équation : $W_{ki}(t+1) = W_{ki}(t) + a(t) * h(k,k_*) * [X_i(t) - W_{ki}(t)]$ pour $1 \leq i \leq$ N où $a(t)$ est un terme de gain ($0 \leq a(t) \leq 1$) décroissant en fonction du temps et convergeant vers 0, et $h(k;k_*)$ est la fonction de voisinage (ou d'interaction), qui est une fonction de la distance $d(k;k_*)$ entre les unités k et k_* sur la carte. Cette fonction vaut 1 lorsque $d(k;k_*) = 0$ et décroit quand la distance augmente.
A l'issue de l'algorithme d'apprentissage, les données en entrée peuvent être reprojetées sur la carte obtenue. Le nœud d'affectation d'une donnée représente alors celui dont le vecteur de poids est le plus proche du vecteur descriptif de la donnée. Il peut donc être sélectionné en réutilisant l'étape (1) ci-dessus.
Cette première approche exploite le pouvoir de séparation non linéaire des SOM de Kohonen. Il faut noter que peu d'informations sur les variations du signal EEG pendant les périodes d'hypovigilance sont disponibles. Ceci explique le choix de ce type d'algorithme à apprentissage non supervisé dans notre application. Les cartes auto-organisatrices de Kohonen ont permis de donner une visualisation topographique du spectre du signal EEG enregistré au moment de la phase de transition éveil-sommeil. Les résultats dégagés ont permis de trouver des associations de voisinage possibles entre différents niveaux de vigilance. Nous avons réussi à avoir des résultats comparables à ceux trouvés dans [6] en réduisant le nombre de dérivations d'EEG de 20 à une seule.

Le Learning Vector Quantization : LVQ. Les cartes non supervisées de Kohonen constituent déjà un outil de prétraitement efficace pour la séparation et la redistribution des

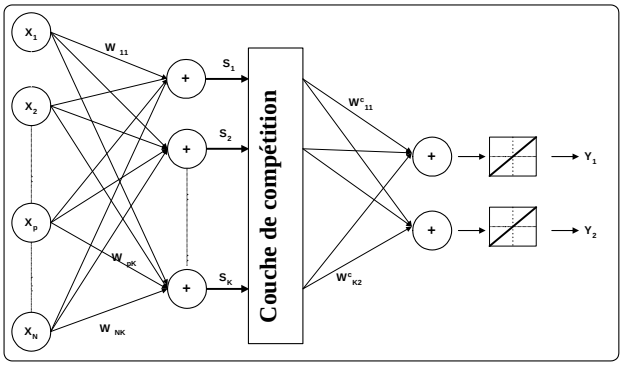

FIG. 2 – *LVQ de Kohonen*

vecteurs d'entrées en différentes classes. En effet, les SOM nous permettent d'avoir une idée sur la distribution statistique des vecteurs d'entrées sur la couche de sortie. Après apprentissage chaque neurone de cette couche peut être activé par des vecteurs d'entrée qui correspondent à différentes classes, ce qui pose un problème dans le processus de décision et d'étiquetage des neurones.

Pour remédier à cette limite, on a appliqué une deuxième phase d'apprentissage, cette fois-ci supervisée. Cette phase va permettre un réajustement des distributions de probabilités sur la carte de sortie et prendre par la suite des décisions sur l'étiquette attribuée à chaque neurone.

Notre choix s'est porté sur l'algorithme à apprentissage supervisé proposé par Kohonen : Le Learning Vector Quantization (LVQ). Deux variantes significatives de cette méthode ont été présentées : LVQ2.1 [7] et LVQ3 [7]. L'architecture du LVQ est similaire à celle de la carte de Kohonen, sans connexions latérales pour les cellules de la deuxième couche. Cet algorithme constitue, avec ses différentes variantes, une amélioration de la séparation en classes à partir de la solution proposée par l'apprentissage non supervisé.

La méthode consiste à rapprocher le prototype le plus activé de l'entrée s'il est de la bonne classe (apprentissage supervisé), et à le repousser dans le cas contraire. Les autres prototypes (c'est-à-dire les perdants) restent inchangés. Les prototypes deviennent ainsi les représentants des classes.

Cette deuxième approche fondée sur l'exploitation des LVQ a permis de faire d'une part une classification des niveaux de veille et de sommeil et d'autre part un filtrage des séquences artefactées. Les résultats dégagés par ce réseau de neurones sont similaires (et même meilleurs) à ceux présentés par Vuckovic dans [12]. En s'alignant sur leur protocole, nous atteignons un taux de reconnaissance des deux états de l'ordre de 100% sur des corpus de test. Le LVQ a permis de gommer les différences inter-individuelles entre les sujets. En effet, le taux de reconnaissance total des niveaux de vigilance atteint sur des corpus de test 76,73% [1].

2.3 Problématique de l'implantation

Les résultats trouvés jusqu'à maintenant ont été obtenus à partir des simulations logicielles effectuées sur un PC. Le but final de ce travail est de réaliser un système embarqué pour la détection de l'hypovigilance.

Pour cela, nous avons effectué une implantation des SOM de Kohonen en phase de décision sur un FPGA (en exploitant les paramètres d'apprentissage obtenus par simulation). Pour une telle réalisation il faut tenir compte de plusieurs paramètres et caractéristiques utiles comme la puissance consommée en fonction de la fréquence de l'horloge externe, le nombre d'entrées/sorties, la surface totale d'intégration, les besoins en signaux de contrôle (CLK, reset des opérateurs), l'architecture choisie et enfin le parallélisme neuronal (les différents neurones peuvent travailler de façon concurrente). Il est intéressant de signaler que certains de ces paramètres sont difficilement estimables avec précision avant la synthèse.

La vitesse d'exécution est parfois un autre critère essentiel. Dans notre cas, la prise de décision (détection d'une situation d'hypovigilance) se fait en temps réel sans difficulté grâce à une implantation totalement parallèle ne nécessitant ni utilisation séquentielle des ressources de calcul, ni reconfiguration dynamique du FPGA. La vitesse d'exécution n'est donc pas une véritable contrainte pour les choix technologiques effectués. En revanche, l'hypothèse d'un circuit embarqué de manière ambulatoire implique le besoin d'aboutir à une implantation très basse-puissance. Parmi les paramètres cités ci-dessus, le nombre d'entrées/sorties, mais surtout l'obtention d'une solution qui exploite entièrement le parallélisme neuronal ont une influence directe sur la consommation de l'implantation obtenue. La taille du réseau de neurones et celle du FPGA utilisé font de cette obtention un véritable défi.

3 Matériels et méthodes : les choix technologiques

3.1 Environnement d'implementation

La carte prototype RC1000-PP. Pour tester nos implantations, nous avons utilisé la carte PCI RC1000-PP commercialisée par Celoxica. Cette carte est architecturée autour d'un circuit XCV1000E−4BG560 de la famille Virtex de Xilinx. Cet environnement de développement propose essentiellement une horloge programmable (de 400 KHz à 100 MHz) et une mémoire 32 bits, organisée en quatre banques de 2 MB accessibles indépendamment depuis l'ordinateur hôte ou le FPGA. Cette mémoire va être utilisée surtout pour l'échange d'informations entre l'environnement externe et le FPGA (stockage des données d'entrée et des résultats).

Les FPGA : Virtex . Les FPGA, tels que les FPGA Xilinx ([13]), sont basés sur une matrice de cellules appelées CLB (*configurable logic blocks*). Chaque CLB peut implanter des fonctions logiques simples (4 ou 5 entrées boo-

léennes) avec quelques éléments de stockage (bascules par exemple) et des multiplexeurs. En fonction des capacités des CLB, certains opérateurs peuvent être implantés efficacement ou pas sur des FPGA. Les FPGA de la série Xilinx Virtex sont ainsi actuellement parmi les mieux adaptés à l'implantation d'opérateurs sériels, en raison d'un nombre d'éléments de stockage relativement élevé par rapport à d'autres FPGA du marché. Plus spécifiquement, un circuit Virtex comporte des CLB composés de quatre cellules logiques (LC ou Logic Cell) réparties en deux tranches (slices) identiques.
Chaque LC contient essentiellement :

1. Un générateur de fonctions à quatre entrées, réalisé à l'aide d'une table associative (LUT ou look-up table). Chaque table permet également la conception d'une mémoire synchrone 16×1 bit. Une fois appariées, les deux LUT d'une tranche offrent une mémoire synchrone 16×2 bits, 32×1 bits ou 16×1 bits à double accès (dual-port synchronous RAM). Une LUT fonctionne également comme un registre à décalage de seize bits.
2. Un élément de mémorisation, possédant des signaux d'initialisation (set et reset) synchrones ou asynchrones.
3. De la logique destinée à la conception de circuits arithmétiques performants.

Notre carte est équipée d'un circuit Virtex XCV1000E contenant 6144 CLB ($64 * 96$), soit l'équivalent de plus de 1,5 millions de portes.
Les CLB peuvent être connectés grâce à une structure de routage configurable. Dans un Virtex, les CLB peuvent être efficacement connectés aux CLB voisins et aux CLB de la même ligne ou colonne de la matrice de cellules. La structure de communication configurable peut également connecter des CLB à des blocs IOB qui gèrent les tampons d'entrée/sortie du circuit.
Une implantation à base de FPGA peut donc utiliser la flexibilité des configurations des CLB et de leurs communications pour s'adapter simplement à l'application visée, tandis que les implantations VLSI habituelles doivent être profondément modifiées puis refondues lorsque certaines caractéristiques de l'application changent. La conception sur FPGA demande la description de quelques blocs de base, d'élément de contrôle et d'un schéma de communication. Les outils de synthèse permettent ensuite de placer automatiquement et de façon fiable l'implantation décrite sur le circuit utilisé.
Les principaux atouts de cette approche intermédiaire entre matériel et logiciel par rapport aux circuits ASIC sont les suivants.

- Leur coût est très inférieur.
- Le temps de développement est également très inférieur.
- Certains supports sont réutilisables.
- Un prototypage direct est possible avec ces implantations matérielles, par reprogrammation du circuit.
- Il est possible, avec un même circuit, d'implanter des phases de calcul très diverses d'une même application embarquée.
- Une implantation sur matériel configurable peut bénéficier ultérieurement des progrès rapides des constructeurs de ces circuits.
- Une grande partie du code est réutilisable pour synthétiser des circuits ASIC si les performances obtenues sur FPGA justifient un tel investissement.
- Ce type d'implantation est rapidement accessible pour les gens ayant néanmoins une connaissance limitée des implantations matérielles.

3.2 Arithmétique sérielle sur FPGA

L'arithmétique sérielle permet de réaliser des architectures de calcul où les chiffres circulent en série, chiffre après chiffre. Deux types d'architectures sont possibles :

- Architecture sérielle poids faibles en tête [LSBF].
- Architecture sérielle poids fort en tête [MSBF]:
  L'intérêt de cette architecture est de pouvoir calculer toutes les fonctions (en arithmétique série poids faibles en tête, il n'est pas possible de calculer certaines opérations comme la division, la comparaison ainsi que les fonctions mathématiques élémentaires).

L'arithmétique série permet d'obtenir des opérateurs:

- Occupant des petites surfaces d'implantation.
- Nécessitant moins d'entrées/sorties que leurs équivalents en arithmétique parallèle.
- Nécessitant aussi un nombre minimal de connexions (internes et externes).
- Permettant de régler aisément la précision, sans augmentation excessive de surface d'implantation.
- Permettant de paralléliser de manière temporelle, à grain très fin (en introduisant la notion du pipeline au niveau des bits et des données) des suites de calculs intrinsèquement séquentielles

Nous présentons dans cet article une implantation d'un modèle de réseaux de neurones de type Kohonen sur un circuit FPGA de type VIRTEX, à l'aide d'une arithmétique série LSBF : ce choix est motivé par la nature des opérations arithmétiques utilisées pour ce type d'ANN comme la soustraction, le carré et l'addition. Le cas des comparaisons (naturellement MSBF) introduit un traitement spécifique, par bufferisation des résultats des autres calculs.
Le parallélisme introduit par le pipeline au niveau du bit et par le traitement de plusieurs entrées et sorties (23 neurones sur la couche entrées et 5 x 5 neurones sur la couche de sortie) simultanément permet d'obtenir des performances (en terme de surface d'intégration et puissance de consommation) très nettement meilleures que celles obtenues par des solutions basées entièrement sur une arithmétique parallèle.

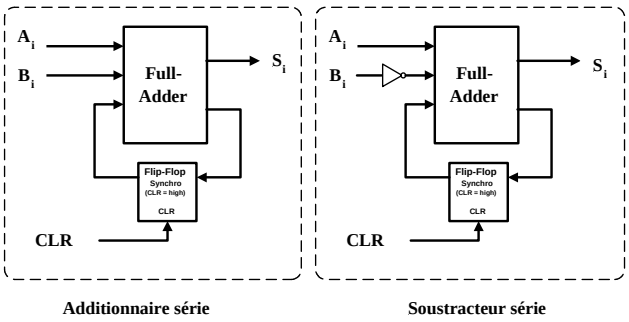

FIG. 3 – *Additionneur et soustracteur sériels LSBF*

L'addition et la soustraction sérielles. L'addition série, en complément à deux, s'effectue en LSBF à l'aide de l'architecture de la figure 3(a), avec un délai nul, la retenue étant mémorisée dans une bascule initialisée à '0'.
Pour transformer ce circuit en soustracteur, il suffit d'inverser l'entrée B_i et d'initialiser la bascule avec la valeur '1', figure 3(b).

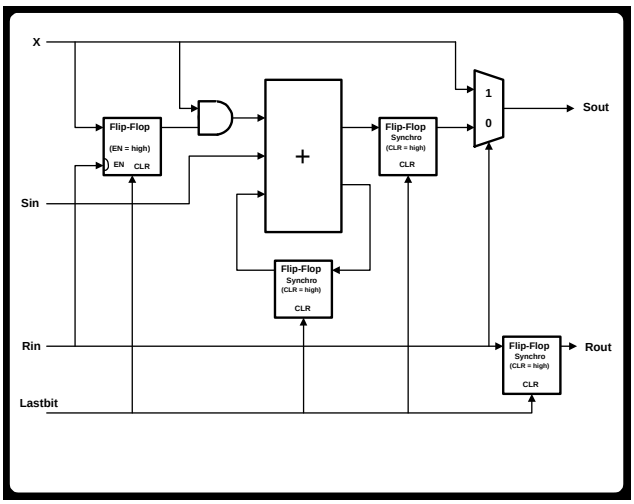

FIG. 4 – *Unité élémentaire de calcul du carré*

Le calcul du carré. L'opérateur utilisé pour le calcul du carré correspond à celui étudié et réalisé par Ienne dans [5]. L'architecture de l'opérateur est présentée sur les figures 4 et 5. L'idée consiste, dans le cas plus général d'une multiplication, à recoder les entrées sur $2n+1$ bits (n répresente le nombre de bits des entrées), ce qui permet de décaler la gestion du bit de complément à 2 au-delà des $2n$ du résultat attendu. Après la présentation des deux bits d'ordres i, le résultat du produit partiel est :

$$P_i = X_i.Y_i = P_{i-1} + x_i.Y_{i-1}.2^i + y_i.X_{i-1}.2^i + x_i.y_i.2^{2.i}$$

X_i et Y_i sont les valeurs X et Y réduites aux bits 0 à i : $X_i = X \bmod 2^{i+1}$. Notons que les valeurs initiales sont : $P_{-1} = X_{-1} = Y_{-1} = 0$.

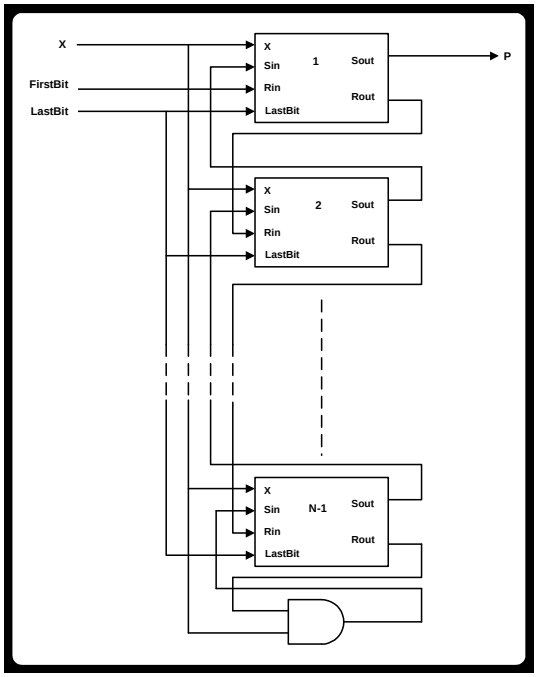

FIG. 5 – *Calcul du carré (à base d'unités élémentaires)*

Cette équation est valable jusqu'à l'apparition du bit de complément ($i < n-1$).
Pour $(i \geq n)$ l'équation ci-dessous est exploitée:

$$\overline{P_i} = \overline{P_{i-1}} + x_{n-1}.Y_{n-2}.2^i + y_{n-1}.X_{n-2}.2^i + x_i.y_i.2^{2.i} \quad (1)$$

$\overline{P_i} = P_i$ est toujours vrai pour $i < n$.
Pour $i \geq n$ l'erreur générée à la suite de l'utilisation de l'équation 1 est la suivante :

$$E_i \quad = \quad x_{n-1}.y_{n-1}.2^{2.i}.\sum_{j=n}^{i} 2^j$$

D'après cette dernière égalité on a $E_i \bmod 2^{i+1} = 0$. On peut extraire par la suite le résultat de la multiplication (ou du carré).

Le choix de cette implantation correspond à nos besoins de minimisation de surface sur le FPGA, grâce à l'absence de registre spécifique de stockage interne des résultats intermédiaires (cf opérateurs à addition et décalage).

3.3 La précision du calcul

La précision requise par notre application neuronale a été étudiée par simulation logicielle. Les précisions ont été étudiées en fonction de la nature des données : poids, entrées et carrés. Huit chiffres binaires sont suffisants pour réprésenter les poids et les données d'entrées (1 bit de complément, 4 bits pour la partie entière et 3 bits pour la partie

décimale). De plus 8 bits parmi les 16 issus de la mise au carré suffisent pour représenter les sorties des calculs de distance (6 bits pour la partie entière et 2 bits pour la partie décimale, pas de bit de signe, les valeurs étant positives). L'implantation peut être aisément étendue à des precisions supérieures en cas de besoin, n'entraînant qu'une augmentation linéaire de surface de certains opérateurs. D'autre part un changement de précision implique une modification des intervalles de temps du contrôle de l'architecture.

3.4 Architecture de la carte auto-organisatrice

L'implantation du réseau de neurones SOM (Kohonen ou LVQ, puisque les deux modèles sont identiques en phase de décision) sera décrite tout d'abord dans la sous-section 3.4 où on va détailler l'architecture d'un neurone de la couche de sortie et dans la section 3.4 où on va détailler l'architecture globale du réseau.

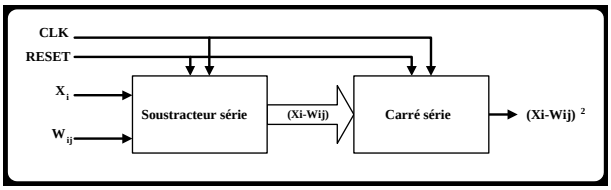

FIG. 6 – *Unité élémentaire pour le calcul série de la soustraction et du carré*

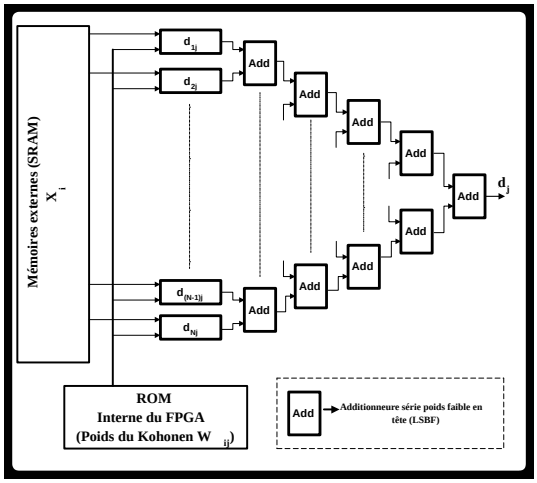

FIG. 7 – *Architecture d'un neurone*

Architecture d'un neurone. Pour chaque neurone j de la couche de sortie on doit calculer la combinaison $d_j = \sum_{i=0}^{N-1}(X_i - W_{ij})^2$ (N représente le nombre des entrées de la SOM, dans notre cas $N = 23$). L'architecture de l'opérateur $(X_i - W_{ij})^2$ est présentée sur la figure 6 : association entre un soustracteur sériel et un multiplieur carré sériel. L'architecture globale de la fonction $\sum_{i=0}^{N-1}(X_i - W_{ij})^2$ est présentée sur la figure 7. La taille relativement réduite des opérateurs série permet d'effectuer la totalité des opérations en parallèle, au moyen d'une colonne de N soustracteurs suivis d'une colonne de N multiplieurs carré. Les N sorties obtenues sont alors connectées aux N entrées d'un additionneur série (architecture arborescente simple). Il est intéressant de noter que les poids W_{ij} ont été figés sur la ROM interne du FPGA et les entrées X_i sont directement connectées aux blocs SRAM externes de la carte RC1000-PP.

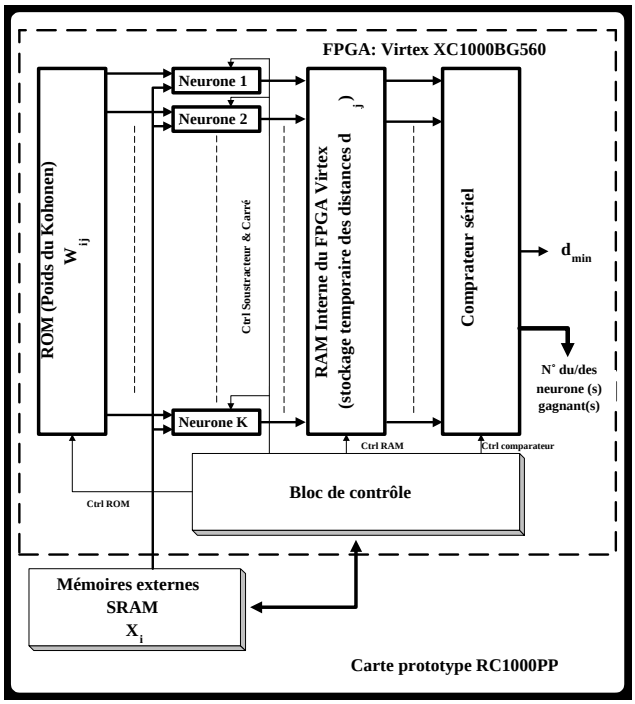

FIG. 8 – *Architecture du Réseau de neurones de Kohonen*

Architecture globale de la SOM implantée sur le FPGA. L'architecture globale du réseau est formée de 4 blocs (figure 8) :

- Une colonne de $K = 25$ neurones (détaillés dans la sous-section précédente).
- Un bloc de mémoires SRAM internes pour le stockage temporaire des distances euclidiennes de chaque neurone de la couche de sortie. Ce bloc sera nécessaire pour changer le sens de la propagation sérielle des bits (passer d'une propagation LSBF à MSBF). Ce changement est nécessaire pour le fonctionnement du comparateur.
- Un comparateur sériel pour la sélection de la distance euclidienne minimale d_i correspondant au neurone gagnant. L'architecture du comparateur est spécifique aux besoin de la SOM : elle permet d'extraire en série (en mode MSBF) simultanément la distance et le numéro correspondant au neurone gagnant (figure 9).
- Une unité de contrôle global qui supervise et pilote le flot de signaux entre les différentes entités. Le contrôleur est basé essentiellement sur un automate fini dont

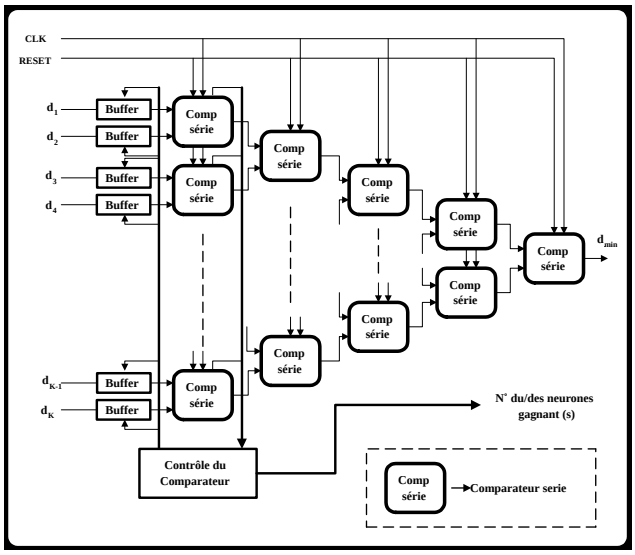

FIG. 9 – *Comparateur*

l'architecture de base s'articule autour d'un compteur d'états.

4 Résultats

Pour valider nos choix matériels et algorithmiques, on a exploité une carte de test (carte de prototypage rapide) qui se base sur une électronique d'acquisition et un FPGA. Cette plate-forme s'applique pour valider les performances d'intégration sur cible. Nous avons présenté dans la sous-section 3.1 les caractéristiques techniques de cette plate-forme.

4.1 Temps d'exécution et surface d'implantation

Dans un premier temps, les performances matérielles ont été mesurées indépendamment pour chaque module du réseau, puis le réseau complet a été étudié à son tour. Le tableau 1 regroupe les données de temps et d'espace d'implantation des différents opérateurs arithméthiques, blocs de contrôle, blocs mémoires, ainsi que du réseau complet, en fonction des solutions arithmétiques envisagées (sériel en mode LSBF ou MSBF en-ligne, ou parallèle). Les résultats sont obtenus avec une fréquence de 25 MHz, lors de la synthèse avec l'outil ISE 5.2 de Xilinx des différentes architectures étudiées dans cet article.

Le FPGA disponible sur la carte RC1000-PP s'avère trop petit pour accueillir la SOM de Kohonen constituée entièrement d'opérateurs en arithmétique parallèle, avec une précison sur 8 bits. En effet, un neurone nécessite alors 1390 slices, ce qui limite à 9 le nombre de neurones simultanément implantables. De plus, une telle solution technologique demande un nombre de ports d'entrées/sorties qui dépasse le nombre d'IOB disponibles.

La deuxième solution que nous avons étudiée est l'utilisation d'une arithmétique sérielle MSBF (ou en-ligne). Une telle arithmétique sérielle permet un calcul poids fort en tête de tous les opérateurs arithmétiques et fonctions élémentaires grâce à l'utilisation d'un système redondant de représentation des nombres, cf [2, 4]. Cette approche nécessite environ 19000 slices pour implanter tout le réseau de Kohonen (23 entrées et 5x5 neurones sur la couche de sortie), ce qui dépasse les ressources du FPGA utilisé.

L'utilisation d'une architecture sérielle LSBF, en exploitant un pipeline au niveau du bit, s'avère la plus adaptée à notre application. En effet une implantation entièrement parallèle de tout le réseau nécessite 12886 slices avec un temps de prise de décision T_{exec} de 1,37 μ s pour une fréquence d'horloge de 25 MHz (voir le chronogramme de la figure 10), au prix d'une intégration d'une SRAM interne utile pour le changement de sens de propagation des bits LSBF à MSBF : un comparateur série fonctionne toujours en MSBF.

4.2 Etude de la consommation

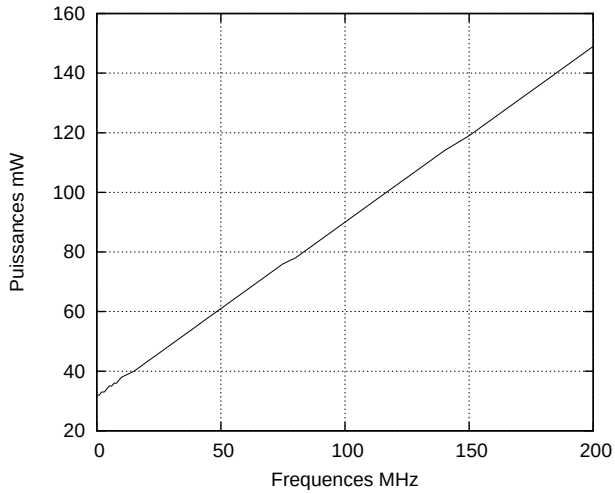

FIG. 11 – *Variation de la consommation*

Nous avons aussi abordé dans notre étude le problème de la consommation de la SOM de Kohonen, pour effectuer des comparaisons avec des solutions conventionnelles. Ce point est très important mais assez complexe car un grand nombre de paramètres entrent en compte dans cette étude. La figure 11 montre la variation de la consommation en fonction de la fréquence d'horloge du FPGA.

A la fréquence minimale tolérée par notre carte, soit 400 KHz, le temps d'exécution est de 86 μs, ce qui est encore largement suffisant pour les contraintes temps-réel de la détection d'hypovigilance. La consommation est alors de 32 mW (soit 1000 à 2000 fois moins que les processeurs actuels). Si on tient compte de la période de veille autorisée lors d'une utilisation concrète avec prise de décision toutes les secondes, la consommation reste inférieure à 4 μW.

Type du bloc	T_{exec} (μ s)	densité d'intégration		
		Slices (2 par CLB)	LUT	FF
Soustracteur/additionneur [LSBF]	0,32	1	2	1
Soustracteur/additionneur en-ligne [MSBF]	0,32	3	4	1
Soustracteur/additionneur [parallèle]	0,04	4	8	8
additionneur [LSBF] 23 entrées	0,32	24	48	24
additionneur en-ligne [MSBF] 23 entrées	0,32	72	96	24
additionneur [parallèle] 23 entrées	0,04	91	179	0
Calcul du carré [LSBF]	0,51	22	21	26
Calcul du carré en-ligne [MSBF]	0,64	26	45	16
Calcul du carré [parallel]	0,04	40	60	16
Comparateur en-ligne [MSBF]	0,20	90	107	90
ROM (sauvegarde W_{ij})	-	3	4	0
SRAM temporaire (sauvegarde d_j pour le cas LSBF)	0,4	13	25	0
Contrôleur de l'architecture	-	16	27	14
$(X_i - W_{ij})^2$ [LSBF]	0,60	24	23	28
$(X_i - W_{ij})^2$ [parallel]	0,08	44	68	8
Un neurone : $\sum_{i=0}^{N-1}(X_i - W_{ij})^2$ [LSBF]	0,69	614	582	690
Un neurone : $\sum_{i=0}^{N-1}(X_i - W_{ij})^2$ [parallèle]	0,12	1390	1936	568
SOM de Kohonen 23 entrées et 5x5 Sorties [LSBF]	1,37	12286	14557	17659
SOM de Kohonen 23 entrées et 2x2 Sorties [LSBF]	1,37	4320	3269	2882

TAB. 1 – *Performances de l'implantation des cartes auto-organisatrices de Kohonen*

5 Conclusion

Dans cet article, nous avons présenté les travaux d'implantation matérielle numérique sur circuit reprogrammable d'un réseau de neurones de type carte auto-organisatrice de Kohonen appliqué à la détection embarquée de l'hypovigilance. Les contraintes d'utilisation en ambulatoire temps-réel expliquent la recherche d'une implantation très basse-puissance. Le choix d'un support FPGA est avant tout justifié par sa flexibilité, son grain très fin de parallélisme, ainsi que par une consommation assez réduite. L'implantation recherchée devait minimiser les entrées/sorties, et maximiser le parallélisme interne pour aboutir à une consommation très basse. Les choix technologiques opérés ont donc eu pour objectif de parvenir à une solution entièrement parallèle où tous les neurones fonctionnent simultanément et traitent toutes leurs entrées en parallèle.

Les performances obtenues pour notre implantation remplissent largement les objectifs visés (implantation très basse-puissance par architecture totalement parallèle). Nous développons par ailleurs une carte plus spécifique à base de deux FPGA Xilinx Spartan de taille réduite et d'un microcontrôleur, qui est directement reliée aux capteurs EEG analogiques. L'implantation finale est destinée à combiner dans un système portable les atouts de notre implantation très basse puissance et les enseignements tirés des observations in-situ de cette carte spécifique.

Références

[1] K. Ben Khalifa, M.H. Bedoui, L. Bougrain, R. Raytchev, M. Dogui, and F. Alexandre. Analyse et classification des états de vigilance par réseaux de neurones. Technical report, INRIA, 2003. rapport de recherche RR-4714.

[2] M.D. Ercegovac. On-line arithmetic: an overview. In *SPIE, Real Time Signal Processing VII*, pages pp 86–93, 1984.

[3] X. Fang, P. Thole, J. Goppert, and W. Rosenstiel. A hardware supported system for a special online application of self-organizing map. In *Proceedings of ICNN96*, pages 1040–1045, 1996.

[4] B. Girau and A. Tisserand. MLP computing and learning on FPGA using on-line arithmetic. *Int. Journal on System Research and Information Science*, special issue on *Parallel and Distributed Systems for Neural Computing*, 9(2-4), 2000.

[5] P. Ienne and M.A. Viredaz. Bit-serial multipliers and squarers. *IEEE Transactions on Computers*, 43(12):1445–1450, 1994.

[6] S.L. Joutsiniemi, S. Kaski, and T.A. Larsen. Self-organizing map in recognition of topographic patterns of EEG spectra. *IEEE Transactions on Biomedical Engineering*, 42:1062–1068, 1995.

[7] T. Kohonen. *Self-Organization Maps*. Springer, 2001.

[8] D. Macq, M. Verleysen, P. Jespers, and J.D. Legat. Analog implementation of a kohonen map with on-chip learning. *IEEE Trans. on Neural Networks*, 4(3):456–461, 1993.

[9] M. Porrmann, H. Kalte, U. Witkowski, J.-C. Niemann, and U. Rückert. A dynamically reconfigurable hardware accelerator for self-organizing feature maps. In *Proc. of the 5th World Multi-Conference on Systems, Cybernetics and Informatics, SCI 2001*, vo-

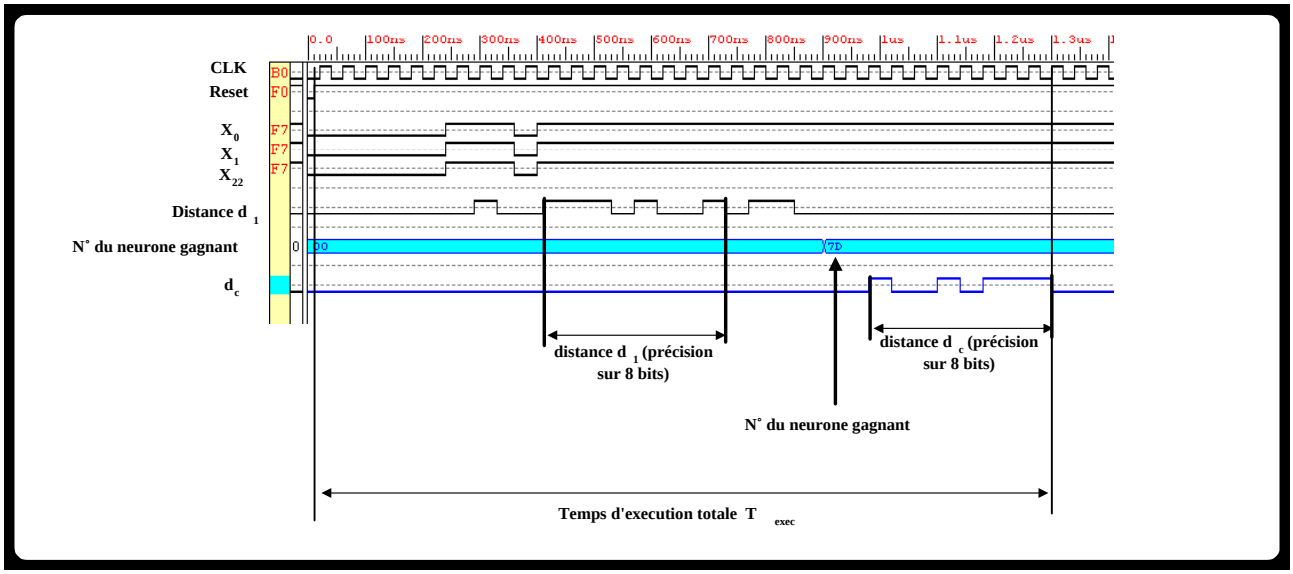

FIG. 10 – *Chronogramme*

lume 3, pages 242–247, Orlando, Florida, USA, July 2001.

[10] H. Speckmann, P. Thole, and W. Rosenstiel. COKOS: A coprocessor for kohonen's selforganizing map. In *Proceedings of ICNN93*, pages 1040–1045, 1993.

[11] N. Vassilas, P. Thiran, and P. Ienne. On modifications of kohonen's feature map algorithm for an efficient parallel implementation. In *Proceedings of ICNN96*, pages 932–937, 1996.

[12] A. Vuckovic, V. Radivojevicb, A.C.N. Chena, and D. Popovica. Automatic recognition of alertness and drowsiness from EEG by an artificial neural network. *Med Eng Phys*, 24(5):349–360, 2002.

[13] Xilinx, editor. *The Programmable Logic Data Book*. Xilinx, 2002.