N° d'ordre : 07/2008-M/EL

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

« HOUARI BOUMEDIENE »

FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

Filière : ELECTRONIQUE

Spécialité : Instrumentation Electronique

Par : Mr OUNAS Mouloud

Sujet

Implémentation Hardware d'algorithme Séparateur de Signaux Sur Circuit Reconfigurable FPGA

Soutenu le: 24 /01/2008, devant le Jury composé de :

M[r] M. ATTARI :	Professeur,	(USTHB) Alger	Président
M[ême] R. TOUHAMI	Professeur,	(USTHB) Alger	Directeur de thèse
M[r] A.R. BABA ALI	Maître de conférences,	(USTHB) Alger	Examinateur
M[r]. DEBYECHE	Maître de conférences,	(USTHB) Alger	Examinateur

AVANT PROPOS

Je remercie tout d'abord Dieu le tout puissant pour m'avoir aidé à réussir mes études et m'a guidé dans la bonne voie pour l'aboutissement de mon projet de Magistère dans de bonnes conditions.

Le travail présenté dans ce mémoire a été réalisé au sein de l'équipe Systèmes Radio-Fréquences et Micro-ondes du laboratoire d'Instrumentation de la Faculté d'Electronique et d'Informatique de l'Université des Sciences et de la Technologie Houari Boumediene sous la direction de Dr. Rachida TOUHAMI, Maître de Conférences à l'U.S.T.H.B. Qu'elle soit chaleureusement remerciée en lui exprimant toute ma gratitude et mon respect.

J'adresse mes sincères remerciements à Monsieur M. Attari, Professeur à l'U.S.T.H.B, pour avoir accepté de présider le jury.

Que Messieurs, A.R. Baba Ali, Maître de Conférences à l'U.S.T.H.B, M. Debyeche, Maître de Conférences à l'U.S.T.H.B, et Madame, L. Fergani, Chargée de cours à l'U.S.T.H.B soient remerciés pour avoir accepté d'examiner ce mémoire.

Mes remerciements s'adressent aussi à Monsieur Salim Chitroub, Maître de Conférences à l'U.S.T.H.B, pour l'aide pratique qu'il m'a apportée concernant les algorithmes de traitement numérique des signaux.

Et pour terminer, je ne saurai exprimer ma profonde gratitude à mes collègues chercheurs de l'équipe Systèmes Radio-Fréquences et Micro-ondes qui m'ont aidé dans la réalisation de cette thèse et à tous ceux qui m'ont apporté leur amitié et leur soutien moral pendant la préparation de ce mémoire de Magister.

Table des matières

Résumé

L'algorithme de l'Analyse en Composantes Indépendantes (ACI) est un outil mathématique très performant inspiré de la théorie des réseaux de neurones qui permet de supporter la puissance de calcul de traitement avec efficacité pour la séparation de signaux. L'algorithme de l'ACI exige un nombre important d'opérations arithmétiques pour le calcul de l'algorithme. Dans cette optique, grâce à l'avancée technologique dans la fabrication des circuits intégrés VLSI, il a été envisagé la réalisation des applications de séparation de signaux en utilisant des implémentations hardwares pour le traitement de l'algorithme ACI en temps réel. Une solution alternative envisagée est offerte par les circuits FPGA programmables qui permettent de diminuer les temps de conception et de produire des prototypes à moindre coût. Vu la limitation en nombre de ressources matérielles dans un circuit FPGA, une méthodologie de conception a été appliquée aux circuits matériels afin d'effectuer des calculs d'optimisation de l'algorithme. D'autre part, une conception d'une architecture matérielle d'un seul neurone a été employée en langage VHDL afin de permettre l'optimisation de la surface matérielle du circuit FPGA. Un traitement séquentiel de calculs du neurone dans le circuit FPGA nous a permis d'effectuer le traitement de 1000 échantillons à 2 signaux d'entrée pour la séparation de 2 signaux indépendants en sortie. Compte tenu du temps d'exécution dans le cycle d'apprentissage qui est de l'ordre de 17.3us, la fréquence d'échantillonnage maximale a atteint 57Khz.

Mots clés : Analyse en Composantes Indépendantes, Séparation de signaux, Réseaux de neurone, FPGA, Implémentation, Optimisation, VHDL, Temps d'exécution, Taux des échantillons, Fréquence d'échantillonnage, Temps réel.

Abstract

The Independent Component Analysis (ICA) algorithm is inspired from the theory of neural network algorithm which supported high power computation of performances for Blind Signal Separation (BSS). Thus, the ICA algorithm needs an important computation of arithmetic operation of the algorithm. In this context, due to the advanced technology of the VLSI digital circuit, it has been envisaged the realisation of BSS applications which used hardware implementation to process the ICA algorithm in real time. An alternative solution that has been envisaged is offered by the FPGA programmable device reduces the development time of design with low cost of prototype manufacturing. Due to the limitation number of hardware resources of FPGA device, the methodology of design has been applied to hardware circuits, which used the optimisation computation of the algorithm. On the other side, the design of the hardware architecture of neurone has been described with VHDL language to optimize the hardware area size of FPGA device. Hence, the sequential processing of the neuron computation in FPGA device have allowed to process 1000 samples of 2 input signals for separating 2 independent output signals. Considering the execution time of 17.3us obtained from the learning cycle, the maximum sampling frequency has been reached the value of 57 KHz.

Key words: Independent Component Analysis, Blind Signal Separation, Neural network, FPGA, Implementation, Optimisation, VHDL, Execution time, Sample rates, Sampling frequency, Real time.

Introduction générale

L'avancée technologique enregistrée dans la conception des circuits intégrés VLSI a permis l'intégration de circuits digitaux complexes. Cette technologie est constituée par une densité d'intégration de circuits ayant des structures complexes en silicium avec des dimensions de l'ordre du micron. Les circuits intégrés numériques qui réalisent des applications complexes dans le traitement numérique des signaux effectuent des fonctions pour l'acquisition d'une grande quantité d'informations. D'où, la nécessité de l'utilisation de ces traitements dans des applications qui sont devenues une discipline très active des circuits intégrés numériques dans plusieurs domaines, notamment, la télédétection, communication, signaux de mélange, réseaux de neurones, ingénierie biomédicale, radar, traitement d'images et de la parole.

Afin de traiter numériquement des signaux complexes [1] de façon efficace, les scientifiques ont introduit une méthode de calcul pour assurer un processus de traitement intelligent de l'algorithme qui permet d'effectuer l'acquisition d'une grande quantité d'informations de données. Cependant, l'emploi d'un outil mathématique qui fournit des fonctions de calcul non linéaires et parallèles (exigés par les traitements de problèmes complexes dans le traitement numériques des signaux) a été trouvé comme une solution alternative à la difficulté de traitement rencontrée dans les calculs linéaires en utilisant des quantités de calculs algébriques importantes pour le traitement du problème.

Ainsi, l'utilisation des réseaux de neurones artificiels a été envisagée dans le traitement numérique des signaux pour apporter des solutions en terme de performances, permettant ainsi un traitement de ses applications en temps réel. D'où, l'habilité des réseaux de neurones qui est basée sur le traitement d'apprentissage, a la propriété d'avoir une fonction d'approximation universelle. Ceci permet alors d'effectuer l'ajustement des poids synaptiques par des algorithmes de calculs parallèles moins complexes [2] de la fonction non linéaire, appelée fonction d'activation.

Dans plusieurs applications en traitement numérique des signaux, la séparation de signaux a été utilisée pour la reconstitution de signaux sources à partir de signaux de mélange issus de sources de signaux multi capteurs. Cette séparation est effectuée par des algorithmes d'apprentissage des réseaux de neurone sur les signaux de mélange inconnus non observables et supposés indépendants [1-2].

Les applications de séparation de signaux qui caractérisent les informations partagées dans les signaux de mélange, sont souvent réalisées en utilisant la technique par l'analyse en composantes indépendantes. Ainsi, le traitement de la séparation de signaux par l'analyse en

composantes indépendantes (ACI) effectue des calculs sur les informations statistiques des signaux [3]. Alors, l'idée principale du traitement par cette technique réside sur la minimisation des calculs de la distribution gaussienne dans les signaux de mélanges pour l'estimation des signaux indépendants [4-5].

Les implémentations des algorithmes d'apprentissage des réseaux de neurones dans un calculateur, nécessite un temps de calcul élevé à cause des opérations séquentielles produites par des architectures de type Von Neumann qui permettent l'exécution de l'algorithme avec une très faible vitesse de calculs. Pour résoudre ce problème, une solution alternative a permis d'envisager des implémentations hardware pour le traitement parallèle des calculs. Ainsi, à travers le développement de la technologie VLSI, le circuit matériel qui avait permis l'intégration d'un nombre très dense en portes logiques, dans une très petite surface en silicium, a pu implanter des traitements parallèles présentés dans les fonctions d'activation des neurones, interconnectés entre eux par des liaisons en forme d'architectures de types réseaux de neurones.

Jusqu'ici, les concepteurs employaient la technique des circuits programmables (FPGA), à cause de la flexibilité offerte dans la programmation des interconnexions entre les blocs des circuits configurables. D'où, une conception rapide des prototypes qui permet de réduire les temps et les coûts de développement est suivie par la fabrication en série des ASICs (Circuits Intégrés à Application Spécifique) pour atténuer les coûts de fabrication. En outre, ils permettent aussi la vérification du fonctionnement des algorithmes d'apprentissage des réseaux de neurones en utilisant l'avancée technologique des circuits FPGA, avec des coûts de consommation très réduits [6-8].

Afin d'éviter le déficit des ressources matérielles dans les circuits FPGA, le concepteur de l'implémentation hardware, cherche une méthodologie de conception des circuits matériels qui utilise des calculs d'optimisation de l'algorithme afin de construire des circuits logiques d'implémentations hardware de taille réduite ou de pouvoir implémenter le maximum du traitement des neurones sur la même surface du circuit FPGA. En outre, l'optimisation de la surface des circuits FPGA qui consomme le même nombre de ressources matérielles permet d'améliorer les performances du circuit matériel mais aussi augmente la consommation du coût de développement [9].

Dans ce travail, une étude a été développée pour la réalisation d'une architecture d'un circuit logique moins complexe afin de minimiser sa taille. En se basant sur les informations des articles récemment publiés [10-14], nous avons illustré une méthodologie de conception par des calculs d'implémentation qui permettent l'optimisation de calculs de l'algorithme par l'ACI pour la séparation de signaux.

Pour réduire la complexité de l'architecture du circuit logique d'implémentation, nous avons adopté une conception hardware basée sur la représentation des nombres en format de point fixe. On a aussi utilisé la fonction sigmoïde appliquée aux fonctions d'activation non linéaires des réseaux de neurones [15-16]. Cependant, dans le but de fournir des implémentations hardwares simples en préservant des compromis entre les performances du circuit matériel et celles du traitement de la séparation de signaux, nous avons construit une méthodologie de conception pour le traitement d'un seul neurone [17-21] qui minimise la complexité de la taille de l'architecture du circuit matériel. En outre, dans notre conception, le traitement du circuit matériel a été limité pour un traitement élevé de neurones afin d'éviter le temps élevé de la latence produits par leurs traitements séquentiels.

Cette conception a été réalisée en langage VHDL [22] sur une plate forme constituée par un ensemble d'outils CAO pour la synthèse des circuits logiques d'implémentation ainsi que son placement et son routage dans les circuits FPGA. Les résultats de simulation obtenus ont été validés par comparaison à celles obtenus dans la littérature.

Le mémoire est organisé comme suit :

Le chapitre I présente la théorie de base du traitement de la séparation de signaux par l'analyse en composantes indépendantes.

Dans le chapitre II, nous avons défini la méthodologie de conception des circuits matériels.

Le chapitre III, illustre la conception de la mise en œuvre des calculs d'implémentation de l'algorithme (ACI).

Dans le chapitre IV, nous avons présenté la mise en œuvre de l'adéquation algorithme et architectures d'implémentation hardware.

Le chapitre V, illustre la conception de l'architecture d'implémentation sur le circuit matériel en effectuant des traitements de simulation sur l'architecture d'implémentation et le circuit matériel.

Communications générées par ce travail :

1. **M. Ounas**, R.Touhami, and S. Chitroub," ***Design of Hardware Implementation Based ICA training algorithm for Blind Signal Separation ".*** Proc. of Information and Communication Technologies International Symposium, ICTIS'07, Fes, Morocco, April 3- 4-5, 2007.

2. **M. Ounas**, R.Touhami, and S. Chitroub," ***Architecture Design of Digital Circuit based ICA training algorithm for Hardware Implementation".*** La 5^{éme} Conférence sur le Génie Electrique, CGE'05, EMP Bordj-El-Bahri, Alger, Algérie, 16-17 avril, 2007.

3. **M. Ounas**, R.Touhami, and M.C.E. Yagoub***," Low Cost Architecture of Digital Circuit for FPGA Implementation Based ICA training algorithm of Blind Signal Separation".*** Proc. of International Symposium on Signals, Systems, and Electronics, ISSSE'2007, Montreal, Canada, July 30-August 2, 2007.

4. **M. Ounas**, S.Chitroub, et R. Touhami, *''**Méthodologie de Conception d'une Implémentation Matérielle d'un Algorithme par l'Analyse en Composantes Indépendantes pour la Séparation de Signaux''.*** Le 21ème Colloque International du Traitement du Signal et des Images, GRETSI'2007, Troyes, France, 11-14 Septembre, 2007.

5. **M. Ounas,** and R. Touhami, "***Design Methodology of Hardware Implementation Based ICA training algorithm for Blind Signal Separation".*** Instrumentation Laboratory Workshop, Journées du Laboratoire d'Instrumentation, JLINS'07, Université USTHB, Alger, Algérie, 30 et 31 Octobre 2007.

Chapitre 1 :

Théorie de base des calculs du traitement de la séparation de signaux

1.1 Introduction

Dans la communauté du traitement du signal, un grand nombre de calculs statistiques a été développé pour résoudre le problème de séparation de signaux. Intuitivement, la séparation de signaux est obtenue lorsque aucune information commune n'a été trouvée entre les signaux à séparer. Dans cette étude nous employons les calculs statistiques d'indépendance spatiale d'ordre supérieur à deux dans lesquels l'apprentissage des réseaux de neurone pour l'ajustement des poids synaptiques est réalisé. Dans cette approche, nous avons mis en évidence un traitement par l'intermédiaire du critère d'optimisation qui agit en particulier sur la connaissance de la mesure des fonctions de densité de probabilité de chaque signal de mélange afin de contrôler le degré d'indépendance entre les signaux de mélange par les calculs statistiques et de la distribution non gaussiennes [4]-[5]-[23]. Il existe une autre approche, dans laquelle on peut aussi utiliser une formulation qui effectue la décorrélation spatiale et temporelle [24] où l'ordre des calculs de l'indépendance statistique a été limité à deux. Ainsi, nous supposons que les signaux des sources indépendantes ont des calculs statistiques stationnaires. Récemment, une autre approche a été proposée qui est basée sur une formulation de signaux non stationnaires [25].

Dans cette étude, on va se limiter aux applications ayant des calculs statistiques stationnaires. Dans ce sens, Comon [26] avait proposé les concepts de base théorique du traitement par l'analyse en composantes indépendantes (ACI) qui utilise les calculs statistiques pour l'estimation de signaux indépendants. Il a utilisé la fonction Kurtosis comme fonction de contraste qui optimise les calculs statistiques pour l'évaluation des signaux indépendants. La séparation de signaux a été accomplie en maximisant la valeur absolue de la sortie du Kurtosis. L'avantage de la méthode qui utilise le principe de la fonction de contraste comme une fonction d'optimisation, ne s'appuie pas essentiellement sur la connaissance de la mesure de la fonction densité de probabilité. Parmi les méthodes qui maximisent la fonction de contraste, l'algorithme de FastICA [27] a été récemment développé. Cet algorithme a été formulé sur la base d'un principe d'apprentissage itératif du point fixe (fixed point algorithm). Pour la première fois, l'ACI a été mis en œuvre par Hérault et Jutten [28] qui a constitué une première solution pour le traitement du problème de la séparation de signaux indépendants.

Dans le cas de notre étude, on va illustrer le traitement de l'algorithme de l'ACI qui utilise le calcul de l'indépendance spatiale par la mesure de la convergence de KulBuck Leibler [23]. Cette mesure s'appuie sur la connaissance de la différence entre la densité de probabilité conjointe des signaux de sortie et de la densité de probabilité marginale de chaque signal de sortie à séparer, mettant en œuvre dans la dérivée de cette mesure une équation d'apprentissage du Gradient Naturel pour l'ajustement des poids synaptiques.

1.2 Le principe de traitement de la séparation de signaux

Le principe de traitement de la séparation de signaux est illustré par le schéma de la Figure1.1 :

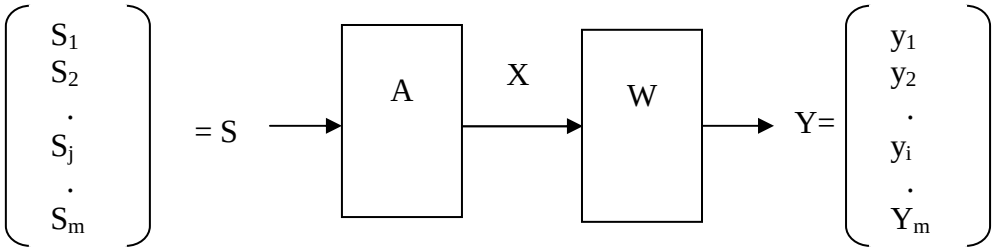


Figure 1.1. Principe de traitement de la séparation de signaux

où ; m désigne la valeur du nombre de signaux sources supposés indépendants qui représente la dimension du vecteur S, A est une matrice de mélange de dimension (n x m), et X le vecteur des signaux de mélange de dimension n tel que :

$$X=A.S \tag{1}$$

Y est le vecteur d'estimation des signaux séparés, supposés indépendants et égale aux signaux sources du vecteur S dont les valeurs sont réajustés, pendant l'ajustement des valeurs des poids synaptiques contenant les coefficients de la matrice W, selon l'équation:

$$Y=W.X \tag{2}$$

En remplaçant (1) dans (2), conduit à l'expression suivante :

$$Y=W.\ A\ .S \tag{3}$$

Le vecteur Y représentant les signaux de sortie est caractérisé par l'estimation des signaux supposés assez indépendants pour se rapprocher des valeurs des signaux sources réels du vecteur S. Le vecteur de signaux de sortie sera supposé à priori égal au vecteur de signaux d'entrées. Ce qui permet d'identifier la valeur de la matrice (W.A) comme étant égale à l'unité, dans laquelle la valeur de la matrice W est déterminée théoriquement selon l'équation :

$$W= A^{-1} \tag{4}$$

qui constitue la valeur de la matrice inverse de A.

1.3 La séparation des signaux instantanés et l'algorithme ACI

Cette section, illustre le calcul théorique de l'algorithme par l'ACI appliqué à la séparation de signaux instantanés. Cependant, on considère le problème de la séparation de signaux instantanée qui pourra simplifier la présentation générale des concepts et des méthodes du traitement de l'algorithme pour lesquels ni les effets de dispersion ni de la convolution sont présentées. Jusqu'à présent, le traitement de la séparation de signaux a été utilisé dans le cas des signaux non bruités. Soit m le nombre de signaux de mélange, les signaux x_i sont calculés selon une combinaison linéaire de n signaux S_j inconnus supposés mutuellement et statistiquement indépendants. L'expression des signaux de mélange x_i est donné par :

$$x_i = \sum_{j=1}^{n} A_{ij} S_j \text{ , } (i=1, 2,\ldots,m) \tag{5}$$

où A_{ij} représente le coefficient de mélange inconnu de la j^{ieme} source du signal indépendant au i^{eme} signal de mélange, supposé fourni par un capteur, avec ($m \geq n$). En pratique, par simplification, on pose ($m=n$) pour permettre d'effectuer les calculs sur des matrices comportant des nombres de lignes et de colonnes égales. La fonction densité de probabilité est supposée de distribution non gaussienne, à priori, pour au moins (n-1) signaux sources S_j. Le coefficient de l'ajustement des poids synaptiques w_{ij} qui sera estimé par l'algorithme d'apprentissage (ACI) est utilisé pour l'évaluation du calcul de l'entrée synaptique u_i du réseau de neurones dans l'expression de l'équation suivante:

$$u_j = \sum_{i=1}^{n} w_{ij} x_j \tag{6}$$

En fonction de la valeur de l'entrée synaptique calculée précédemment, le calcul de la sortie y_j estimé du réseau de neurones sera calculée en utilisant sa fonction d'activation non linéaire f. Alors, l'expression de la sortie y_j du réseau de neurones est déterminée selon l'équation de la fonction suivante:

$$y_j = f(u_j) \tag{7}$$

1.4 L'algorithme d'apprentissage par l'approche de la descente du gradient naturel

On s'intéresse dans cette partie, le traitement des calculs de l'ajustement des poids synaptiques w_{ij} par l'algorithme (ACI). La procédure d'ajustement dépend de la connaissance a priori sur l'information de l'état des signaux sources que peut fournir les calculs de leurs dépendances statistiques. De ce fait, on va exploiter l'approche qui utilise la formulation de

l'indépendance spatiale pour l'apprentissage du réseau de neurones qui permet l'ajustement des poids synaptiques par l'équation de la descente du gradient naturel [4], [23]. L'idée de base, concernant le traitement de cette formulation est d'ajuster les valeurs des poids sachant que la densité de probabilité de distribution conjointe $P_y(y)$ sera égale à la densité de probabilité conjointe du modèle de distribution $P1_y(y)$ supposé de référence connu au moment de la convergence des poids synaptiques. Cardoso [4] a montré que tous les calculs utilisés dans le traitement de ces formulations sont unifiés par les calculs du principe de la divergence de Kullback–Leibler, selon l'expression :

$$D(\mathrm{Py} \parallel \mathrm{P1y}) = \int \mathrm{Py(y)} \log\left(\frac{\mathrm{Py(y)}}{\mathrm{P1y(y)}}\right) dy \tag{8}$$

où, $P_y(y)$ indique la mesure actuelle de la distribution de la fonction densité de probabilité des signaux du vecteur de sorties Y. En effet, l'équation (8) permet de mesurer la distance entre $P_y(y)$ et $P1_y(y)$, malgré que cette mesure est asymétrique. On va mesurer l'indépendance statistique des signaux par la mesure de l'entropie relative $H(y)$ qui est définie par :

$$H(Y) = -\int \mathrm{Py(y)} \log(\mathrm{Py(y)}) dy \tag{9}$$

En supposant que tous les signaux S_i de densité de probabilité identiquement distribués, le choix de la valeur $P1_y(y)$ est gouverné par la supposition et la connaissance à priori des signaux sources S_i du vecteur S, dont la valeur actuelle est supposée égale au produit des valeurs de la densité de probabilité marginale de chaque signal de sortie, selon :

$$P1y(y) = \prod_{i=1}^{N} P1y_i(y_i) \tag{10}$$

Une équation qui assure la convergence de l'ajustement des poids synaptiques lorsque les signaux estimés deviennent indépendants selon l'obtention de l'égalité de l'équation suivante :

$$Py(y) = \prod_{i=1}^{N} P1y_i(y_i) \tag{11}$$

On peut définir les entropies marginales par l'équation suivante:

$$H_i(Y_i) = -\int \mathrm{P1y_i(y_i)} \log(\mathrm{P1y_i(y_i)}) dy \tag{12}$$

Donc, la fonction d'optimisation qui permet de minimiser l'équation de la divergence de Kullback–Leibler est de la forme :

$$J(w) = -\log(\det|w| \prod_{i=1}^{N} Py_i(y_i)) \tag{13}$$

Avec: $$H(Y) = H(x) + \log(|\det(w)|) \quad (14)$$

ΔW, représente l'écart entre l'ajustement des poids synaptiques de l'itération présente et l'ajustement des poids synaptique de l'itération précédente, et est donnée selon l'équation:

$$\Delta W = -\mu\partial \frac{J(W)}{\partial W} \quad (15)$$

Ceci permet de retrouver l'équation d'apprentissage du gradient naturel pour l'ajustement des poids synaptiques :

$$\Delta W = \mu\left[W^{-T} + \varphi(u)x^{T}\right] \quad (16)$$

φ désigne la fonction non linéaire d'apprentissage de l'algorithme, μ est un nombre supposé positif et inférieur à l'unité, et illustre la valeur du facteur d'apprentissage de l'algorithme. W représente la matrice qui contient les coefficients de l'ajustement des poids synaptiques w_{ij}, x est le vecteur d'entrée des signaux de mélange x_i, u est le vecteur des entrées synaptiques u_i, et (W^{-T}) est l'opération transposée de la matrice inverse.

AMARI [23] par l'approche du gradient naturel et CARDOSO [4] par celle du gradient relatif, ont proposé une modification de l'équation (16) pour le calcul de l'apprentissage des poids synaptiques. Ainsi, ils ont évité le calcul complexe donné par la matrice transposée inverse de l'ajustement des poids synaptiques (W^{-T}) en permettant d'améliorer la convergence de l'algorithme. La nouvelle expression de l'équation d'apprentissage des réseaux de neurone pour l'ajustement des poids synaptiques est donnée par:

$$\Delta W = \mu\left[I + \varphi(u)u^{T}\right]W \quad (17)$$

où I désigne la matrice unité.

En effectuant un développement du produit matriciel et du produit vectoriel pour le calcul de chaque coefficient de l'ajustement w_{ij} de la matrice W donnée pour chaque valeur de i et j selon (i=1,2,…,n et j=1,2,…,n) :

$$W_{ij}^{+} = W_{ij} + \mu\left[1 + \sum_{k=1}^{n} \varphi(u_k)u_k\right]W_{ij} \quad (18)$$

W_{ij}^{+} représentant le coefficient de l'ajustement des poids synaptiques de l'itération présente, et W_{ij} représentant le coefficient de l'ajustement des poids synaptiques de l'itération précédente.

1.5 Calcul de l'approche de l'information mutuelle

Bell et Sejnowski [5] ont proposé une approche qui permet de maximiser le transfert de l'information entre l'entrée et la sortie du réseau de neurones par l'algorithme d'apprentissage, en utilisant la fonction d'activation non linéaire du neurone *f*. La théorie qui maximise l'information mutuelle a été obtenue en utilisant le développement de l'équation suivante :

$$I(Y, X) = H(Y) - H(Y / X) \tag{19}$$

I(Y, X) est l'information mutuelle qui existe entre la sortie et l'entrée, *H(Y)* l'entropie de la sortie, et *H(Y/X)* celle de la sortie qui ne provient pas de l'entrée. Le problème de traitement de l'information mutuelle, revient à maximiser cette information pour le contrôle de l'ajustement des poids synaptiques *w*. Comme l'entropie de *H(Y/X)* ne dépend pas de la variable d'ajustement du poids synaptique *w*, la maximisation de l'information mutuelle s'obtient en utilisant la dérivée de l'équation (6) et devient selon l'équation suivante:

$$\frac{\partial}{\partial w} I(Y, X) = \frac{\partial}{\partial w} H(Y) \tag{20}$$

Ainsi, la valeur de la fonction non linéaire d'apprentissage utilisée dans l'équation d'apprentissage de l'ajustement des poids synaptiques sera donnée par l'équation suivante :

$$\varphi(u_i) = \frac{f''(u_i)}{f'(u_i)} \tag{21}$$

f' et f'' désignent respectivement la valeur de la dérivée première et de la dérivée deuxième de la fonction d'activation *f*, à laquelle on assigne la fonction sigmoïde selon:

$$f(u_i) = \frac{1}{1 + e^{-u_i}} \tag{22}$$

Les expressions de la dérivée première et de la dérivée deuxième de la fonction sigmoïde deviennent, respectivement :

$$f'(u_i) = f(u_i)(1 - f(u_i)) \tag{23}$$

$$f''(u_i) = f'(u_i)(1 - 2f(u_i)) \tag{24}$$

Le calcul de la valeur $\varphi(u_i)$ et de la fonction non linéaire d'apprentissage devient selon l'expression de l'équation suivante:

$$\varphi(u_i) = (1 - 2f(u_i)) \tag{25}$$

où, $f(u_i)$ est la fonction estimée y_i de la sortie du réseau de neurones.

1.6 Conclusion

Le traitement de la séparation des signaux indépendants a eu récemment un intérêt croissant grâce aux développement rapides dans le domaine des télécommunications, traitement des images, et traitement des signaux biomédicaux. Le mélange des signaux qui a été utilisé était linéaire et instantané, ce qui nous a permis de simplifier la complexité de la matrice mélange et de généraliser le traitement des applications de la séparation de signaux. Par une approche neuronale, nous avons exploité le traitement des informations mutuelles statistiques des signaux de mélange à la sortie, par un algorithme d'apprentissage par l'ACI. Nous avons à cet effet la méthode du Gradient Naturel qui nous a permit de minimiser l'information mutuelle entre les signaux de sorties. La maximisation de l'information entre l'entrée et la sortie, illustre le calcul d'une fonction non linéaire d'apprentissage simple et moins complexe selon l'approche du Gradient Naturel lorsque la fonction sigmoïde est utilisée comme fonction d'activation du neurone.

Chapitre 2 :

Etat de l’art de la méthodologie de conception des circuits matériels

2.1 Introduction

L'étude des systèmes d'implémentation hardwares a pour objectif principal la fabrication des circuits intégrés en silicium pour des applications de traitement performantes. En effet, grâce à l'avancée technologique des circuits VLSI, les concepteurs cherchent à établir des méthodologies de conception pour étudier la réalisation des systèmes matériels. Pour cela, ils ont cherché à implémenter des algorithmes complexes des circuits intégrés digitaux pour réaliser des calculs parallèles et accélérer l'exécution des applications dans le circuit matériel. Ainsi dans l'analyse en temps réel des applications de traitement numérique des signaux, les algorithmes d'apprentissage des réseaux de neurones artificiel [2] ainsi que des fonctions non linéaires ne peuvent s'exécuter que par un traitement dans un circuit matériel grâce a des calculs parallèles intensivement multipliés.

Par ailleurs, des études d'optimisation ont été illustrées par la présence des méthodologies de conception des circuits logiques d'implémentation effectués par des algorithmes complexes sur les circuits matériels [29]. Les circuits logiques d'implémentation sont réalisés à base de portes logiques afin de cibler l'utilisation de la technologie VLSI pour la conception des circuits ASICs et FPGAs (voir Annexe A). En effet, la conception des circuits matériels est illustrée par le flot de cycles de conception de l'implémentation hardware dont les détails seront illustrés dans les chapitres suivants.

L'outil principal de traitement de la conception de l'implémentation hardware réside sur l'utilisation du langage de programmation VHDL qui est un outil de description des systèmes matériels, associé à des outils de logiciels CAO pour l'obtention de prototypes de circuits matériels à moindre coût de fabrication.

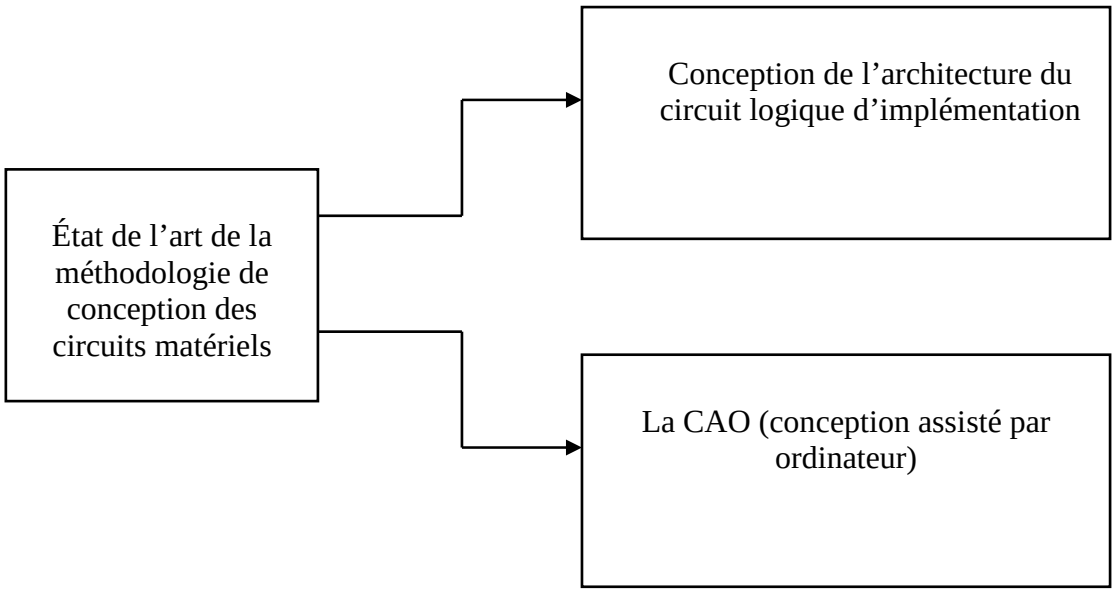


Figure 2.1 : Schéma de la méthodologie de conception des circuits matériels

L'étude générale de la conception des implémentations des circuits matériels est constituée par deux grandes étapes de conception principales. Une étape pour la conception de l'architecture du circuit logique d'implémentation, et une étape effectue l'étude par la CAO qui permet la simulation et la synthèse du circuit logique pour le prototypage rapide du circuit matériel (fig. 2.1).

2.2 Organigramme du cycle de conception des circuits d'implémentation hardware

L'organigramme du cycle d'implémentation hardware présente trois étapes de conception principales [17] (fig 2.2) : l'étape des calculs d'implémentation de l'algorithme, mise en œuvre de l'adéquation algorithme et architecture, conception de l'architecture d'implémentation sur le circuit matériel.

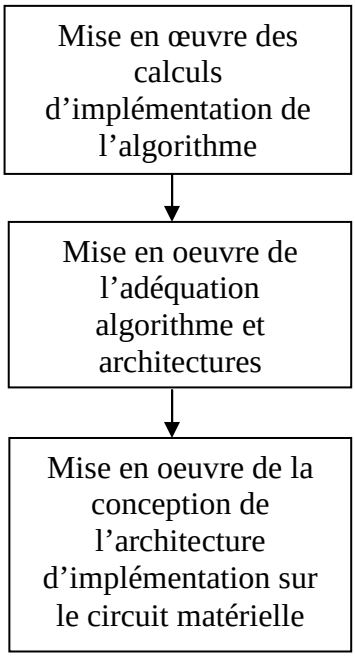


Figure 2.2 : Organigramme du cycle des implémentations hardware

La conception de l'implémentation matérielle peut être réalisée en ciblant plusieurs technologies des circuits intégrés digitaux (fig. 2.3).

.

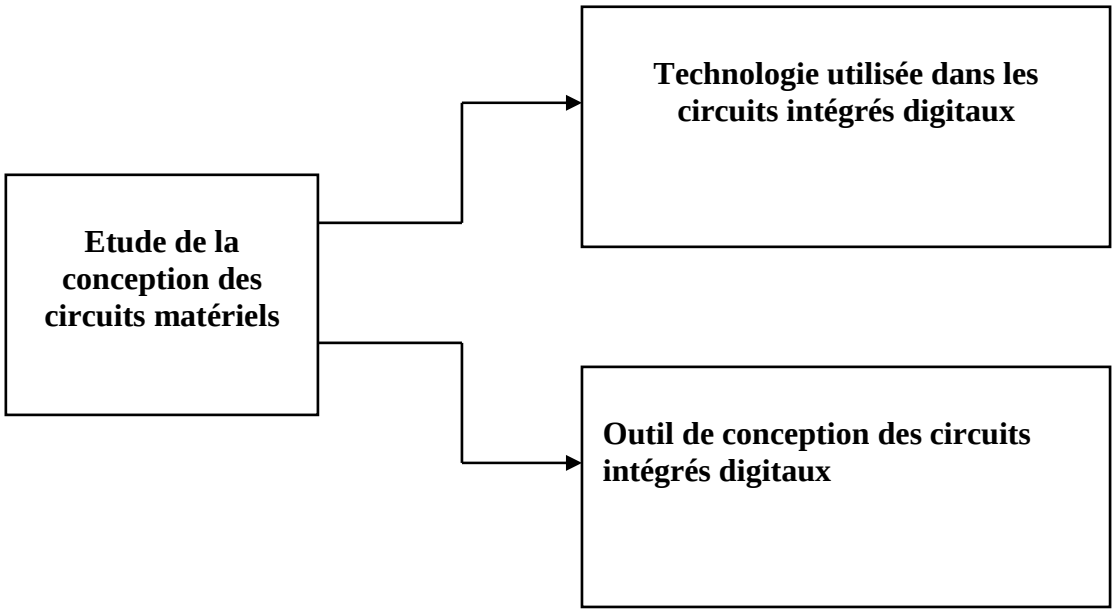


Figure 2.3 : Schéma d'étude de la conception des circuits matériels

2.3 Technologie utilisé dans les circuits intégrés digitaux

Le développement de la technologie VLSI des circuits intégrés digitaux a permis la réalisation des circuits de haute intégration pour des applications de traitement numérique des signaux. Avant quelques décennies, les applications qui réalisent le traitement d'algorithmes complexes nécessitent la réalisation des circuits digitaux avec de composants volumineux. Cependant, grâce à la miniaturisation dans la technologie des circuits en silicium, la densité des circuits intégrés utilisés a atteint des millions d'éléments de portes logiques qui ont permet de réaliser des circuits intégrés spécifiques à une application appelés ASICs (voir Annexe A) avec de grandes performances en mettant en œuvre des coûts de production et de consommation très réduits. Toujours, dans l'optique de réaliser des circuits intégrés numériques avec un faible coût de développement et de réalisation, les concepteurs des circuits intégrés digitaux ont exploité le développement de la technologie des circuits programmable FPGA (voir Annexe A) afin de construire des prototypes matériels numériques avec un temps minimal de conception.

L'implémentation hardware des calculs d'algorithmes de traitement numérique du signal dans un circuit DSP permet de conduire à des traitements de calcul séquentiels. Cependant, les circuits DSP exécutent des calculs moins complexes employant des calculs simples et répétitifs. En effet, le traitement de ces opérations de calcul est réalisé par des architectures de type Harvard présenté dans les architectures des circuits DSP. Ainsi, la conception de l'implémentation hardware des applications de traitement numérique de signaux dans un circuit DSP s'exécute par des opérations composées d'instructions séquentielles à l'aide d'un langage de programmation évolué classique (Figure 2.5).

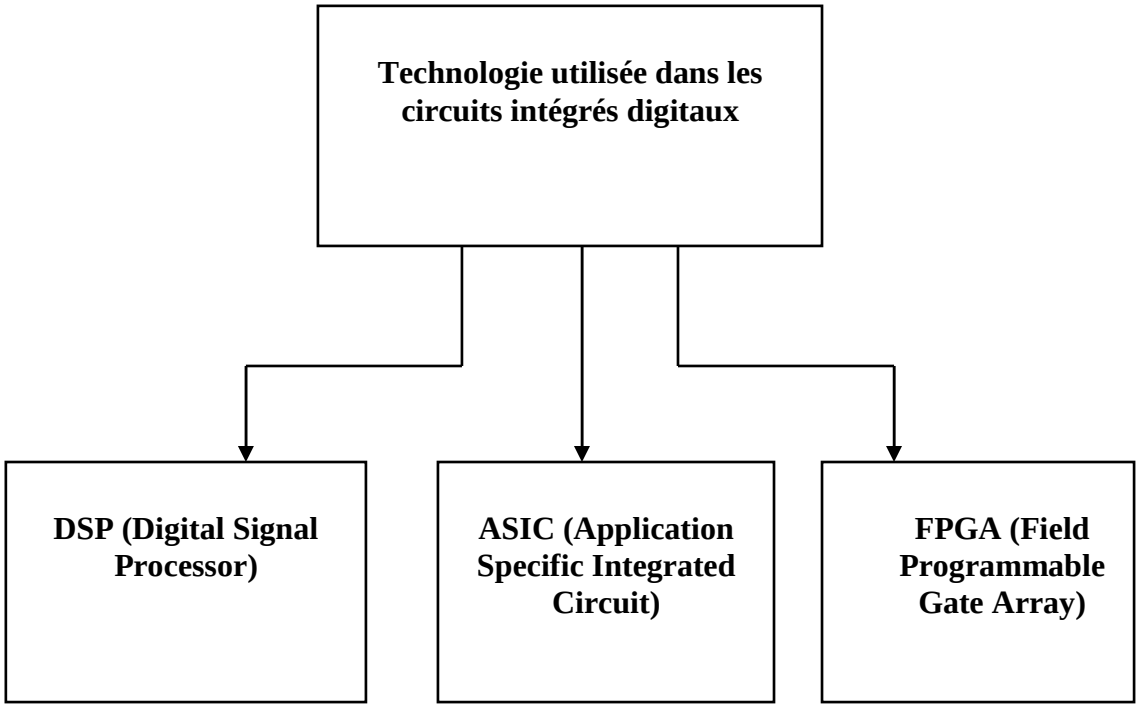


Figure 2.5 : Schéma des différentes technologies utilisées dans la conception des circuits intégrés digitaux

2.4 Outils de conception des circuits intégrés digitaux

Dans le domaine de la conception des circuits digitaux, le langage VHDL permet la description des systèmes matériels et la modélisation des circuits intégrés numériques. Actuellement, le VHDL associé à un ensemble d'outils logiciels de CAO, constitue une entrée pour la conception des circuits matériels complexes intégrés dans plusieurs plateformes de développement, en permettant de réduire le temps de développement. Ainsi, il permet la simulation et la synthèse du code VHDL qui décrit une multitude de technologie cible dans une plate forme de developpement en utilisant des outils logiciels spécifiques. Il permet aussi de synthétiser le code VHDL et de décrire l'architecture d'implémentation indépendante des technologies matérielles ciblées. A cet effet, le code VHDL peut être archivé pour une modification ou une réutilisation ultérieure. Ainsi la connaissance du langage VHDL permet de développer l'habileté du concepteur à communiquer des idées avantageuse pour la conception des circuits digitaux. Le schéma de la Figure 2.6 illustre la présentation de l'outil de conception des circuits intégrés digitaux.

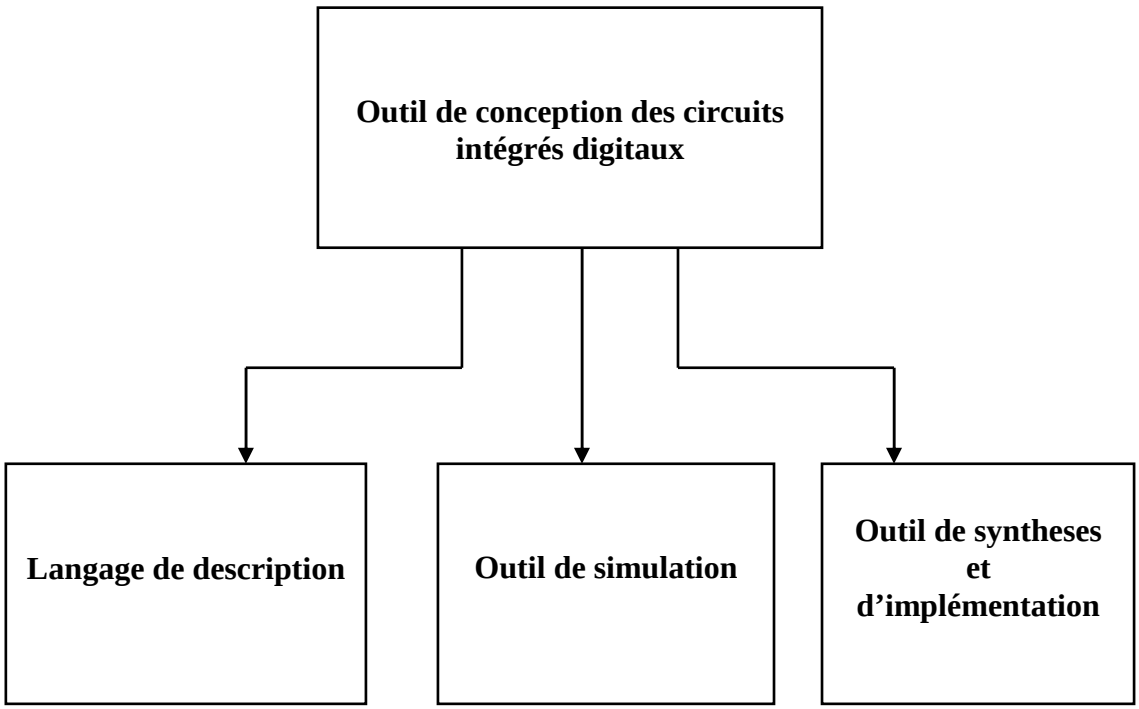


Figure 2.6. Schéma de l'outil de conception des circuits intégrés digitaux

2.5 Conception des circuits digitaux

Actuellement, le développement de la technologie VLSI des circuits intégrés numériques permet de conduire à des recherches scientifiques intenses afin de concevoir des circuits complexes en vue d'implémenter des algorithmes d'apprentissage des réseaux de neurones [2]. En effet, il a été exploité dans le traitement des circuits digitaux, des calculs parallèles fournis par les réseaux de neurone pour accélérer l'exécution de calculs de l'algorithme. Dans le traitement des circuits digitaux, nous avons utilisé des calculs arithmétiques de somme et de produit pour les calculs de l'entrée synaptique et de la fonction d'activation du neurone, qui sont utiles pour l'application de traitement de signaux numériques en temps réel.

Une conception des circuits d'implémentation hardware complexe a été mise en œuvre pour atténuer les coûts de fabrication et de développement. Par conséquent, le traitement du cycle de l'implémentation hardware (Figure 2.3) a été utilisé comme un traitement principal pour adopter la conception des circuits intégrés digitaux. A cet effet, une méthodologie de conception a été appliquée à chaque étape du cycle d'implémentation hardware afin d'optimiser les calculs de l'algorithme conduisant à des architectures d'implémentation hardware avec des surface réduites.

Actuellement, les concepteurs effectuent des implémentations hardware des algorithmes d'apprentissages des réseaux de neurone dans les circuits FPGA pour construire des prototypes rapides des circuits matériels en atténuant les coûts de conception. Mais l'inconvénient majeur de ces circuits pose le problème de la limitation des ressources matérielles. Pour cela, les concepteurs ont employé des techniques de calculs optimisés afin d'atténuer la complexité du traitements aboutissant a des architectures d'implémentation hardware ayant des surfaces réduites.

Dans cette optique, des travaux ont été réalisés pour la conception des implémentations hardware des circuits FPGA pour construire des prototypes de circuits matériels digitaux à moindre coût de fabrication [6]-[8]. Pendant la conception de l'implémentation hardware des calculs d'optimisation ont été utilisés pour le traitement d'algorithme d'apprentissage des réseaux de neurones afin d'atténuer la complexité de la conception [15]-[16]. Ainsi ces calculs d'optimisation permettent de construire des architectures d'implémentation hardware simple pour éviter le problème de la limitation des ressources dans les circuits FPGA en favorisant l'accélération du temps d'exécution des circuits matériels [12]-[14] pour des fins de traitement en temps réel.

2.6 Conclusion

Des travaux de recherche ont été menés dans la conception des implémentations hardware des réseaux de neurones dans les thèmes de la recherche de conception des architectures d'implémentation hardware de séparation de signaux. L'utilisation du langage de programmation VHDL qui a pour objectif, la description des systèmes matériels d'architectures complexes, permettra de réduire fortement les coûts de développement et de conception. Ainsi l'utilisation de ce langage permettra d'obtenir des prototypes rapides en utilisant des circuits programmables FPGA dans la conception de l'implémentation hardware.

Chapitre 3 :

Mise en œuvre des calculs d'implémentation de l'algorithme (ACI)

3.1 Introduction

Dans ce chapitre, un développement théorique du traitement de calculs de l'algorithme a été mis en œuvre afin de fournir une simple conception des circuits d'implémentation digitaux. Grâce à une méthodologie de conception optimisée, des circuits digitaux d'implémentation moins complexes seront mis en œuvre pour réaliser une architecture d'implémentation en adéquation avec le calcul théorique développé. A cet effet, un organigramme a été proposé dans la Figure 3.1 pour illustrer le traitement global de l'algorithme d'apprentissage des réseaux de neurones. L'organigramme qui illustre l'algorithme de l'ACI effectue les calculs de l'ajustement des poids synaptiques finaux des réseaux de neurones pour la séparation des signaux. Ses performances dépendent fortement de la convergence de l'ajustement des poids synaptiques.

En outre, on peut effectuer des calculs d'optimisation pour établir des calculs d'approximation sur des fonctions non linéaires complexes. Ces traitements de calculs sont basés sur les calculs des segments de droites linéaires permettant la réalisation des architectures des circuits digitaux d'implémentation moins complexes.

L'organigramme de la Figure 3.1 effectue les différentes opérations de l'algorithme [17]-[21] et illustre l'optimisation des calculs. En effet, cet organigramme met en œuvre principalement trois étapes: calcul de l'ajustement des poids synaptiques, celui de l'accumulation des poids synaptiques ajustés, et celui calcul de l'ajustement des poids synaptiques finaux.

3.2 Calcul de l'ajustement des poids synaptiques

Dans cette étape, le calcul de l'ajustement des poids synaptiques W_{ij}^{+}, s'effectue à travers le traitement des calculs dans les boucles d'itérations (ite), jusqu'à la satisfaction de la convergence des poids synaptiques. "ite" désigne le nombre d'itérations. La règle d'arrêt de l'ajustement des poids synaptiques est illustrée dans la Figure 3.1 par le test du nombre ε qui utilise des calculs de l'entrée synaptique et de la fonction non linéaire d'apprentissage des réseaux de neurones, selon l'expression de l'équation :

$$\varepsilon = \sum_{i=1}^{n} \varphi(u_i) u_i \tag{26}$$

avec ; $\varepsilon < \varepsilon_{max}$, et ε_{max} = Cte. Où ε illustre l'information sur les calculs de l'ajustement de la convergence des poids synaptiques. D'où, le calcul de l'ajustement des poids synaptiques, pour (i=1,2,…,n) et (j=1,2,…,n), devient :

$$W_{ite\,ij} = W_{(ite-1)ij} + \mu(1+\varepsilon) W_{(ite-1)ij} \tag{27}$$

L'équation de l'ajustement des poids synaptiques à la convergence devient :

$$W_{c_k ij} = W_{(c_k-1)ij} + \mu(1+\varepsilon)W_{(c_k-1)ij} \tag{28}$$

$(1 \leq \text{ite} \leq c_k)$ et $(1 \leq k \leq M)$ M désigne le nombre maximum d'échantillons de signaux d'entrées, c_k indique le nombre d'itérations effectuées pendant l'étape de l'ajustement des poids synaptiques à priori à la convergence pour chaque nombre de l'échantillon k à l'entrée.

La normalisation des valeurs de l'ajustement des poids synaptiques nous permet de fixer le module de sa plus grande valeur égale à l'unité. D'où, après l'étape de normalisation, l'on a:

$$W_{c_k ij} = \frac{W_{(c_k-1)ij} + \mu(1+\varepsilon)W_{(c_k-1)ij}}{W_{c_k ij\,\max}} \tag{29}$$

où : $W_{(c_k-1)ij}$ est la valeur de l'ajustement des poids synaptiques de l'itération précédente ; $W_{c_k ij\,\max}$ est la valeur maximale des calculs de l'ajustement des poids synaptiques avant le traitement de normalisation de l'itération présente.

3.3 Calcul de l'accumulation des poids synaptiques ajustés

Dans cette étape, le traitement des calculs de l'accumulation des poids synaptiques ajustés $W_{a_k ij}$ s'effectue à travers les boucles d'itérations de l'échantillon k à l'entrée (voir Figure 3.1) jusqu'à la valeur maximum (k=M). L'ajustement de l'accumulation des poids synaptiques pendant l'itération de l'échantillon k se calcule selon :

$$W_{a_k ij} = W_{a_{(k-1)}ij} + W_{c_k ij} \tag{30}$$

où ; $W_{a_{(k-1)}ij}$ est la valeur de l'ajustement de l'accumulation des poids synaptiques pendant l'itération de l'échantillon k précédent ; $W_{c_k ij}$ est la valeur des poids synaptiques ajustés à la convergence de l'échantillon k présent. Le calcul de l'accumulation finale des poids synaptiques ajustés W_{cij} est déterminé par l'équation suivante :

$$W_{cij} = \sum_{k=1}^{M} W_{c_k ij} \tag{31}$$

3.4 Calcul de l'ajustement de la valeur des poids synaptiques finaux

Le calcul de l'ajustement des poids synaptiques finaux Wf_{ij} s'effectue par le calcul de la moyenne des poids synaptiques accumulés de tous les échantillons selon:

$$Wf_{ij} = \frac{W_{cij}}{M} \tag{32}$$

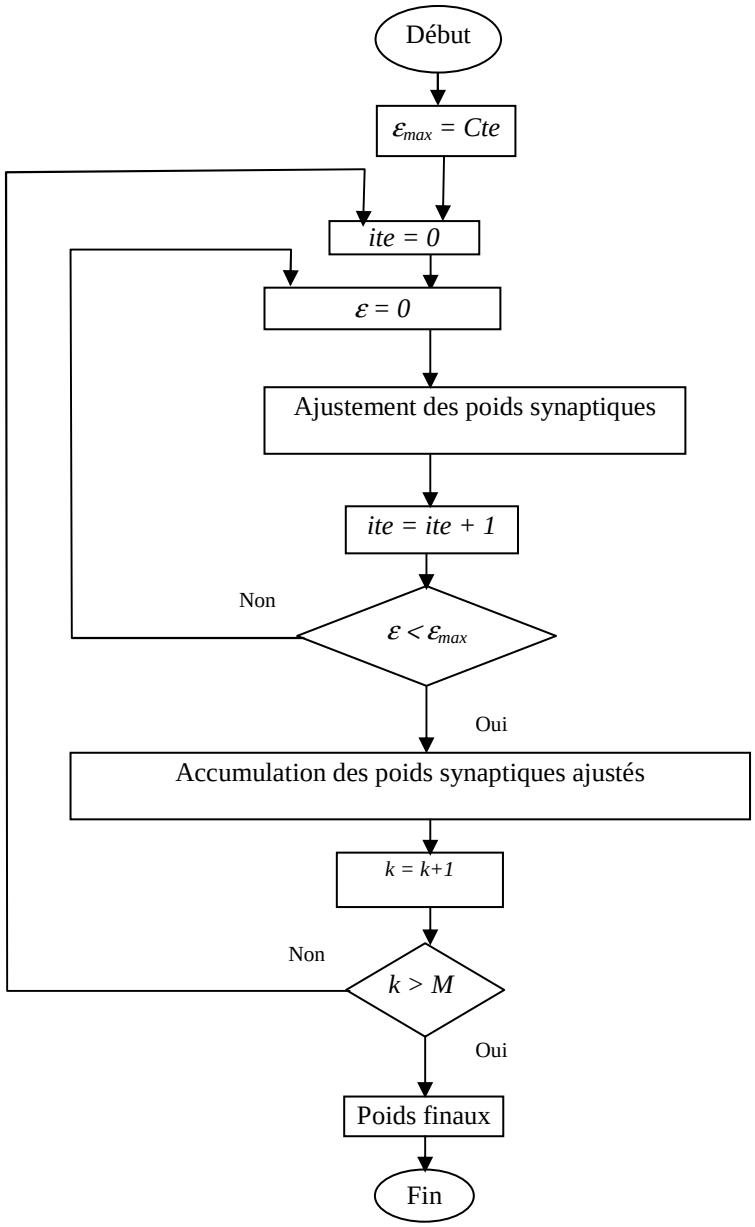


Figure 3.1 : Organigramme des calculs d'implémentation de l'algorithme

3.5 Implémentation de la fonction sigmoïde

Nous effectuons les calculs d'implémentation de la fonction sigmoïde $f(u_i)$ qui permet d'illustrer une approximation par des fonctions linéaires simples (piecewise linear approximation). Ces fonctions d'approximation ont pour effet de réduire la complexité dans les calculs tout en minimisant au maximum l'erreur lors des calculs d'implémentation de la fonction sigmoïde.

La fonction segmentation [15] $f(u_i)$ que nous avons utilisé présente un minimum d'erreur selon les équations suivantes:

$$
\begin{cases}
f(u_i) = 1 - f(u_i), & \text{pour : } u_i < 0, \\
f(u_i) = 0.25 * / u_i / + 0.5, & \text{pour : } 0 =< / u_i / < 1, \\
f(u_i) = 0.125 * / u_i / + 0.625, & \text{pour : } 1 =< / u_i / < 2.375, \\
f(u_i) = 0.03125 * / u_i / + 0.84375, & \text{pour : } 2.375 =< / u_i / < 5, \\
f(u_i) = 1, & \text{pour : } / u_i / >= 5,
\end{cases}
$$

3.6 Conclusion

Pendant le traitement des calculs d'implémentation, nous avons noté que les performances qui illustrent la convergence des poids synaptiques varient en fonction des paramètres de l'organigramme de la Figure 3.1. Ainsi, les valeurs de ces paramètres qui permettent l'ajustement des poids synaptiques finaux par l'ACI, déterminent les performances de la séparation de signaux. En outre, l'obtention d'un compromis entre les valeurs de l'ajustement des poids synaptiques et ceux des circuits digitaux d'implémentation est effectuée en mettant en œuvre l'ajustement du paramètre ε. Ces performances ont été obtenues en fonction du nombre d'itérations utilisées pour la convergence des poids synaptiques.
Ces performances peuvent être évaluées en fonction de l'optimisation de la surface du circuit et par le nombre de cycles utilisés dans les opérations de traitement de l'algorithme afin d'effectuer le calcul d'estimation des signaux indépendants.

Chapitre 4 :

Mise en œuvre de l'adéquation Algorithme / Architecture

4.1 Introduction

Dans cette section, nous allons illustrer l'étape de conception de l'architecture du circuit d'implémentation de l'algorithme ACI. En effet, nous avons utilisé les calculs par l'approche de l'information mutuelle pour la séparation des signaux [17]-[20], qui met en œuvre le traitement par l'ACI afin de réaliser une architecture d'implémentation de taille réduite. Cette approche de calcul illustre le calcul de la fonction non linéaire d'apprentissages des réseaux de neurones en effectuant un calcul arithmétique moins complexe.

La méthodologie de conception proposée doit conduire à la réalisation de l'Adéquation Algorithme et Architecture (AAA) pour optimiser la surface du circuit logique d'implémentation de l'algorithme. En outre, il est possible de commander les calculs de l'algorithme par plusieurs variantes de circuit logique d'implémentation selon des traitements séquentiels ou parallèles. Le traitement parallèle est réalisé par une adéquation algorithme/ architecture d'un circuit logique de taille complexe alors que le traitement séquentiel est réalisé par une adéquation algorithme/architecture taille moins complexe et optimisée afin de traiter un seul neurone.

Néanmoins, les traitements séquentiels engendrent un temps d'exécution important pour le traitement d'un grand nombre de neurones en temps réel. En outre, les calculs parallèles exigent une architecture de taille importante et dont la complexité augmente de façon considérable en fonction du nombre d'échantillons des signaux d'entrées. Ce type de traitement peut accélérer fortement le temps d'exécution de l'algorithme. Ceci pose le problème de compromis entre la rapidité et la complexité de l'architecture d'implémentation qui rend la disponibilité des ressources matérielles insuffisantes pour implémenter une telle architecture.

L'architecture d'implémentation est composée de deux principaux blocs: le bloc de traitement qui illustre les calculs de l'algorithme et le bloc de contrôle qui commande les différentes opérations de traitement de l'algorithme. La commande s'exécute de façon parallèle ou séquentielle de façon synchronisée par l'intermédiaire d'un signal horloge.

En outre, l'organigramme de la figure 3.1 montre toutes les étapes du déroulement séquentiel des opérations arithmétiques et logiques.

4.2 Schéma synoptique d'implémentation du neurone

L'organigramme de la figure 3.1 met en œuvre trois sous blocs pour l'architecture d'implémentation du bloc opérationnel: le bloc de calcul de la fonction non linéaire d'apprentissage, le bloc de calcul de l'ajustement des poids synaptiques avant la convergence, et le bloc d'ajustement des poids synaptiques finaux. Le schéma synoptique de la figure 4.1 présente une topologie d'implémentation d'un seul neurone pour le traitement de l'algorithme ACI. Cette figure montre également les interconnexions pour l'implémentation du neurone.

Ces lignes lient les blocs d'implémentation du neurone par des signaux d'entrée et des signaux intermédiaires qui sont utilisés pour le bloc de contrôle et le bloc de traitement. Le

bloc des mémoires RAM (utilisé pour le stockage des données à l'entrée et des données l'ajustement des poids synaptiques) est relié par des signaux bidirectionnels entre le bloc de l'ajustement des poids synaptiques et le bloc d'ajustement des poids synaptiques finaux.

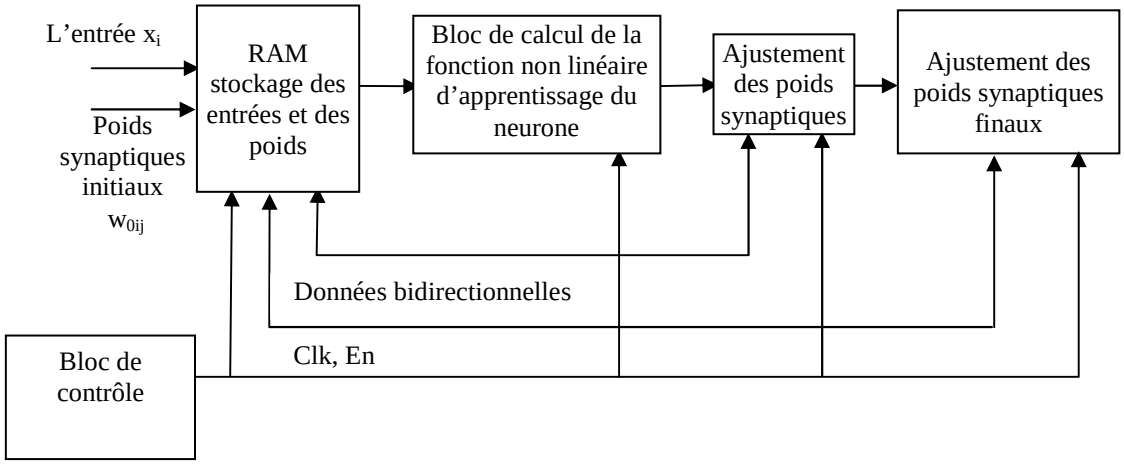


Figure 4.1 : Schéma synoptique d'implémentation du neurone

Le bloc de contrôle commande les calculs de l'algorithme grâce à des interconnexions qui illustrent les signaux de validation et d'horloge pour synchroniser les opérations du bloc de traitement.

4.3 Architecture détaillée du bloc synoptique d'implémentation du neurone

Une architecture a été construite pour le traitement d'implémentation de chaque neurone. Cependant, dans le schéma synoptique montré par la figure 4.2, les trois blocs d'implémentation sont reliés à un bloc mémoire RAM qui illustre la connexion détaillée des mémoires pour le stockage des données. En effet, la RAM1 stocke les échantillons des signaux d'entrées, la RAM2 stocke les données d'ajustement des poids synaptiques ainsi que la valeur finale des poids synaptiques finaux après convergence. La RAM3 est utilisée pour le stockage des données d'accumulation des poids synaptiques. En outre, le signal d'horloge et les signaux de validation effectuent la commande du bloc de traitement en deux phases de traitement pipeline respectivement pour le bloc de la fonction non linéaire d'apprentissage du neurone et le bloc d'ajustement des poids synaptiques. Ce type de traitement permet d'optimiser le nombre de cycles horloge.

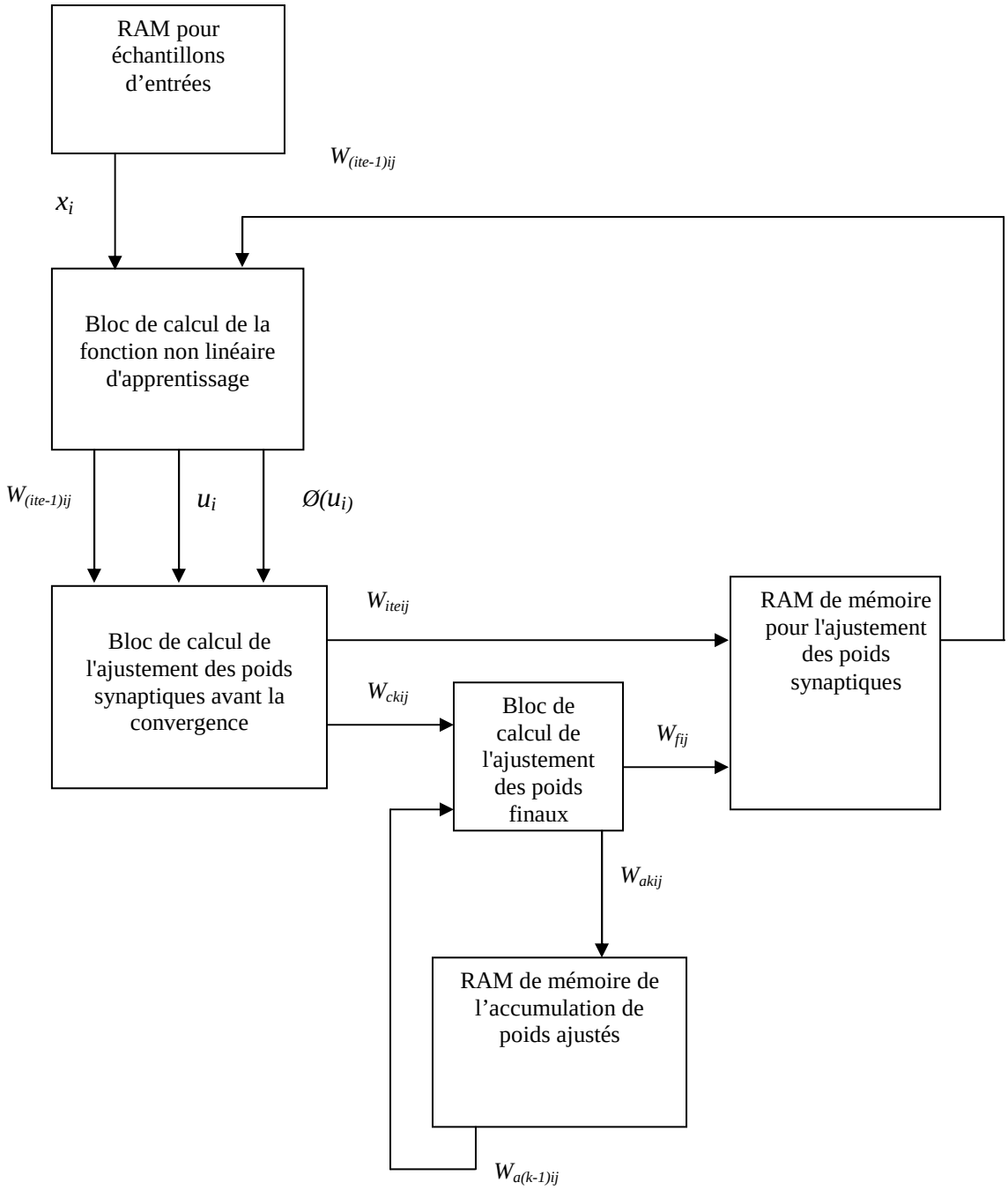


Figure 4.2. Architecture détaillée du bloc synoptique d'implémentation du bloc de traitement du neurone

4.3.1. Implémentation du bloc de calcul de la fonction non linéaire d'apprentissage

L'architecture du bloc de calcul de la fonction non linéaire d'apprentissage du réseau de neurone contient deux sous blocs: un sous bloc de calcul de l'entrée synaptique, et un autre bloc pour le calcul de la fonction non linéaire d'apprentissage.

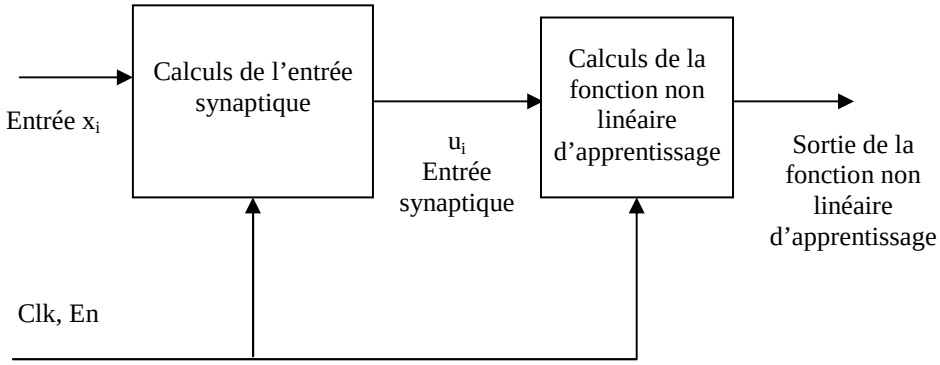


Figure.4.3 : Architecture du circuit logique d'implémentation du bloc de calculs de la fonction non linéaire d'apprentissage

4.3.1.1 Architecture d'implémentation du calcul de l'entrée synaptique

Le schéma d'implémentation du calcul de l'entrée synaptique est illustré par la Figure 4.4. Dans cette architecture, nous utilisons trois éléments de traitement arithmétique pour le calcul de l'entée synaptique. A cet effet, nous utilisons un multiplieur, un additionneur et un accumulateur.

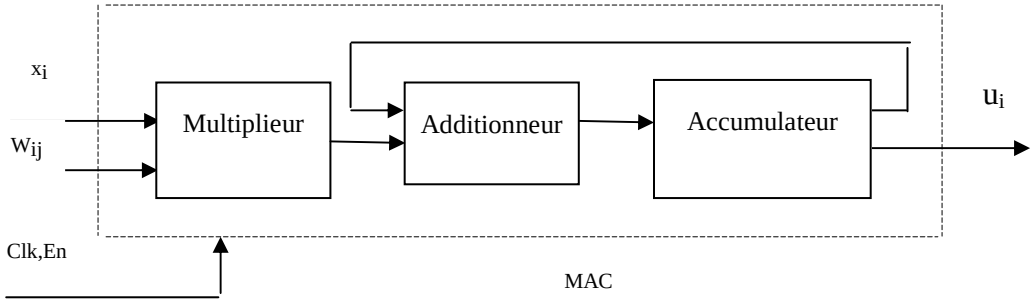


Figure.4.4. implémentation du calcul de l'entrée synaptique

4.3.1.2 Implémentation du bloc de calcul de la fonction non linéaire d'apprentissage

La figure 4.5 illustre le schéma d'implémentation de la fonction non linéaire d'apprentissage basée sur l'approximation de la fonction sigmoïde à l'aide d'un additionneur et d'un circuit logique de calcul du complément à deux et du décalage à droite.

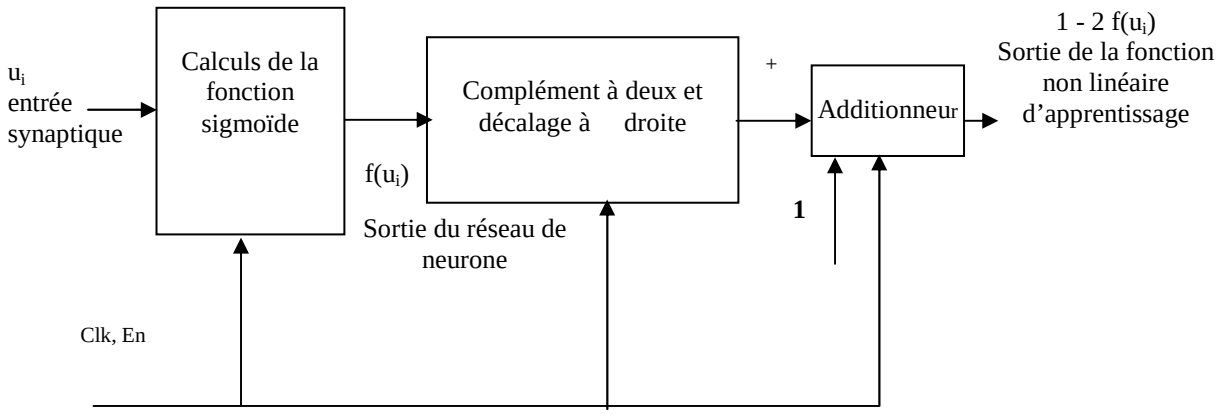


Figure.4.5 : Architecture du circuit logique d'implémentation du bloc de calculs de fonction non linéaire d'apprentissage

➢ Architecture du circuit logique d'implémentation de la fonction sigmoïde

Le calcul de la fonction sigmoïde nécessite un traitement complexe pour obtenir des calculs exacts. Ainsi, nous allons réaliser le calcul d'optimisation de la fonction sigmoïde pour obtenir des calculs moins complexe. Cependant, plusieurs techniques de calculs ont été illustrées dans la littérature [15]-[16] pour définir des fonctions du traitement de calculs qui optimise la fonction sigmoïde.

Parmi les techniques de traitement les plus connues, nous avons retenu la méthode de segmentation de la fonction sigmoïde [15]. Les performances de cette technique ont été évaluées par comparaison à d'autres techniques d'approximation [16]. Ainsi, cette fonction d'approximation réalise une architecture très optimisée.

Le tableau 4.1 illustre les équations de droites qui sont associées aux intervalles d'interpolation de la fonction sigmoïde. Ainsi, le schéma d'implémentation de calcul de la fonction sigmoïde optimisée est donné à la Figure 4.6. Les valeurs des dérivées des segments de droite associées à la fonction d'optimisation sont déterminées pour qu'ils aient des valeurs de puissance de deux. En effet, ces valeurs mettent en œuvre une implémentation du circuit logique par des registres à décalage pour éviter l'implémentation des circuits multiplieurs.

Dans le circuit logique d'implémentation de la fonction segmentation, la mise à 1 de l'un des drapeaux Z1, Z2, et Z3 illustre la sélection de l'équation de droite de la fonction sigmoïde optimisée associées aux différentes intervalles d'interpolation des valeurs du signal d'entrée X. Ainsi, l'implémentation du circuit logique de la fonction sigmoïde, comporte une implémentation: des calculs de complément à 2, de deux multiplexeurs, d'un additionneur, d'un comparateur d'amplitude, et de la transformation directe (x,y).

Le comparateur d'amplitude indique l'intervalle des données d'entrées à laquelle appartient le segment de droite de la fonction segmentation. Un circuit logique combinatoire [15] de la transformation directe (x,y) effectue les calculs de transfert entre les registres d'entrée et de sortie lesquelles mémorisent respectivement les données d'entrées et de sortie de la fonction sigmoïde optimisée permettant d'éviter l'implémentation des circuits mémoires utilisés dans l'implémentation des registres à décalage.

Tableau 4.1 : Implémentation de la fonction sigmoïde

Operation	Condition	Drapeaux Z1	Z2	Z3	Z4
Y1=1	/X/ >= 5	0	0	0	1
Y2=0.03125*/X/+0.84375	2.375 =</XI < 5	0	0	1	0
Y3=0. 125*/X/+0.625	1 =</X/<2.375	0	1	0	0
Y3=0. 25*/X/+0.5	0=</X/< 1	1	0	0	0
Y=1-Y	X< 0				

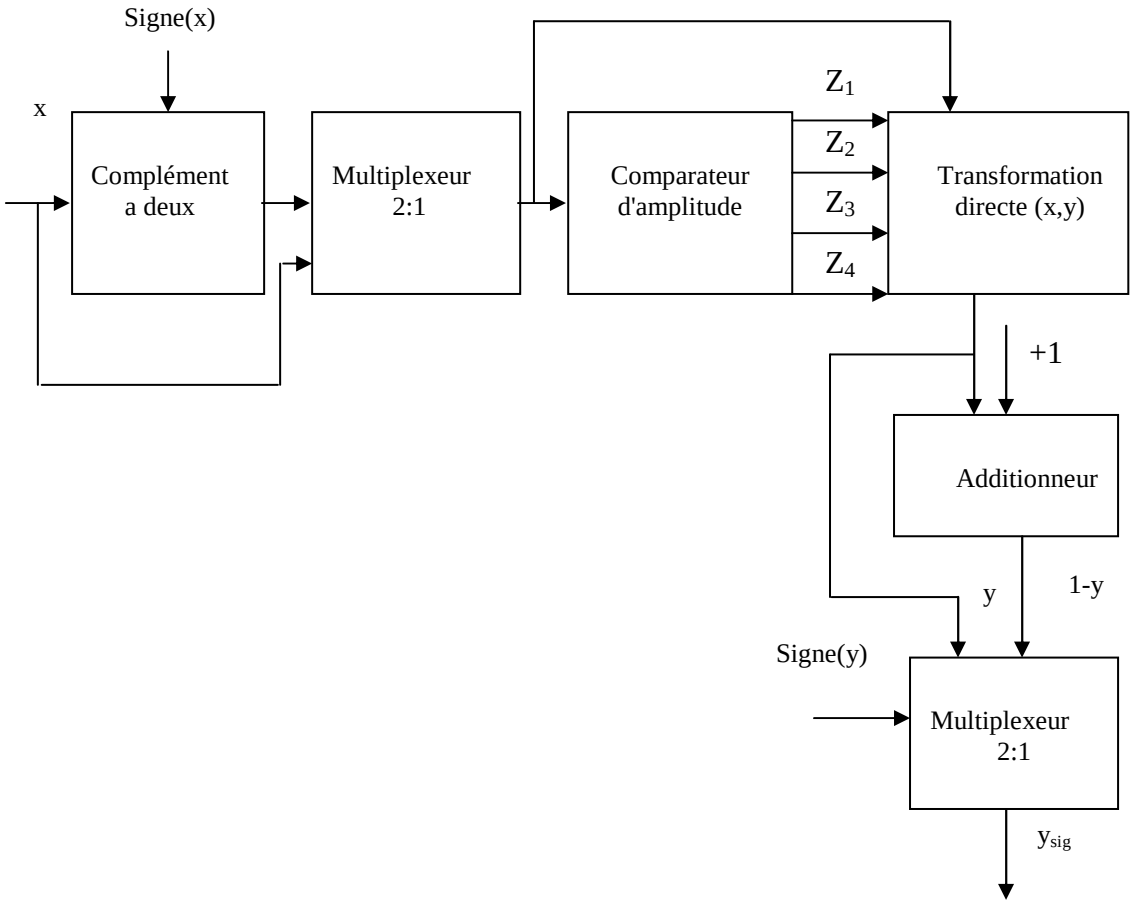


Figure.4.6 : Schéma du circuit logique d'implémentation de la fonction sigmoïde

4.3.2 Architecture d'implémentation des blocs de calculs de l'ajustement des poids synaptiques

La Figure 4.7 représente l'architecture de l'ajustement des poids synaptiques

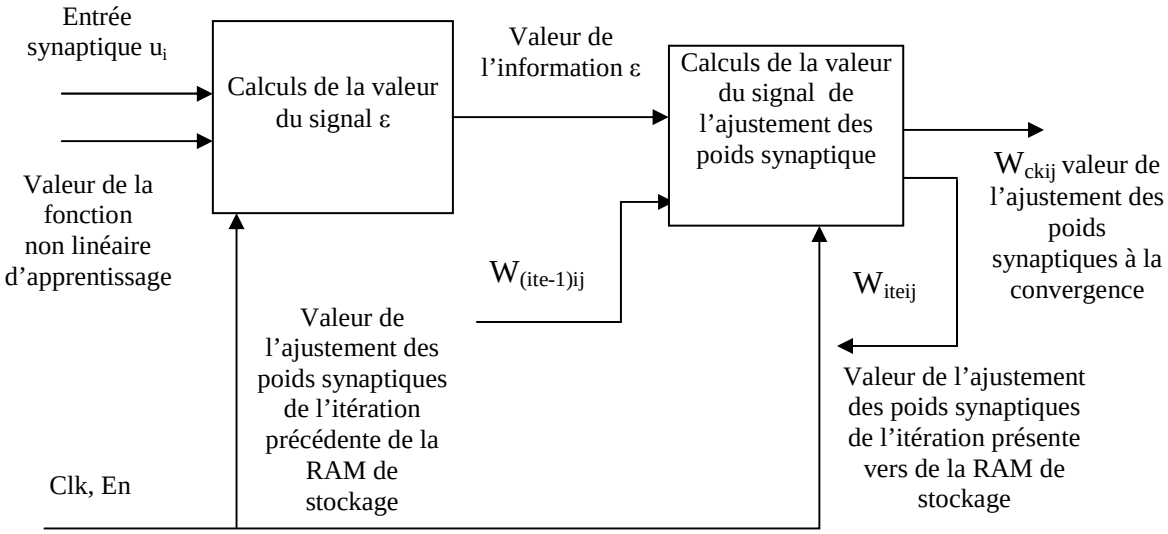


Figure 4.7 : Architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques

4.3.2.1 Architecture du circuit logique d'implémentation du calcul de la valeur du signal ε

La figure 4.8 représente le schéma d'implémentation du circuit logique pour le calcul de la valeur du signal ε.

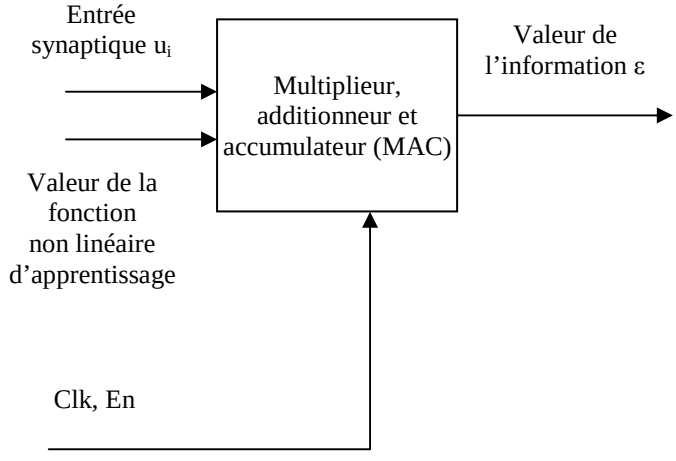


Figure 4.8 : Architecture du circuit logique d'implémentation du calcul de valeur du signal ε

4.3.2.2 Architecture du circuit logique d'implémentation du calcul de la valeur de l'ajustement des poids synaptiques

Dans le bloc du traitement des calculs de l'ajustement des poids synaptiques, nous avons utilisés un MAC (le même composant a été utilisé pour implémenter le traitement de calculs de l'entrée synaptique) qui constitue une architecture d'un circuit logique d'implémentation pour le traitement des calculs du signal d'information ε. Le composant MAC a été utilisé pour le calcul de la valeur de l'information qui indique la précision de la convergence de l'ajustement des poids synaptiques. En outre, nous avons utilisés, deux additionneurs pour le traitement du calcul arithmétique d'addition, un multiplieur pour le traitement du calcul arithmétique de multiplication, un registre de décalage vers la droite pour illustrer le traitement de la valeur du taux d'apprentissage μ, et un démultiplexeur (DEMUX) pour assurer l'aiguillage du signal de l'ajustement des poids synaptique à la convergence, vers le bloc de l'ajustement des poids synaptiques finaux. On a utilisé également une unité de normalisation pour le traitement de la normalisation des valeurs des poids synaptiques ajustés. La Figure 4.9, illustre le circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques.

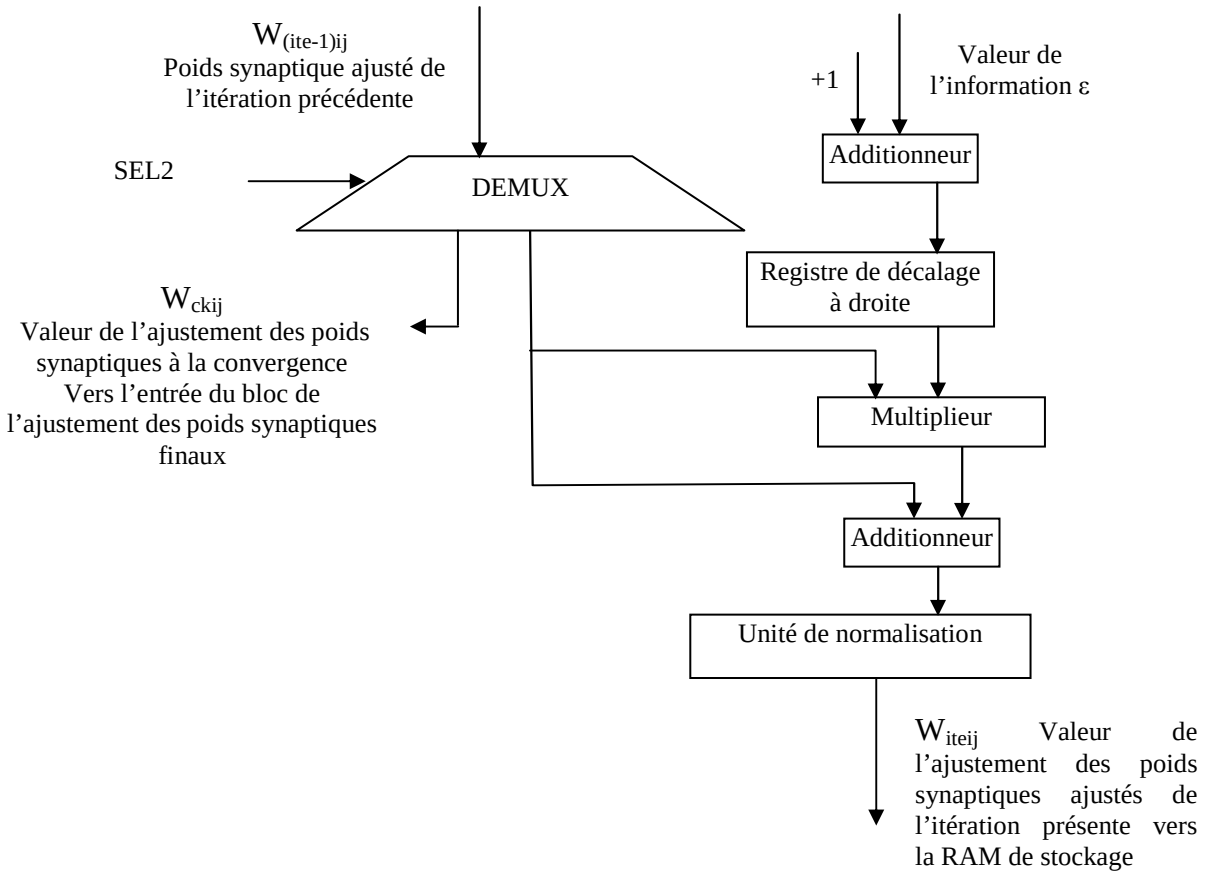


Figure 4.9 : Architecture digitale d'implémentation du bloc de l'ajustement des poids synaptiques

4.3.3 Architecture d'implémentation des blocs de calculs de l'ajustement des poids synaptiques finaux

Le bloc d'implémentation illustré par la figure 4.10, montre l'ajustement des poids synaptiques finaux. Il contient un bloc de calcul de l'accumulation des poids synaptiques ajustés et un bloc pour le calcul de l'ajustement des poids synaptiques finaux. La commande de l'ajustement des poids synaptiques finaux est réalisés par des signaux d'horloge CLK et des signaux de validations EN.

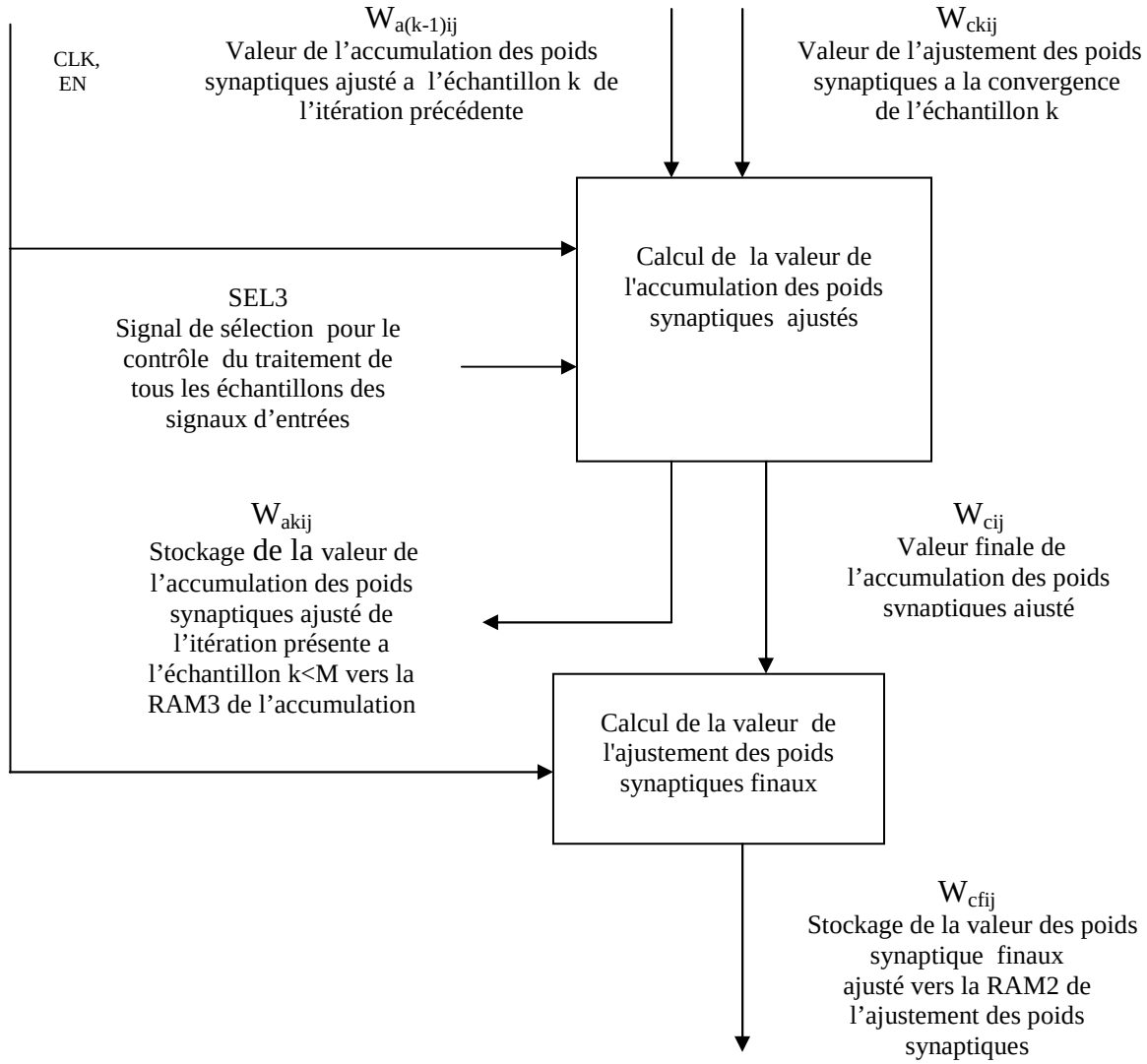


Figure 4.10 : Architecture du circuit logique
d'implémentation du poids synaptique final ajusté

4.3.3.1 Architecture du circuit logique d'implémentation de calculs de l'accumulation des poids synaptiques ajustés

Le bloc d'implémentation illustré par la Figure 4.11 montre l'architecture de l'accumulation des poids synaptiques ajustés. Dans ce schéma, nous avons utilisé une architecture constituée par un additionneur qui reçoit respectivement le signal $W_{ckij,}$ la valeur de l'ajustement des poids synaptiques à l'échantillon k, et le signal $W_{a(k-1)ij}$ (accumulation des poids synaptiques ajustés à l'échantillon k-1) pour calculer la valeur du signal W_{akij} de l'accumulation des poids synaptiques ajusté à l'échantillon k. Nous avons aussi utilisé un démultiplexeur pour le contrôle du traitement de tous les échantillons des signaux d'entrées. Ainsi, ce démultiplexeur reçoit le signal W_{akij} fourni par l'additionneur et le transmet par l'intermédiaire du signal de sélection SEL3 à W_{akij} pour son stockage dans la mémoire RAM des accumulations ou bien vers le signal W_{cij} pour être traité dans le bloc de calcul de l'ajustement des poids synaptiques finaux.

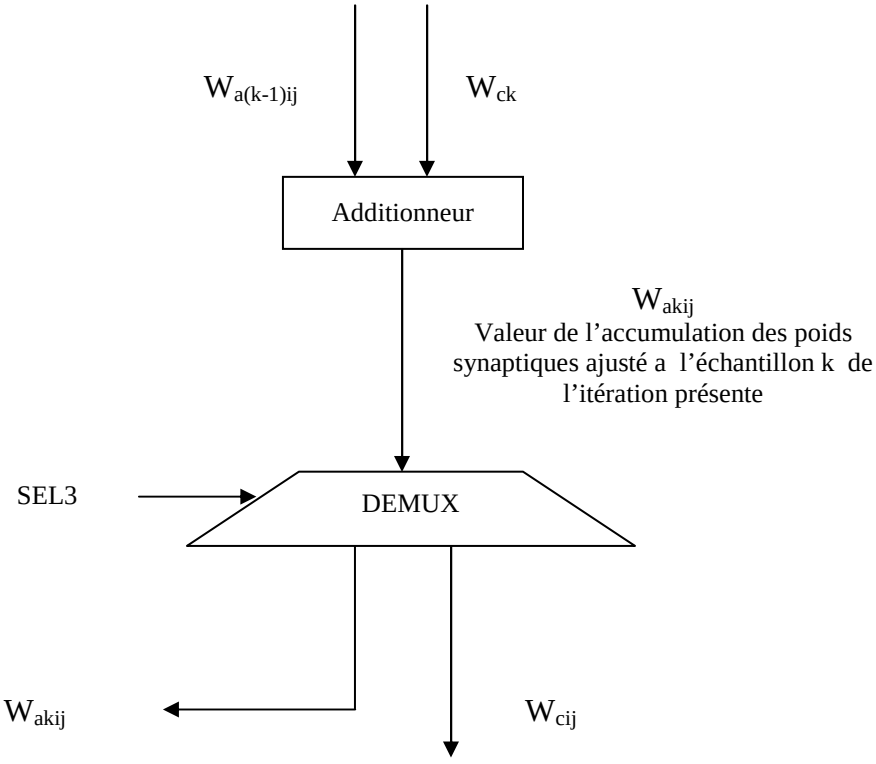


Figure 4.11 : Architecture du circuit logique d'implémentation de l'accumulation du poids synaptique ajusté

.3.3.1 Architecture d'implémentation des calculs de l'ajustement des poids synaptiques finaux

La figure 4.12 illustre le calcul de la valeur des poids synaptiques finaux en utilisant un registre à décalage qui permet d'effectuer les calculs de la valeur moyenne de l'accumulation des poids synaptiques ajustés W_{cij} sur tous les échantillons et les stocke dans la

mémoire RAM de l'ajustement des poids synaptiques. L'implémentation du registre à décalage a été utilisée pour optimiser la division par le nombre M qui désigne le nombre d'échantillons des signaux d'entrées, exprimée en puissance de deux.

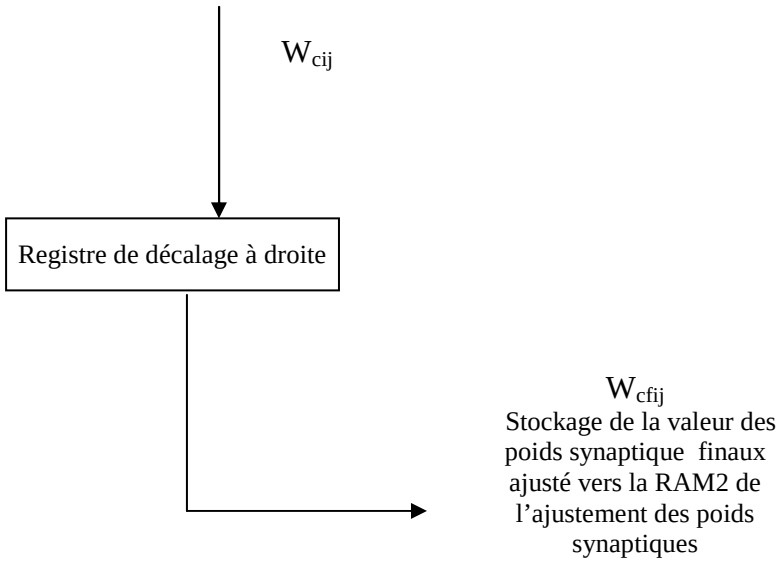


Figure 4.12 : Architecture du circuit logique d'implémentation du poids synaptique final ajusté

4.3.4 Architecture détaillée du circuit logique d'implémentation du bloc de traitement du neurone

La Figure 4.13 donne le schéma détaillé du bloc de traitement du neurone. Ainsi le bloc de traitement fournit un signal ε au bloc de contrôle lequel va commander le signal de sélection SEL2 dans le bloc de l'ajustement des poids synaptiques.

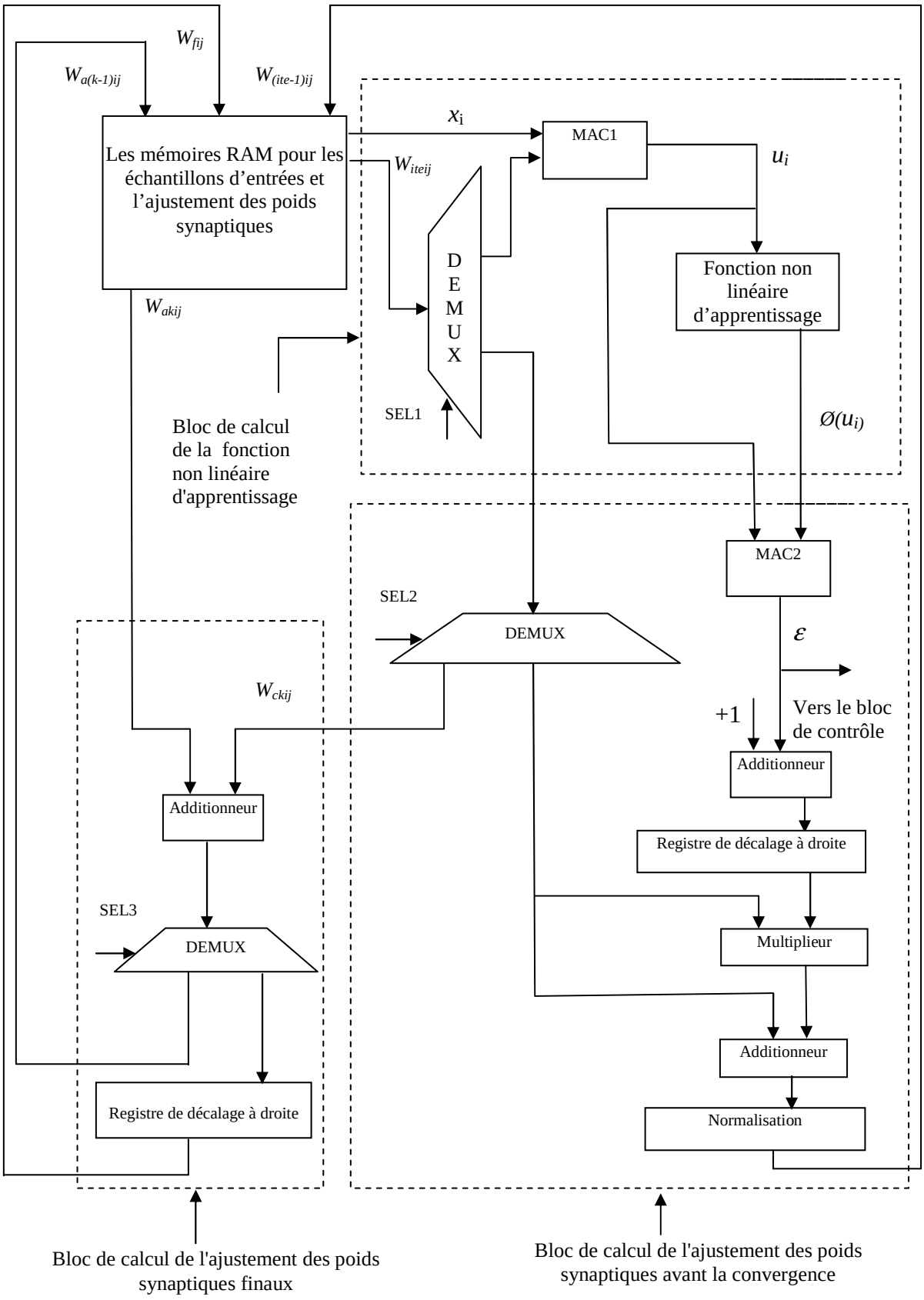


Figure 4.13 : Architecture d'implémentation du bloc de traitement d'un seul neurone

4.4 Architecture d'implémentation du bloc de contrôle

L'architecture d'implémentation du bloc de contrôle assure le contrôle des opérations de calculs de l'algorithme qui réalise le traitement des opérations parallèles ou séquentielles du circuit logique par l'intermédiaire des signaux de validation et de synchronisation. Ainsi, la conception du circuit logique d'implémentation du bloc de contrôle dépend de la complexité du circuit d'implémentation du bloc de traitement. En général, ces circuits sont réalisés par l'intermédiaire des compteurs et des comparateurs. Ainsi, la génération des signaux de contrôle sera réalisée par le traitement de ces derniers qui sont pilotés par des signaux d'horloges, de signaux de validation, et de signaux de RAZ asynchrones pour permettre le contrôle des différentes opérations.

Avant de réaliser le circuit d'implémentation, le bloc de contrôle est conçu sur la base d'une description comportementale des systèmes matériels par l'intermédiaire d'un langage de programmation approprié. Depuis longtemps, la conception du bloc de contrôle a été réalisée par le principe du fonctionnement des diagrammes d'états à partir desquels le circuit d'implémentation est construit par des techniques utilisant des circuits à base de mémoires. Dans notre travail, le bloc de contrôle est construit principalement à partir des compteurs, des registres à décalage et des comparateurs permettant un déroulement séquentiel des opérations de commande. D'après la littérature [9]-[29], les circuits logiques du bloc de contrôle sont construits sur la base de l'étude des circuits séquenceurs qui permettent le contrôle de ces opérations. Dans cette conception nous allons proposés une implémentation à base d'un séquenceur principal qui traduit le fonctionnement du diagramme d'état du bloc de contrôle en une architecture d'implémentation séquentielle et synchronisée.

4.4.1 Diagramme d'état du fonctionnement du bloc de contrôle

Sur la figure 4.14, nous montrons le fonctionnement du diagramme d'états qui contient 8 états pour le traitement du bloc de contrôle. Dans ce diagramme, nous avons fais apparaître les conditions de traitement des transitions entre les états du diagramme qui ont étaient traduites en signaux d'entrées dans la conception du générateur de séquence principale du bloc de contrôle.

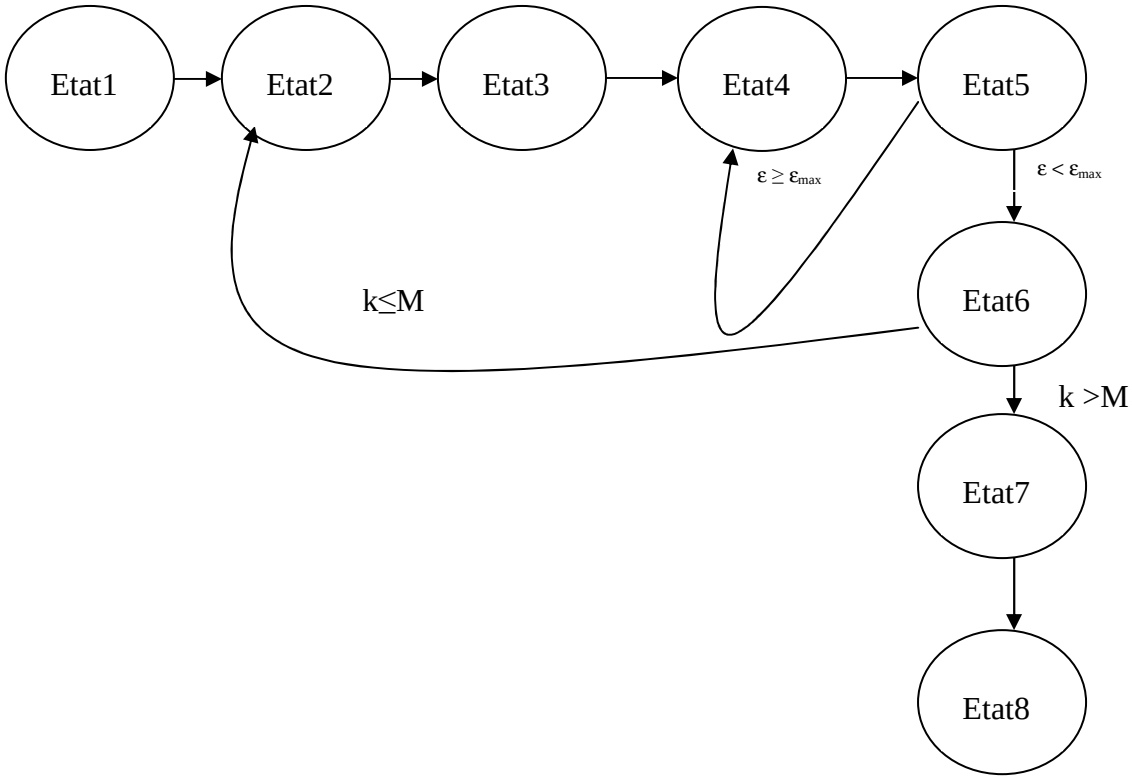


Figure 4.14 : Diagramme d'état du fonctionnement du bloc de contrôle

D'où, la relation ($\varepsilon \geq \varepsilon_{max}$) qui indique la condition de passage de l'état 5 vers l'état 4 si la condition de passage est vérifiée, sinon la transition se fait vers l'état 6, et la relation ($k \leq M$) indique la valeur de la condition de passage de l'état 6 à l'état 2 sinon la transition se fait vers l'état 7. Le traitement de génération des signaux de contrôle de chaque état est la suivante:

Etat 1 : état de contrôle pour le démarrage du système.
Etat 2 : état de contrôle pour l'ajustement du compteur d'adressage de la RAM1 vers les valeurs de signaux d'entrées de l'échantillon k et de l'ajustement du compteur d'adressage de la RAM3 vers les valeurs des poids synaptiques initiaux.
Etat 3 : état de contrôle pour le calcul: des valeurs de l'entrée synaptique, de la fonction non linéaire d'apprentissage et ensuite de l'ajustement des poids synaptiques.
Etat 4 : état de contrôle pour l'ajustement du compteur d'adressage de la RAM1 vers les valeurs de signaux d'entrées de l'échantillon k et l'ajustement du compteur d'adressage de la RAM3 vers les valeurs de l'ajustement des poids synaptiques au cours des autres itérations.
Etat 5 : état de contrôle pour le calcul des valeurs de l'entrée synaptique, de la fonction non linéaire d'apprentissage et des valeurs de l'ajustement des poids synaptique de l'itération précédente.
Etat 6 : état de contrôle pour les calculs des valeurs de l'accumulation des poids synaptiques ajustés.

Etat 7 : état de contrôle pour le traitement des signaux de contrôle pour le calcul des valeurs de l'ajustement des poids synaptiques finaux.
Etat 8 : état de contrôle pour stabiliser le système au repos.

4.4.2 Diagrammes d'états du fonctionnement de calcul de l'entrée synaptique et de la fonction non linéaire d'apprentissage

Les états du diagramme donnés par la Figure 4.15 illustrent les traitements qui réalisent le contrôle de l'entrée synaptique et de la fonction non linéaire d'apprentissage pendant plusieurs cycles horloges de l'état3 où la relation (i≤N) indique la valeur de la condition de passage de l'état33 à l'état31 si la condition de passage est vérifiée, sinon la transition se fait vers l'état34.

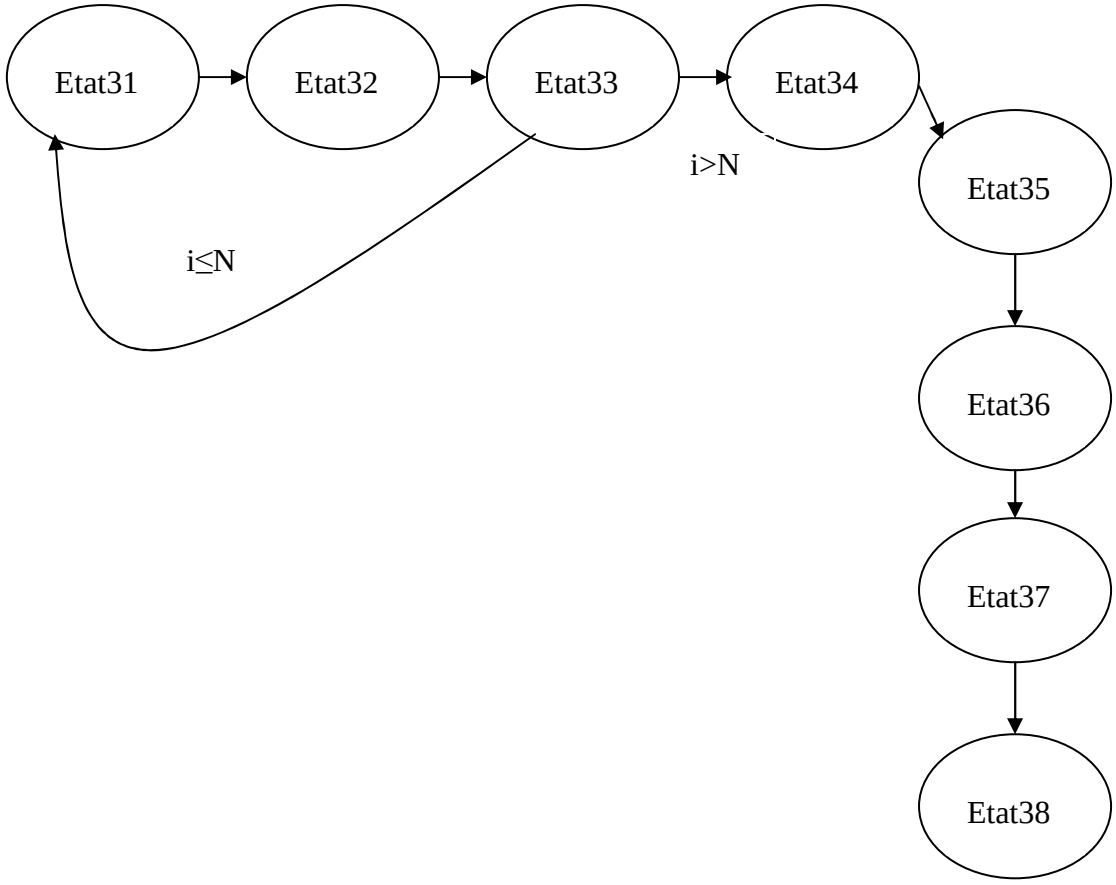


Figure 4.15 : Diagramme d'état de fonctionnement du calcul de l'entrée synaptique et de la fonction non linéaire d'apprentissage

Le diagramme d'état de la Figure 4.16 illustre le traitement des contrôles de l'état3 pour l'ajustement des poids synaptiques. Chaque état correspond à un cycle d'horloge pour le traitement d'une opération de calcul de l'algorithme.

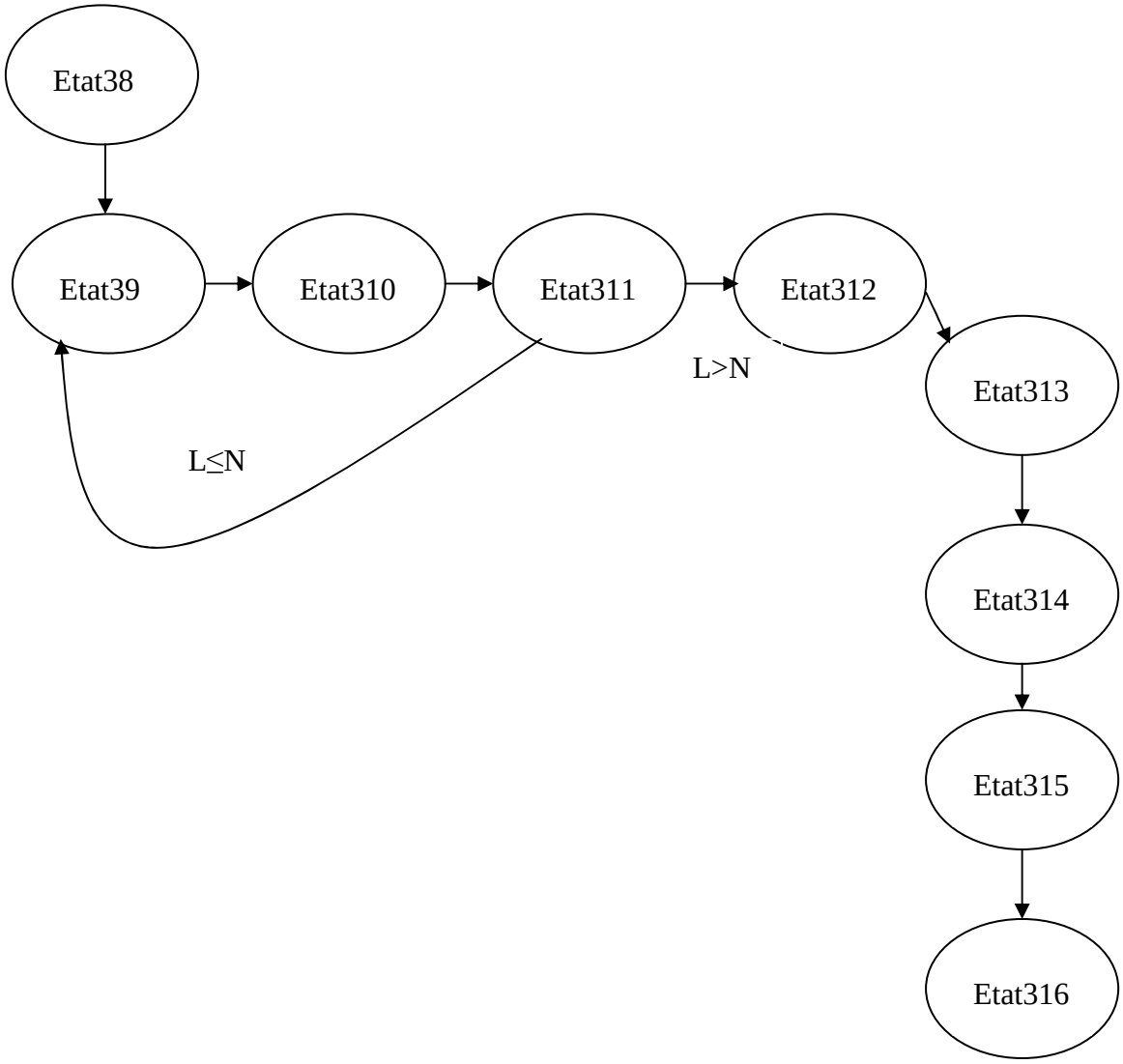


Figure 4.16 : Diagramme d'état de fonctionnement du calcul de l'ajustement des poids synaptiques

La relation (L≤N) indique la condition de passage de l'état31 à l'état39, sinon la transition se fait vers l'état312.

Dans ce qui suit, nous allons exploiter le fonctionnement des diagrammes d'états pour permettre la réalisation de l'architecture d'implémentation du bloc de contrôle qui sert à commander les opérations séquentielles de l'algorithme optimisé.

4.4.3 Conception de l'architecture d'implémentation du bloc de contrôle

La conception du circuit d'implémentation du bloc de contrôle est réalisée suivant une certaine méthodologie de traitement séquentiel pour réaliser la commande du bloc opérationnel. La Figure 4.17 illustre le schéma bloc d'implémentation du bloc de contrôle qui propose l'implémentation d'un séquenceur principal pour la commande du bloc de traitement.

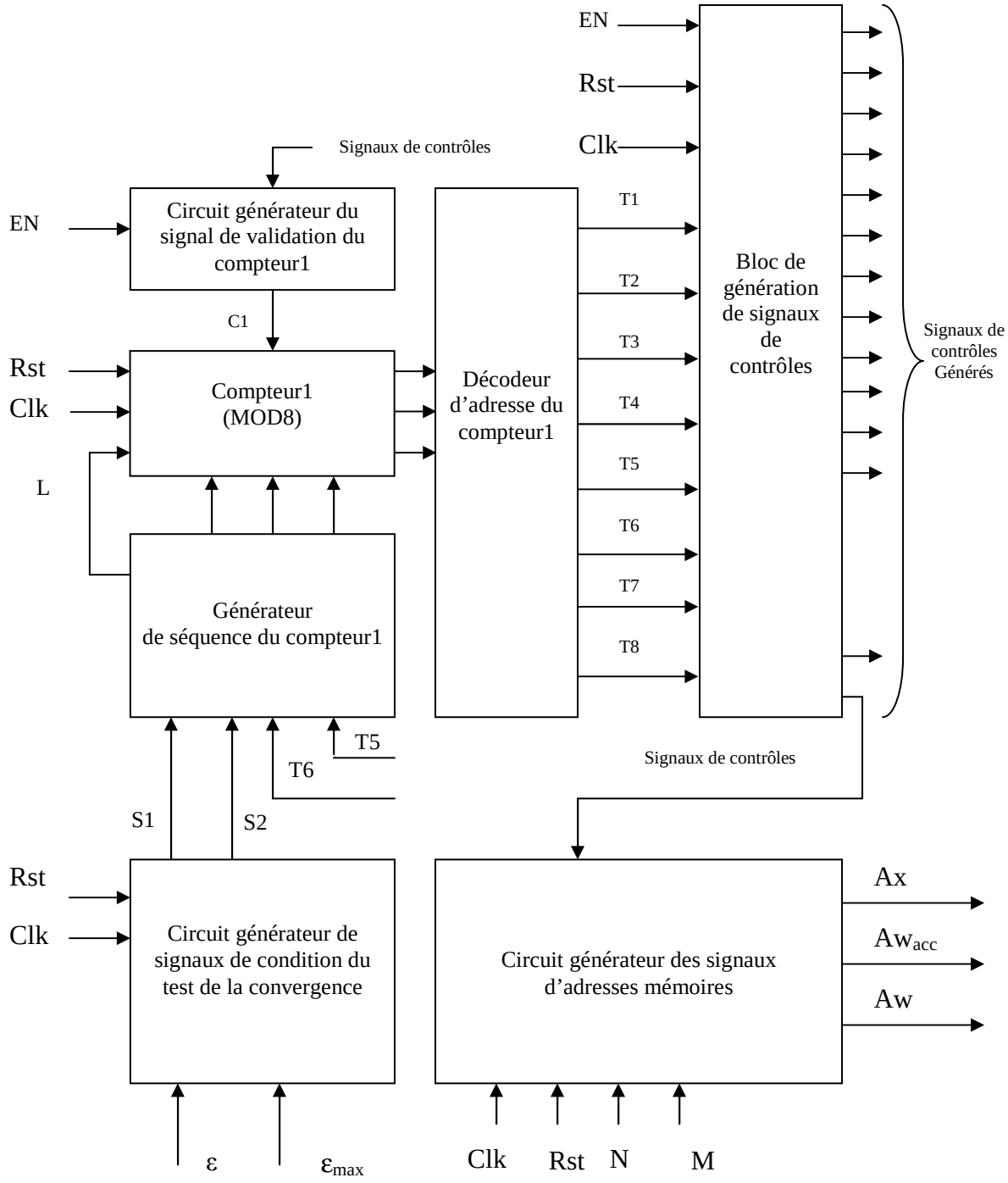


Figure 4.17 : Schéma bloc d'implémentation du bloc de contrôle

Le schéma d'implémentation du séquenceur principal est constitué de, i) un compteur1 (MOD8), d'un générateur de séquence, ii) un circuit générateur de signaux de condition de test de convergence, iii) un circuit générateur du signal de validation du compteur1, iv) un circuit générateur des signaux d'adresses mémoires, et d'un bloc de génération des signaux de contrôles qui permet le traitement des signaux T1 à T8 fournis par le décodeur d'adresse du compteur1 pour le contrôle des états du diagramme fonctionnel du bloc de contrôle.

Le séquenceur principal permet le contrôle du bloc de traitement, le contrôle du circuit générateur du signal de validation du compteur1 ainsi que le contrôle du circuit générateur des signaux d'adresses mémoire où EN est le signal de validation du compteur, Clk est le signal d'horloge, et Rst est le signal de RAZ du compteur, L : signal de chargement du compteur, avec:

T1 : est le signal de contrôle de l'état1,
T2 : est le signal de contrôle de l'état2,
T3 : est le signal de contrôle de l'état3,
T4 : est le signal de contrôle de l'état4,
T5 : est le signal de contrôle de l'état5,
T6 : est le signal de contrôle de l'état6,
T7 : est le signal de contrôle de l'état7,
T8 : est le signal de contrôle de l'état8.

4.4.3.1 Architecture du circuit logique d'implémentation du bloc Générateur de signaux de condition du test de la convergence

La Figure 4.18 présente l'architecture du circuit logique d'implémentation du bloc Générateur de signaux de condition du test de la convergence qui est composée d'un circuit générateur de condition S1 et d'un circuit générateur de condition S2.

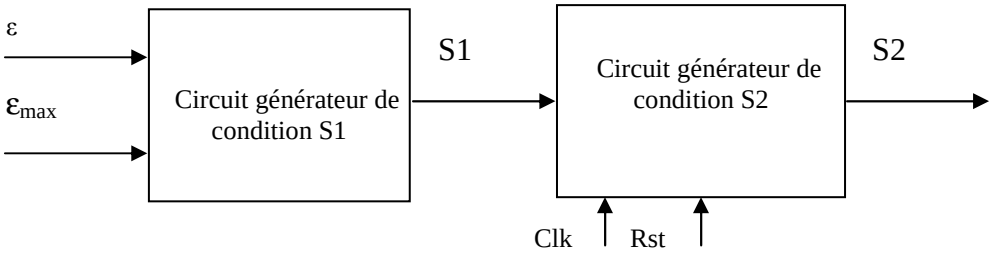


Figure 4.18 : Schéma d'implémentation du signal de condition du test de la convergence

- ***Génération du signal de condition S1***

Dans la Figure 4.19, on illustre le schéma d'implémentation du signal de condition S1. Il est composé par un comparateur qui indique la convergence des poids synaptiques ajustés en fonction des valeurs de ε et de ε_{max} , si est seulement si le signal S1 se met à 1.

D'où : S1=0, pour $\varepsilon \geq \varepsilon_{max}$; et S1=1, pour $\varepsilon < \varepsilon_{max}$.

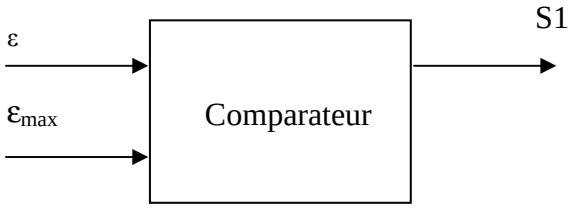


Figure 4.19 : Schéma d'implémentation du signal de condition S1

- ***Génération du signal de condition S2***

Dans la Figure 4.20, on illustre le schéma d'implémentation du signal de condition S2. Nous avons implémenté un compteur MOD M pour le comptage de la valeur de l'échantillon k et un comparateur qui fournit le signal de condition S2 pour l'arrêt du processus de comptage après traitement des échantillons des signaux d'entrées.

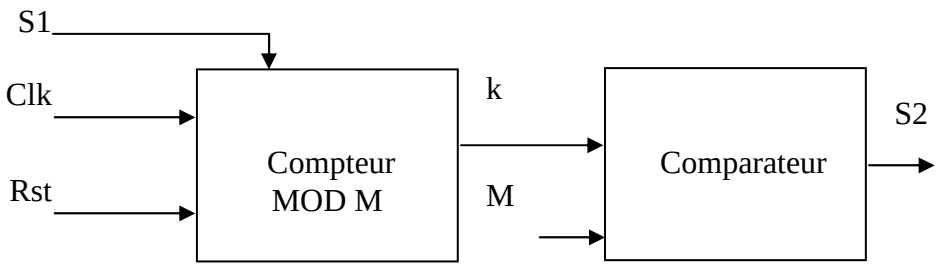


Figure 4.20 : Schéma d'implémentation du signal de condition S2

M étant le nombre maximum d'échantillons des signaux d'entrées, et k le nombre de chaque échantillon. Le signal de validation du compteur reçoit le signal de condition S1 pour le comptage MOD M.

4.4.3.2. Architecture du circuit d'implémentation du bloc de génération de signaux de contrôle

Dans le circuit d'implémentation du bloc de contrôle, nous avons implémenté deux bascules et six séquenceurs pour le contrôle du diagramme d'état qui permettent de contrôler le bloc de traitement et la validation du séquenceur principal et de la génération de la séquence de contrôle des circuits de validation (bascules et séquenceurs) et enfin le contrôle du circuit générateur des signaux d'adresses.

4.4.3.2.1 Implémentation de la génération de signaux de validation du bloc générateur de signaux de contrôles

Dans la Figure 4.21, nous avons implémenté les signaux de validation des différents compteurs et bascules du bloc de contrôle pour le traitement séquentiel du bloc de traitement dans lequel *EN*, est un signal qui permet de désactiver tous les signaux de contrôle lors de la mise à zéro pour arrêter le processus de calcul dans le bloc de traitement, et l'activation des signaux de validation lors de la mise à 1.

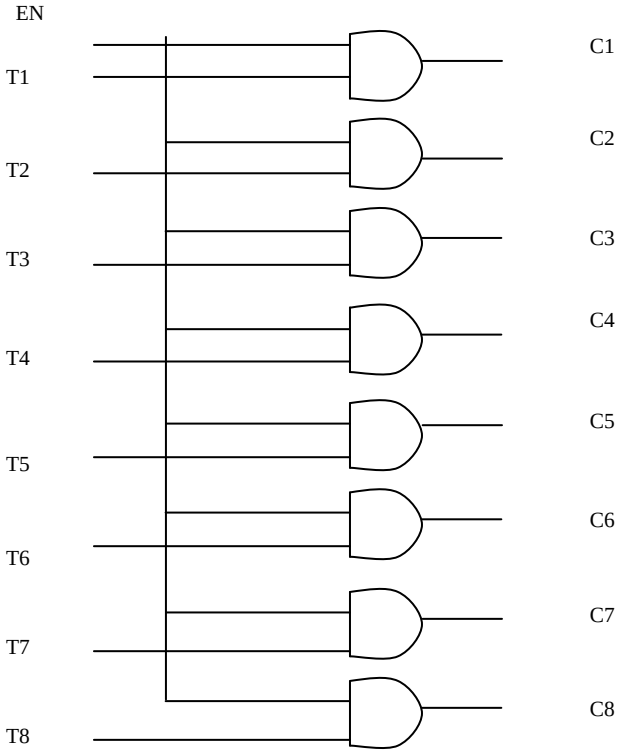


Figure 4.21 : Implémentation des signaux de contrôle

4.4.3.2.2 Implémentation de la génération de signaux de contrôle du traitement du signal T1

La bascule D1 reçoit le signal C1 qui active le signal T1 et le signal de contrôle L1 généré à la sortie de la bascule. La bascule D1 reçoit le signal de remise à zéro Rst et le signal d'horloge Clk. Le schéma d'implémentation du signal T1 est illustré par la Figure 4.22.

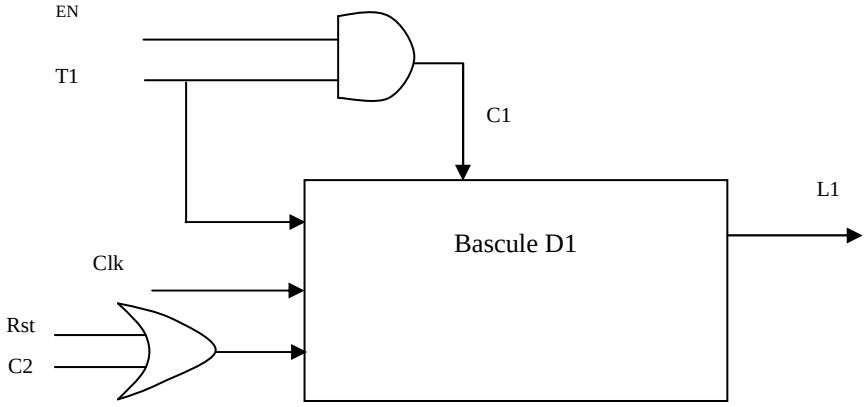


Figure 4.22 : Schémas d'implémentation du traitement du signal T1

4.4.3.2.3 Implémentation de la génération de signaux de contrôle du traitement du signal T2

Le circuit de traitement du signal T2 est une bascule D2 qui génère les signaux de contrôle We1 et We2 pour la lecture de la mémoire RAM1 des échantillons des signaux d'entrées et la mémoire RAM3 de l'ajustement des poids synaptiques. Ainsi, la bascule D2 qui reçoit le signal de validation C_2 permet d'activer la mise à 1 des signaux de contrôles We1 et We2. Le schéma d'implémentation du signal T2 est illustré par la Figure 4.23.

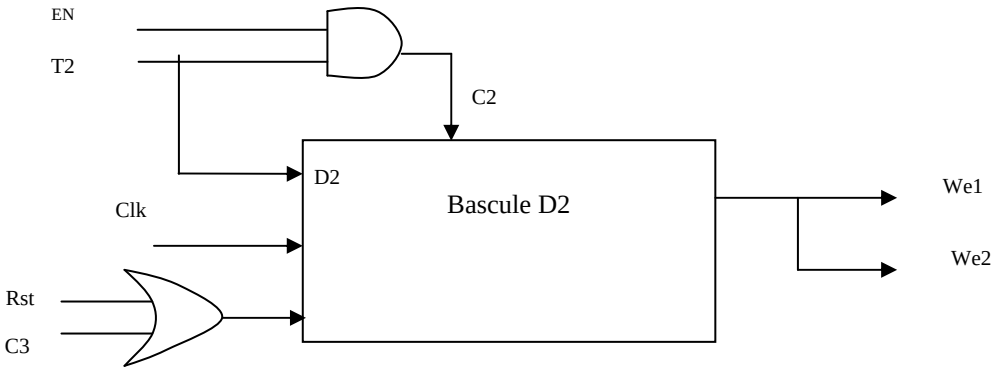


Figure 4.23 : Schéma d'implémentation de traitement du signal T2

4.4.3.2.4 Implémentation des signaux de contrôle du signal T3

Les signaux de contrôle de l'implémentation du signal T3 sont générés par un séquenceur qui est composé d'un compteur2 MOD 16. Il permet d'effectuer la séquence des signaux de contrôle pour le traitement du signal T3 et d'un générateur de séquence du compteur2 qui reçoit les signaux de conditions S3, S4, qui permettent d'effectuer le test du traitement de la fonction d'apprentissage de tous les neurones. D'où, le schéma d'implémentation qui permet d'illustrer le traitement du signal T3 est représenté par la Figure 4.24.

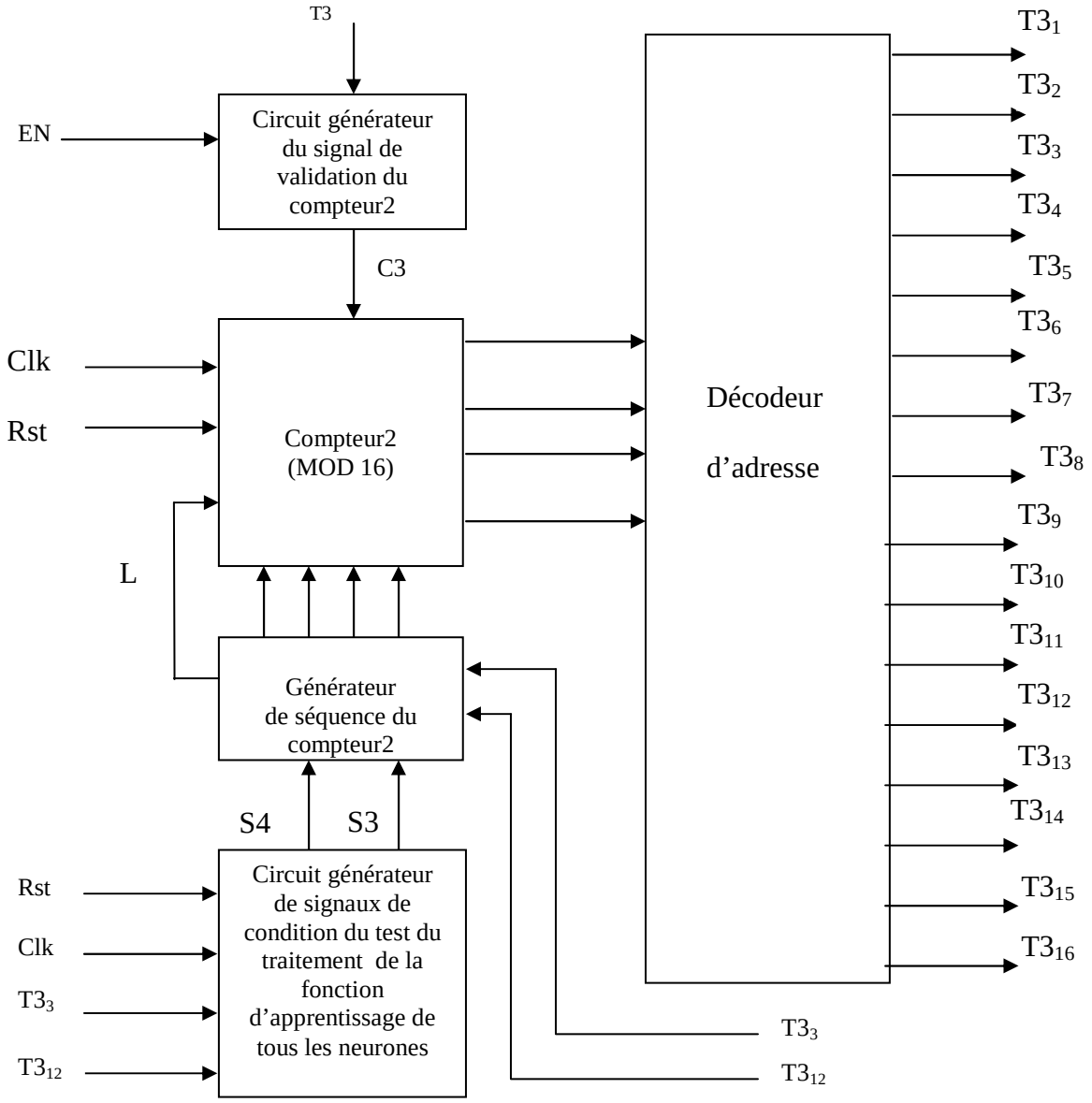


Figure 4.24 : Schéma d'implémentation de la génération des signaux de contrôles du traitement du signal T3

- ***Schéma d'implémentation de l'activation des signaux de contrôle du signal T3***

Le circuit d'implémentation des signaux de contrôle du signal T3 (figure 4.25) permet de désactiver les signaux de contrôle lors de la mise a zéro du signal C3.

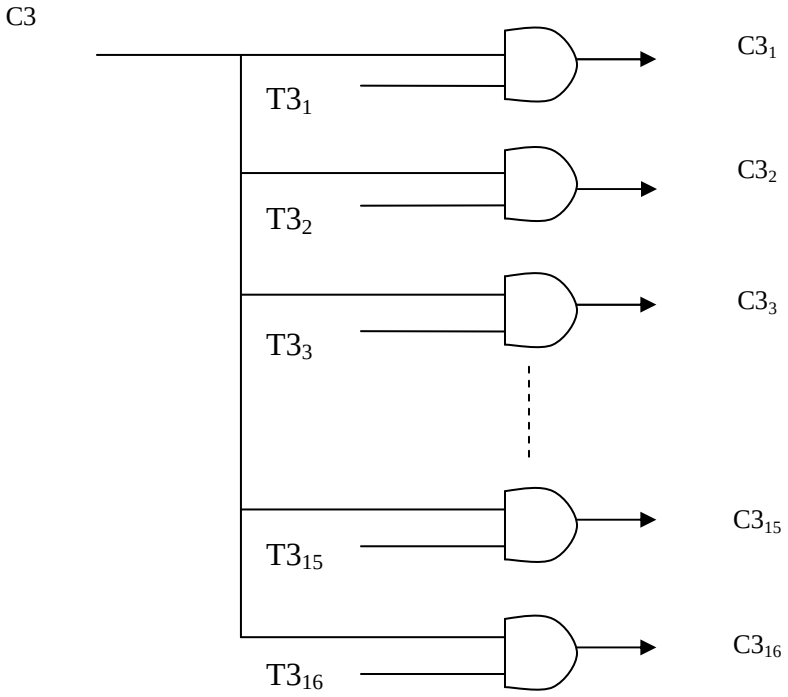


Figure 4.25 : Implémentation de validation des signaux de contrôles de traitement du signal T3

- ***Schémas d'implémentation du circuit générateur du signal condition S3***

La Figure 4.26 représente le schéma d'implémentation du circuit générateur du signal de condition S3.

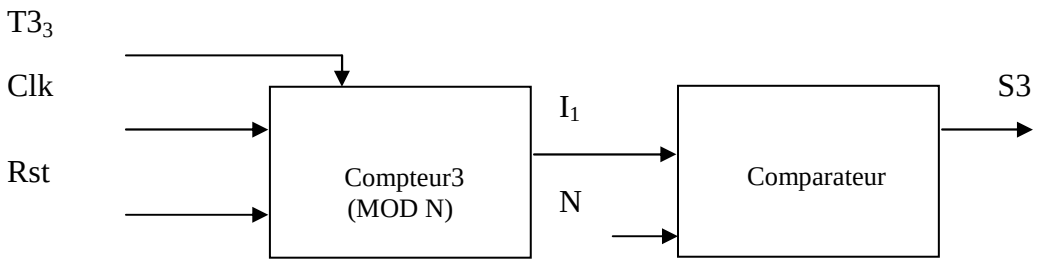


Figure 4.26 : Implémentation du signal de condition S3

$$\begin{cases} I_1 < N \quad \text{pour} \quad S3 = 0, \\ I_2 = N \quad \text{pour } S3 = 1. \end{cases}$$

La valeur de l'entier N représente le nombre de signaux d'entrée.

- ***Schéma d'implémentation du circuit générateur du signal condition S4***

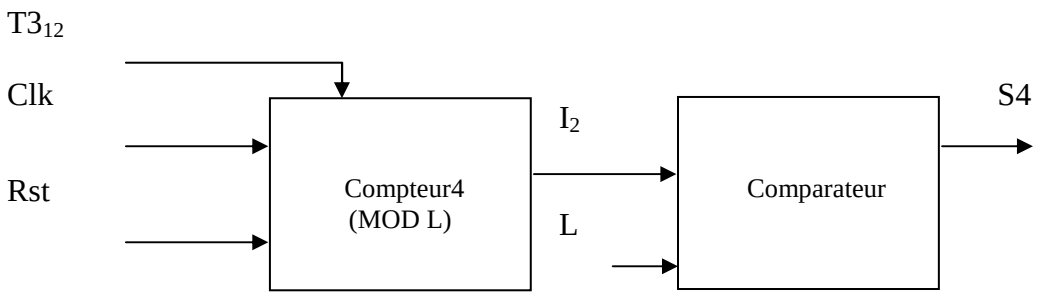


Figure 4.27 : Implémentation du circuit générateur du signal de condition S4

$\begin{cases} I_2 < L \text{ pour } S4=0, \\ I_2 = L \text{ pour } S4=1. \end{cases}$

Le nombre L représente le nombre des valeurs de l'ajustement des poids synaptiques.

4.4.3.2.5 Implémentation des signaux de contrôle du traitement du signal T4

La génération de signaux de contrôle pour le traitement du signal T4 permet l'ajustement du compteur d'adressage de la RAM1 vers les valeurs des signaux d'entrée de l'échantillon k et celui du compteur d'adressage de la RAM3 vers les valeurs de l'ajustement des poids synaptiques au cours des autres itérations.

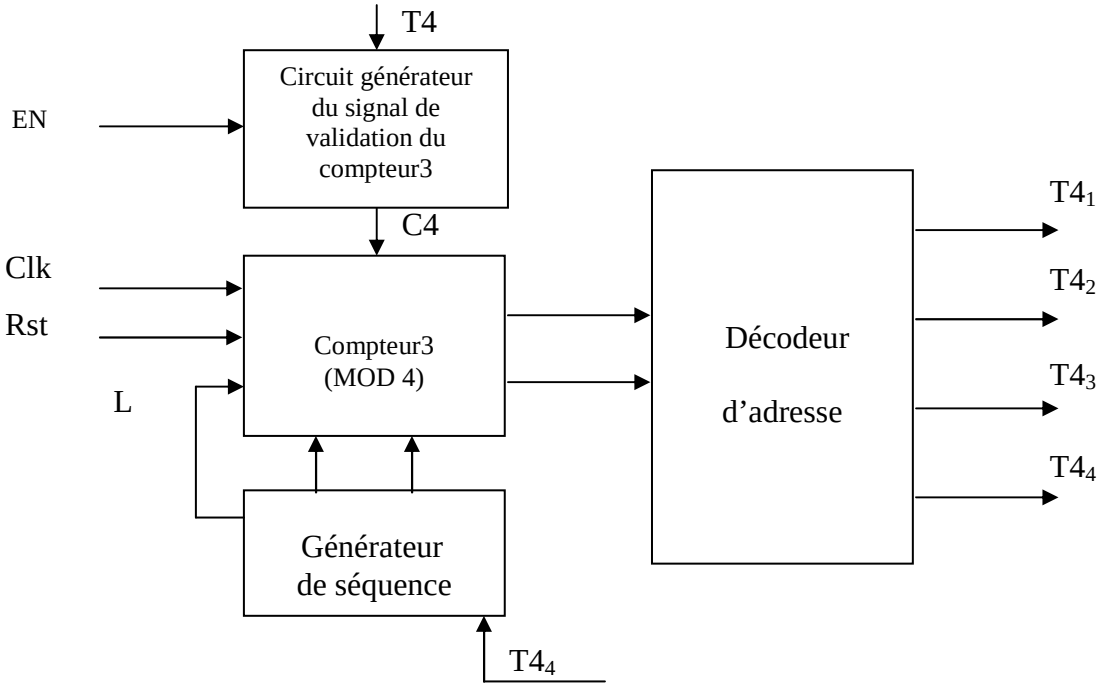


Figure 4.28 : Implémentation du traitement du signal T4

D’où, le schéma d’implémentation du signal T4 qui est illustré par la Figure 4.28 qui est un séquenceur composé d'un compteur3 MOD 4, d’un générateur de séquence, et d’un circuit générateur du signal de validation du compteur3.

- ***Schéma d’implémentation de validation des signaux de contrôle de traitement du signal T4***

La mise a zéro du signal C4 permet de désactiver les signaux de contrôle du signal T4. Le schéma d’implémentation de la Figure 4.29 représente le circuit d’implémentation de validation des signaux de contrôle de traitement du signal T4.

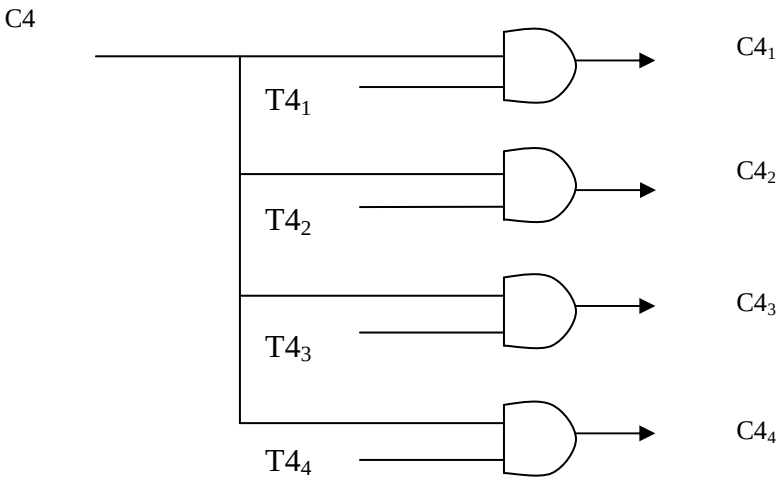


Figure 4.29 : Implémentation de validation des signaux de contrôle de du signal T4

4.4.3.2.6 Implémentation des signaux de contrôle du signal T5

La génération de signaux de contrôle du signal T5 permet le contrôle de calculs en pipeline de des valeurs de l’entrée synaptique, et celui des valeurs de la fonction non linéaire d’apprentissage qui seront utilisées pour calcul des valeurs de l’ajustement des poids synaptiques de l’itération présente, et enfin le contrôle de la partie de calculs des valeurs de l’ajustement des poids synaptiques de l’itération précédente. Ainsi, ce contrôle permet d’effectuer les calculs de l’ajustement des poids synaptiques avec un seul balayage de la mémoire de l’ajustement des poids synaptiques. Ce qui permet d’augmenter la vitesse de calcul de traitement. La Figure 4.30 illustre le schéma d’implémentation pour le contrôle du traitement du signal T5.

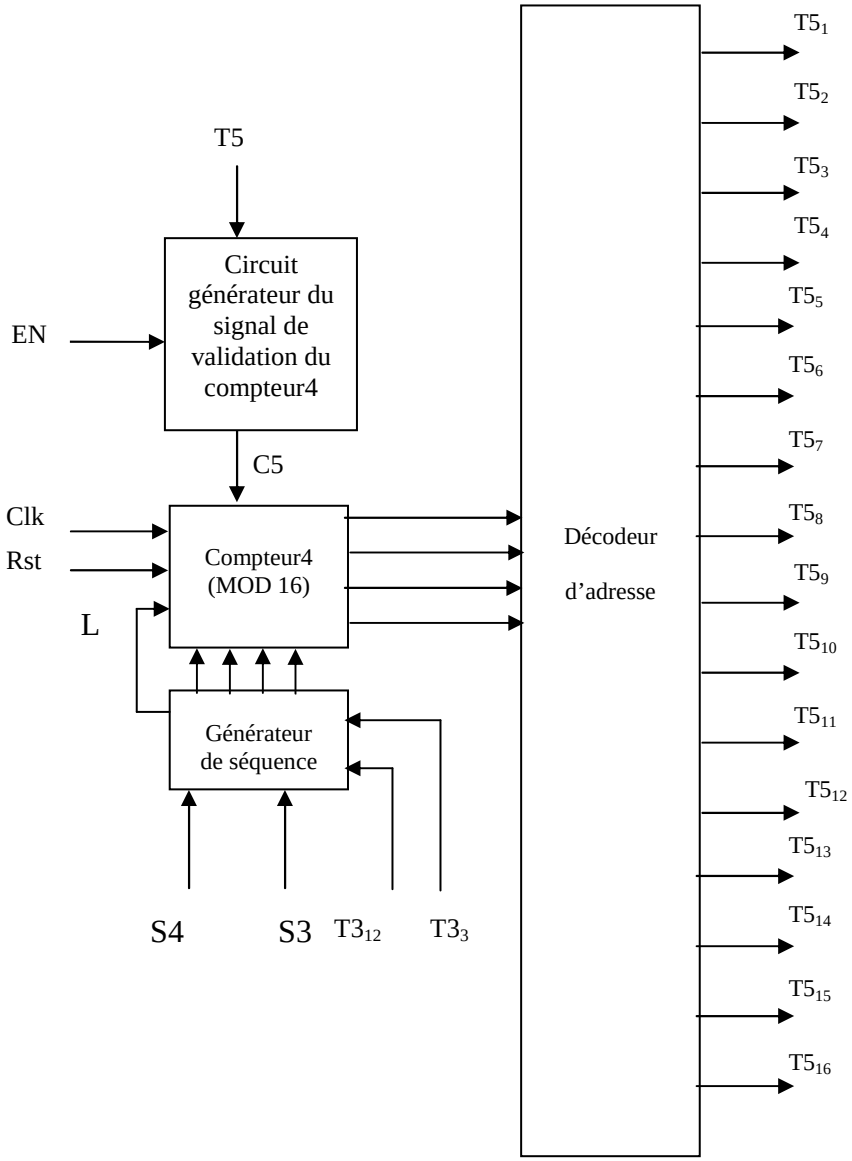


Figure 4.30 : Implémentation de la génération des signaux de contrôle du traitement du signal T5

- ***Schéma d'implémentation de validation des signaux de contrôle du signal T5***

La mise à 1 du signal C5 permet de désactiver les signaux de contrôle pour le traitement du signal T4. Le schéma d'implémentation de la Figure 4.31 donnee le circuit d'implémentation de validation des signaux de contrôle du signal T4.

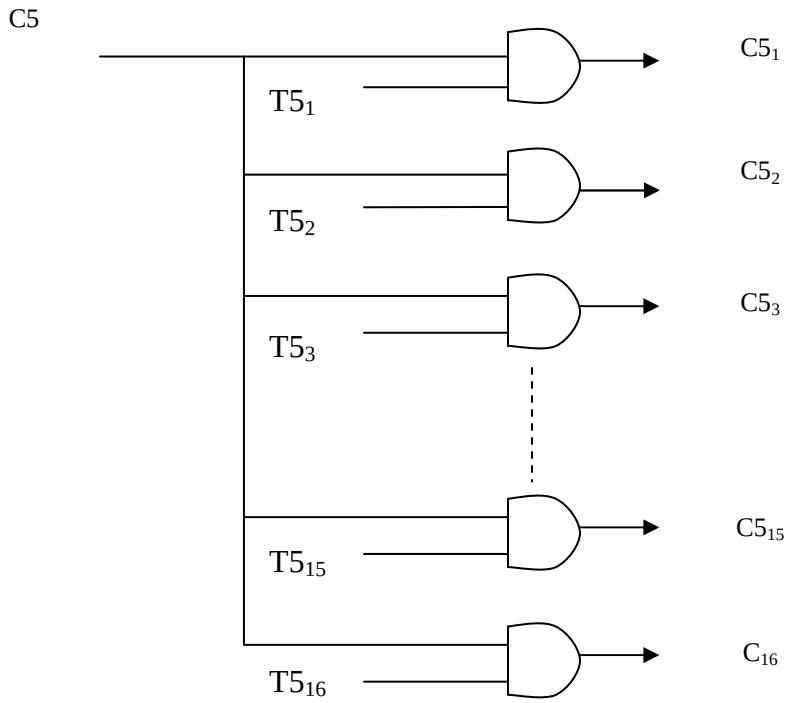


Figure 4.31 : Implémentation de la validation des signaux de contrôles de traitement du signal T5

4.4.3.2.7 Implémentation de la génération de signaux de contrôle du traitement du signal T6

L'implémentation du traitement du signal T6 est un séquenceur qui permet les calculs de l'accumulation des poids synaptiques ajustés. Il est composé d'un compteur5 MOD 8, d'un circuit générateur du signal de validation du compteur, et d'un générateur de séquence qui permet de recevoir les signaux de contrôle et de condition S8 pour le test de l'accumulation de des poids synaptiques ajustés. Le circuit d'implémentation du signal T6 est illustré par la Figure 4.32.

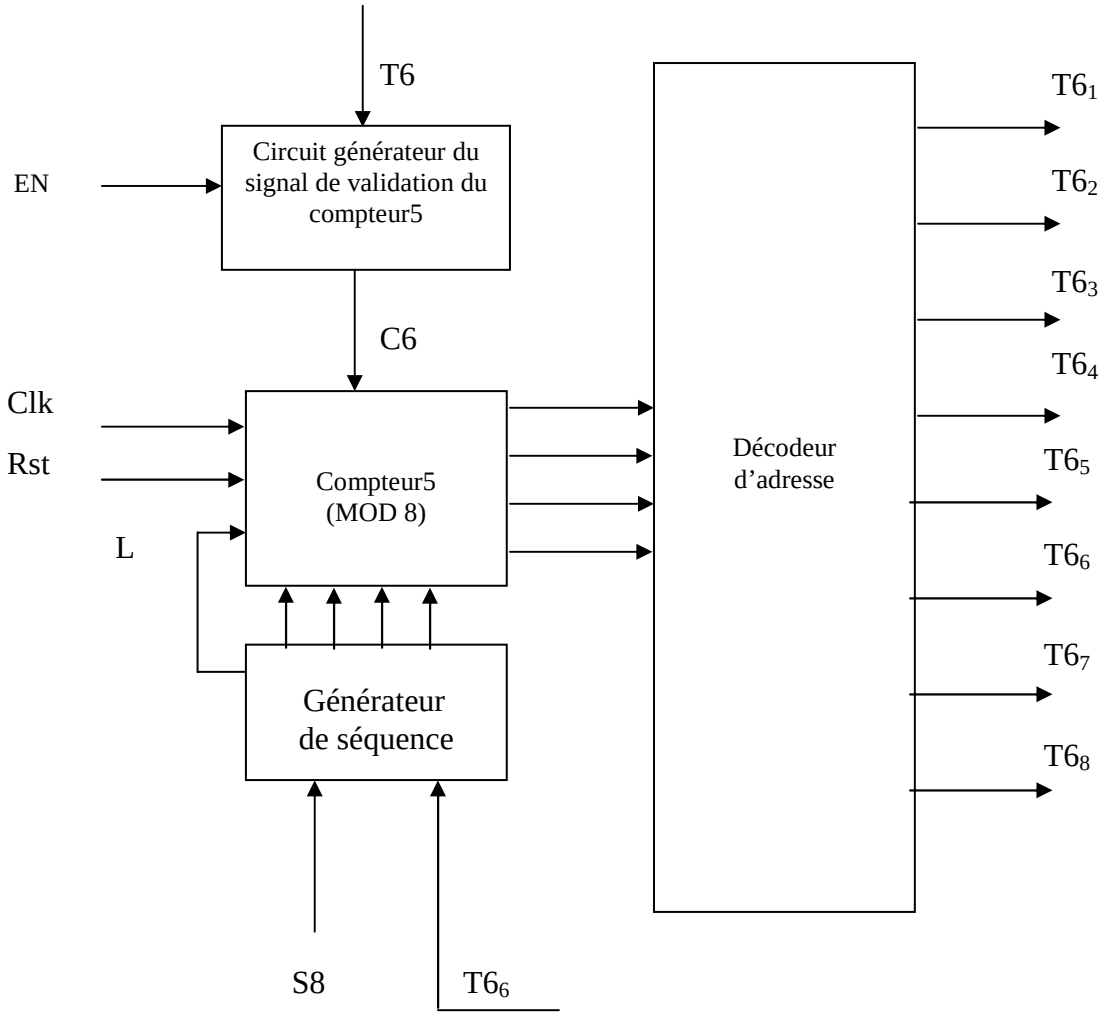


Figure 4.32 : Implémentation de la génération de signaux de contrôles du traitement du signal T6

- ***Schéma d'implémentation de validation des signaux de contrôle du signal T6***

La mise à zéro du signal C6 permet de désactiver les signaux de contrôle du signal T6. Le schéma d'implémentation de la Figure 4.33 représente le circuit d'implémentation des signaux de contrôle du signal T6.

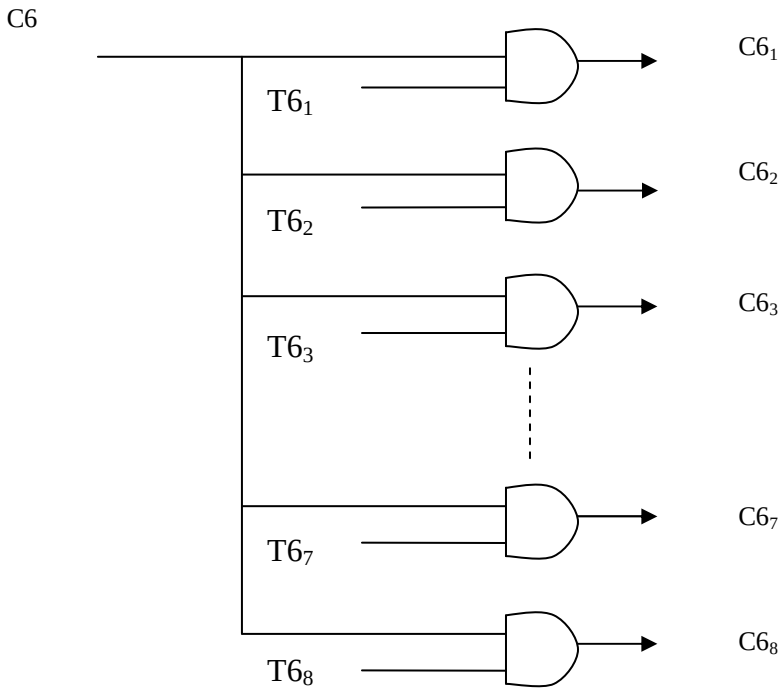


Figure 4.33 : Implémentation de la validation des signaux de contrôles de traitement du signal T6

4.4.3.2.8 Implémentation de la génération de signaux de contrôle du traitement du signal T7

L'implémentation des signaux de contrôle du signal T7 est effectuée par un séquenceur pour le contrôle de l'ajustement des poids synaptiques finaux. Il est composé d'un compteur MOD 4, d'un circuit générateur du signal de validation du compteur, et d'un circuit générateur de séquence qui reçoit les signaux de contrôle et de condition S7 pour le test de l'ajustement des poids synaptiques finaux. Le circuit de traitement du signal T7 pour le contrôle de l'ajustement des poids synaptiques finaux est illustré par la Figure 4.34.

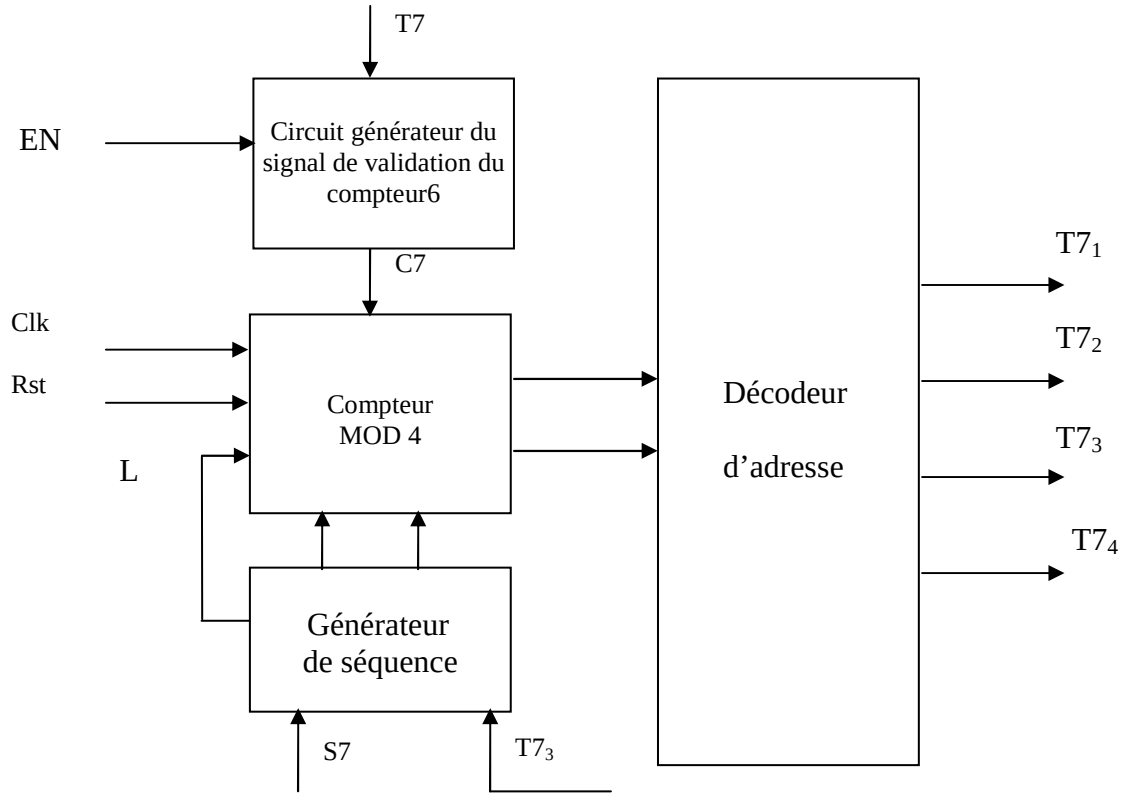


Figure 4.34 : Implémentation du traitement du signal T7

- ***Schéma d'implémentation de la validation des signaux de contrôle de traitement du signal T7***

La mise à zéro du signal C7 permet de désactiver les signaux de contrôle pour le traitement du signal T7. Le schéma d'implémentation de la Figure 4.35 représente le circuit de validation des signaux de contrôle du signal T7.

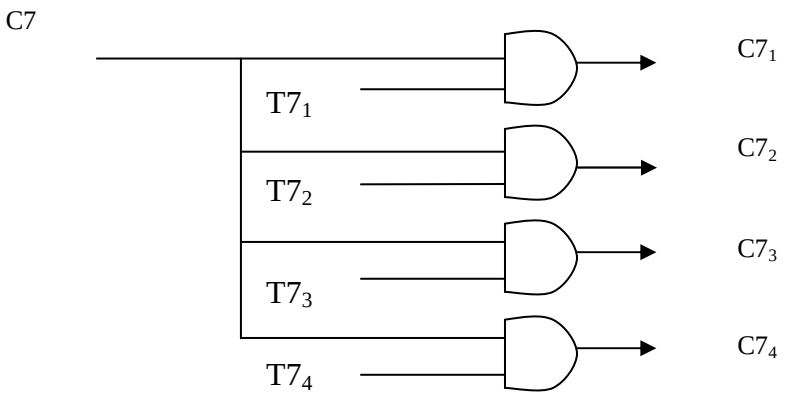


Figure 4.35 : Implémentation de validation des signaux de contrôles du signal T7

4.4.3.2.9 Implémentation de génération de signaux de traitement du signal T8

L'architecture d'implémentation des signaux de contrôle du signal T8 est représentée par la figure 4.36. Il permet le contrôle de la stabilisation du système au repos.

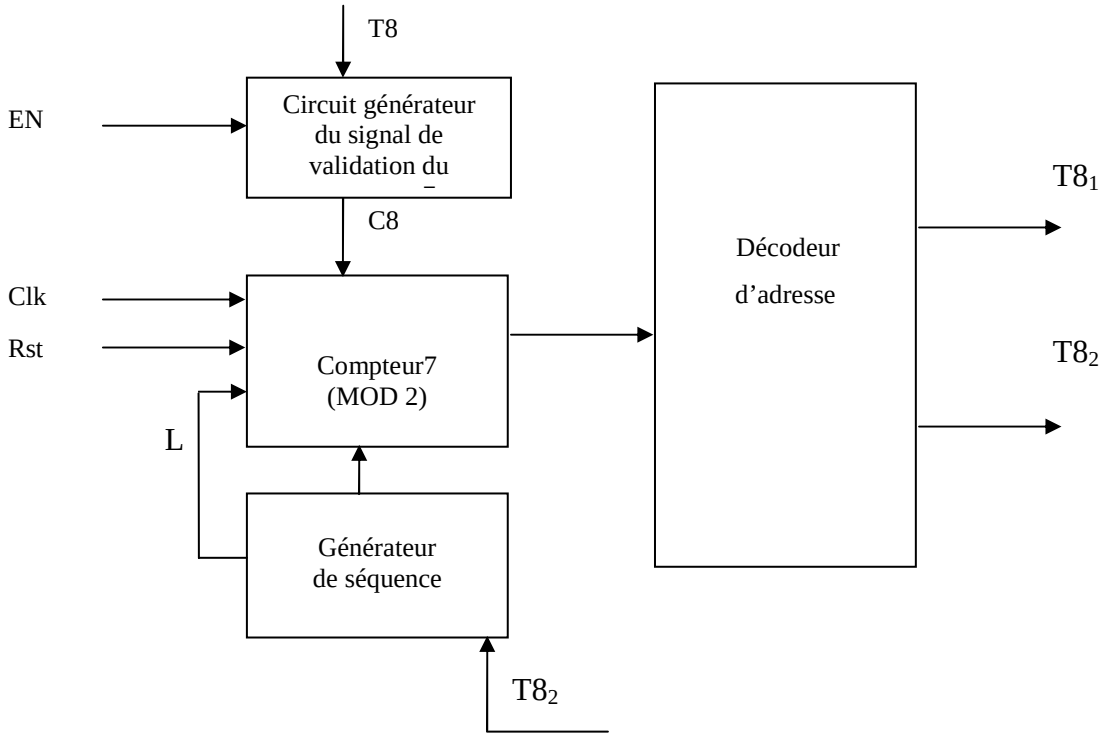


Figure 4.36 : Implémentation de la génération des signaux de contrôle du traitement du signal T8

- ***Schéma d'implémentation de la validation des signaux de contrôle de traitement du signal T8***

La mise a zéro du signal C8 permet de désactiver les signaux de contrôle pour le traitement du signal T8. Le schéma d'implémentation de la Figure 4.37 représente le circuit d'implémentation de validation des signaux de contrôle du signal T8.

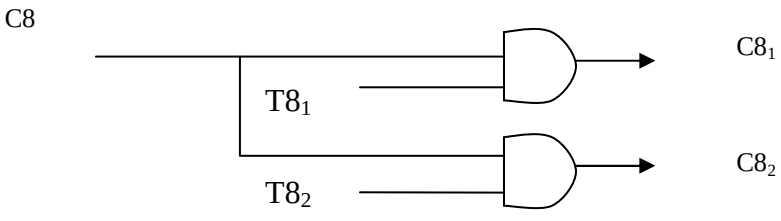


Figure 4.37 : Implémentation de validation des signaux de contrôles de traitement du signal T8

4.4.3.3 Implémentation du circuit d'adressage des mémoires

La figure 4.38 illustre l'adressage de la mémoire RAM1 des signaux d'entrée, celui de la mémoire des valeurs de l'ajustement des poids synaptiques RAM2, et de la mémoire des valeurs de l'accumulation des poids synaptiques ajustés RAM3.

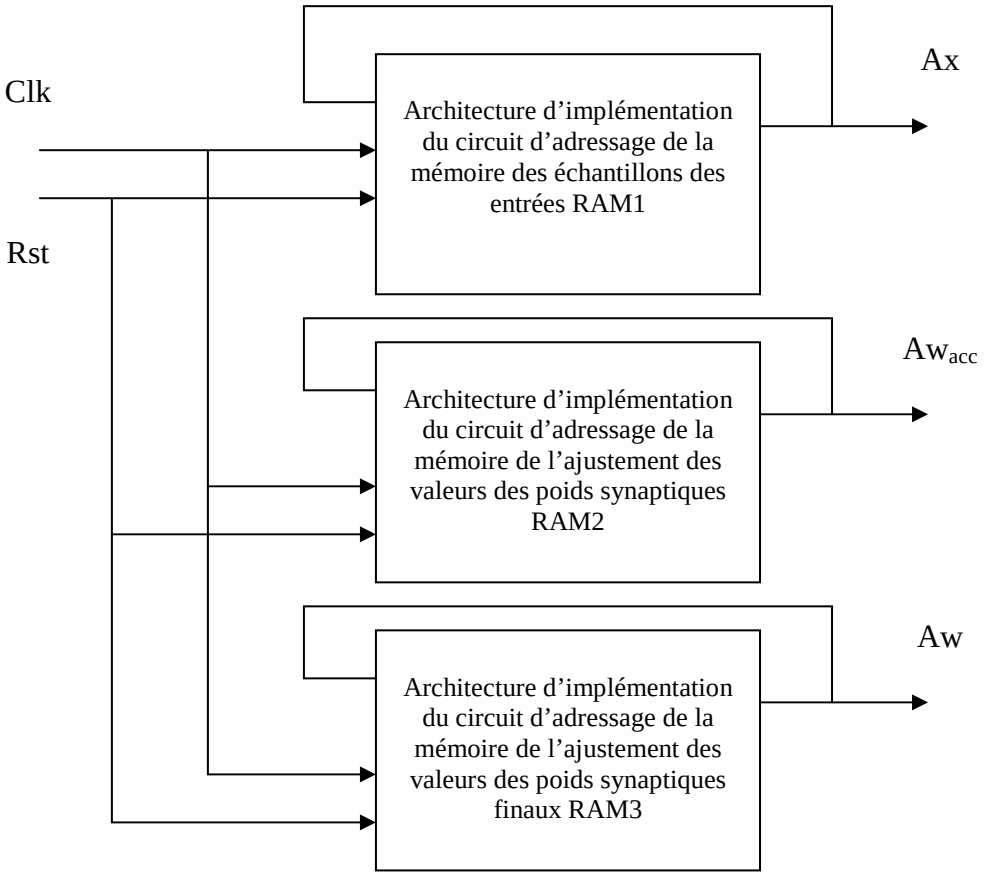


Figure 4.38 : Implémentation du circuit d'adressage des mémoires

4.4.3.3.1 Implémentation du circuit d'adressage de la mémoire RAM1 des échantillons des signaux d'entrée

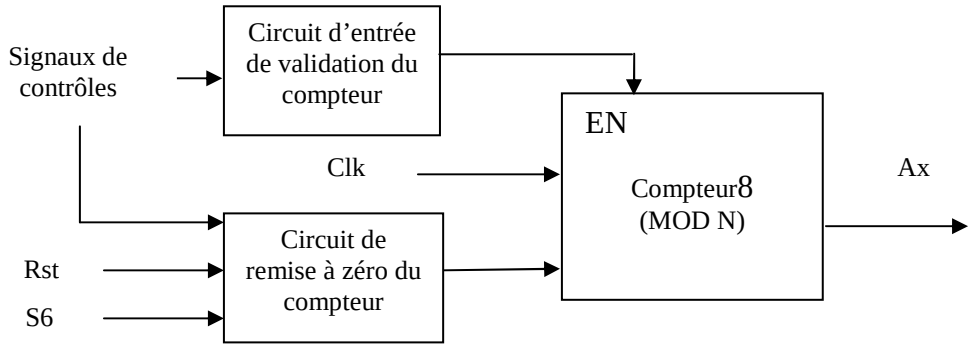


Figure 4.43 : Implémentation du circuit d'adressage de la mémoire de l'ajustement des poids synaptiques

Soit A_x le signal de l'adresse mémoire RAM1 des échantillons des signaux d'entrée qui représente la valeur du compteur8 MOD N. Le compteur d'adresse reçoit le signal S6 qui permet d'effectuer la RAZ du compteur. N représente le nombre de signaux d'entrées.

- ***Implémentation du circuit générateur du signal condition S6***

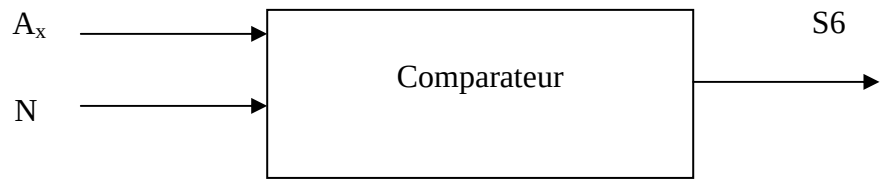


Figure4.40 : Implémentation du circuit générateur du signal condition S6

Le schéma de la Figure 4.40 illustre un comparateur qui effectue la génération du signal de condition S6. D'où, les relations qui permettent d'effectuer les conditions de test :

$$\begin{cases} \text{Si S6=0, } A_x < N \text{ ;} \\ \text{Si S6=1, } A_x = N. \end{cases}$$

4.4.3.3.2 Implémentation du circuit d'adressage de la mémoire de l'ajustement des poids synaptiques RAM2

La Figure 4.43 donne le schéma d'implémentation du circuit d'adressage de la mémoire de l'ajustement des poids synaptiques RAM2 qui fournit le signal d'adresse A_w.

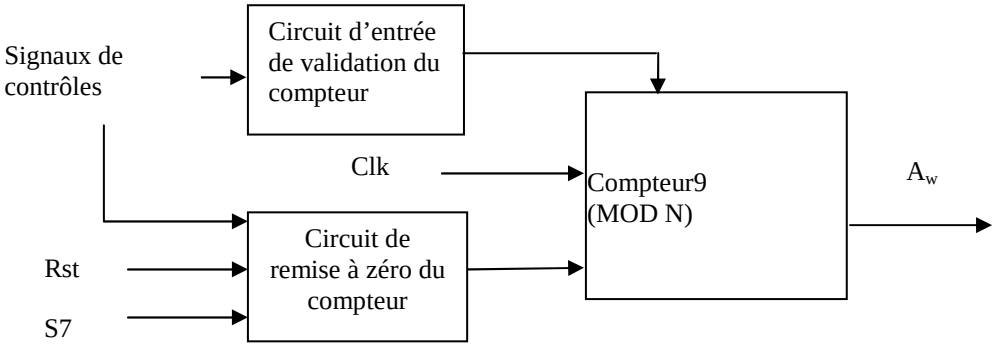


Figure 4.43 : Implémentation du signal d'adresse de la mémoire de l'ajustement des poids synaptiques

Ce dernier est composé d'un compteur9 MOD N (qui reçoit le signal d'horloge de synchronisation Clk), d'un circuit d'entrée de validation du compteur et d'un circuit de RAZ du compteur.

- ***Implémentation du circuit générateur du signal condition S7***

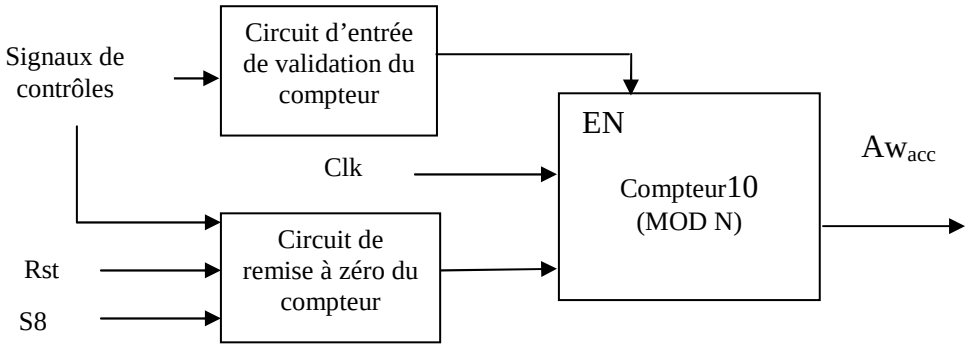
Figure 4.44 : Circuit générateur du signal de condition S7

Le schéma d'implémentation de la condition S7 est représenté par la Figure 4.44. Il permet d'effectuer la remise à zéro du compteur d'adressage de la mémoire de l'ajustement des poids synaptiques RAM2 ainsi que le contrôle des poids synaptiques finaux. D'où, les relations qui permettent d'effectuer le test de la condition S7:

$$\begin{cases} \text{Si S6=0, } A_w < N \text{ ;} \\ \text{Si S6=1, } A_w = N. \end{cases}$$

4.4.3.3.3 Implémentation du circuit d'adressage de la mémoire de l'ajustement des poids synaptiques finaux RAM3

Le schéma de la Figure 4.43 donne le circuit générateur d'adressage de la mémoire RAM3 de l'accumulation des poids synaptiques ajustés.



- Figure 4.43 : Implémentation du circuit générateur d'adresse de la mémoire de l'accumulation des poids synaptiques

Le schéma de la Figure 4.48 est un circuit d'implémentation qui fournit le signal de condition S8. Il permet d'effectuer le test du contrôle de l'ajustement des poids synaptiques accumulés.

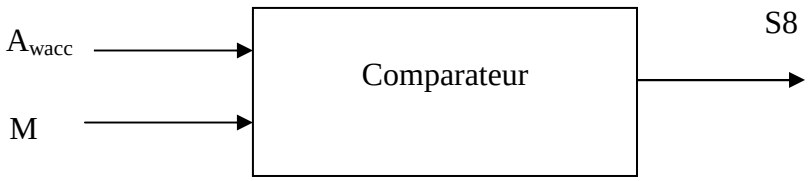


Figure 4.48 : Circuit générateur du signal condition S8

Si S8=0, Aw_{acc} l<M;
Si S8=1, Aw_{acc} l =M.

4.5 Conclusion

La méthodologie de conception qui a été définie dans la mise en œuvre des calculs d'implémentation de l'algorithme nous a permis de réaliser une adéquation Algorithme/Architecture AAA d'un circuit logique moins complexe. En effet, l'architecture du circuit logique d'implémentation de l'AAA a été réalisée pour effectuer les calculs d'un seul neurone, dans le but de réduire la consommation de la surface du circuit matériel. La conception du bloc de contrôle a été mise en œuvre sur la base du fonctionnement des diagrammes d'états. Cela a permis de réaliser l'architecture d'implémentation du bloc de contrôle en utilisant le principe du fonctionnement des circuits séquenceurs.

Chapitre 5

Conception de l'architecture du circuit logique d'implémentation sur le circuit matériel

5.1 Introduction

Afin d'évaluer les performances de l'architecture du circuit logique implémenté, nous élaborons en premier lieu la description en langage VHDL, ainsi que les tests de simulations. L'objectif étant l'évaluation des résultats de simulation en VHDL du circuit matériel, dont les performances, permet l'ajustement des poids synaptiques en temps réel. Nous réaliserons des tests de simulation en VHDL de l'architecture du circuit logique d'implémentation pour déduire un fonctionnement correct.

Cette simulation réalisée sur des types différents de circuits programmables FPGA sera suivie d'une interprétation. Des tests de simulation de synthèse vont être effectués sur la plate forme ISE 6.1 qui ciblera une implémentation matérielle du circuit FPGA, de type Xilinx Virtex II XC2V8000 (voir AnnexeB). Ce circuit peut fournir une architecture matérielle programmable pour implémenter 8 millions de portes logiques, et un grand nombre de circuits mémoires et de circuits d'interconnexions.

5.2 Etat de l'art de la simulation de l'architecture d'implémentation

La procédure de simulation permettra de vérifier le fonctionnement de l'architecture du circuit d'implémentation qui est en adéquation avec les calculs d'optimisation de l'algorithme. En effet, l'évaluation des erreurs produites dans la conception optimisée permettra de revoir la conception de ces calculs. L'outil de simulation est ModelSim 6.8, il est utilisé pour tester la description de l'architecture en langage VHDL.

Ceci permet de la vérifier en utilisant une simulation de la description en VHDL de chaque bloc du circuit logique d'implémentation. Les erreurs qui affectent les résultats de simulation sont dues principalement aux calculs d'optimisation effectués, aux erreurs produites par la numérisation des signaux et par l'approximation des fonctions non linéaires de l'algorithme du réseau de neurones. Les résultats de simulation obtenus sont d'une grande importance, car ils nous permettent de minimiser au fur et à mesure les erreurs introduites sur les résultats de la description en VHDL de l'architecture réalisée. La valeur du taux d'apprentissage sera fixée pendant l'étape de simulation. Concernant l'architecture d'implémentation, la dimension des nombres représentés en binaire pour chaque signal sera déterminée en fonction de la précision exigée dans le traitement des calculs de l'algorithme. Enfin, ils permettent le réajustement des valeurs des poids synaptiques selon les différents tests de simulation réalisés dans les blocs de l'architecture d'implémentation. Ceci consiste à effectuer par la suite, la synthèse de l'architecture, du placement et du routage sur le circuit pour la configuration des circuits d'interconnexions du circuit FPGA.

5.3 Conception sur circuit FPGA

Nous utilisons la plate forme de développement ISE 6.1 pour réaliser la conception hardware de l'architecture du circuit logique à implémenter sur le FPGA. La figure 5.1, nous montre le diagramme permettant de définir toutes les étapes de conception sur le circuit FPGA [9]. Ce diagramme représente la méthodologie à suivre pour la conception d'une implémentation hardware de l'algorithme ACI à élaborer.

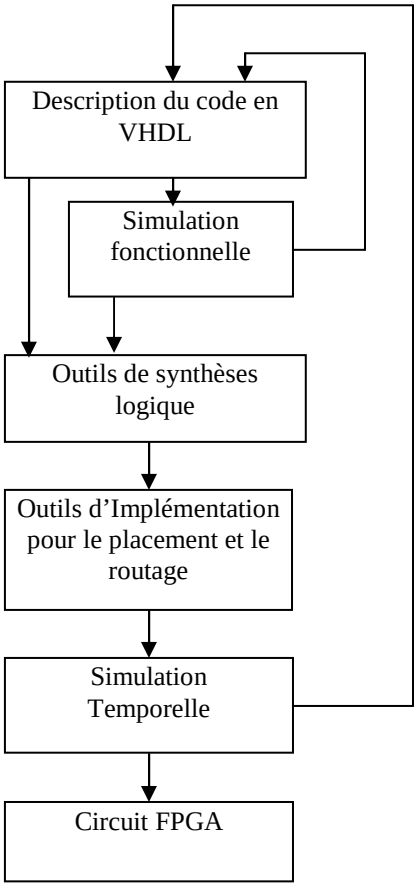


Figure 5.1 : Schéma de conception sur FPGA

5.4 Les résultats des calculs théoriques du circuit logique d'implémentation

Avant de décrire en VHDL l'architecture du circuit logique d'implémentation, on effectuera d'abord des calculs indispensables de sa simulation et de son implémentation. Ainsi, on effectue des calculs pour déterminer la valeur de ε, en virgule fixe, permettant de réaliser la convergence des poids synaptiques pour une précision donnée. Cependant, pour déterminer la fréquence d'échantillonnage des signaux d'entrées, on réalise une série de calculs sur l'architecture du circuit d'implémentation afin de cibler le traitement d'un nombre de neurones supérieure ou égale à deux. Pour une précision donnée, on détermine théoriquement la valeur de ε réalisant une convergence de l'ajustement des poids synaptiques selon la valeur de cette précision.

5.4.1 Calcul de la valeur ε d'ajustement de la convergence des poids synaptiques

La variation de l'ajustement des poids synaptiques est calculée selon la relation :

$$\Delta w_{\max} = \mu(1+\varepsilon), \text{ avec } \Delta w_{\max} < 1$$

Quand on représente la valeur de l'ajustement des poids synaptiques w sur 8 bits, l'erreur permis sera égale à ($1/2^8$=0,004). Alors, la variation maximale de l'ajustement des poids synaptiques sera égale à :

$$\Delta w_{\max} = 0{,}004, \quad \text{avec } \mu=0{,}004$$

5.4.2 Calcul de la valeur du nombre de traitement des cycles horloge

Le traitement du nombre des cycles d'horloge N_{cycle} est l'énumération de toutes les opérations des calculs de l'algorithme. Il est déterminé selon la fréquence d'une cadence de cycles fournis par une horloge.

$$N_{cycle} = M(c_{k\max}(N_{fncycle} + N_{pcycle}) + N_{fcycle})2^{N1} \tag{33}$$

où ; $N_{fncycle}$ est le nombre de cycle total des opérations de la fonction non linéaire d'apprentissage, N_{pcycle} est le nombre de cycle total, N_{fcycle} est le nombre de cycles total pendant une itération de l'ajustement des poids synaptiques finaux.
N1 est la valeur du nombre de traitement des neurones, M est le nombre d'échantillons des signaux d'entrées, 2^{N1} est la valeur du nombre des poids synaptiques ajustés, et c_{kmax} le nombre d'itérations maximales pendant l'ajustement des poids synaptiques.

5.4.3 Calcul de la valeur du temps de traitement d'apprentissage

Le calcul du nombre d'itérations maximales des poids synaptiques permet de calculer le temps de traitement maximal entre deux échantillons. La relation qui relie la valeur T_e du temps du traitement total d'apprentissage de l'ajustement des poids synaptiques et de la valeur de traitement du nombre de cycles d'horloge N_{cycle}, est donnée par l'équation suivante :

$$T_e = \frac{N_{cycle}}{f_{clock}} \tag{34}$$

où N_{cycle} désigne la valeur du nombre total de cycles d'horloge effectués pendant le traitement des échantillons de signaux d'entrées, dont la valeur dépend du nombre d'itérations au cours de l'ajustement des poids synaptiques, et f_{clock} la valeur de la fréquence d'horloge.

5.4.4 Calcul de la valeur de la fréquence d'échantillonnage

La fréquence f_e étant la fréquence d'échantillonnage qui peut être calculée selon l'équation suivante :

$$f_e = \frac{N_{ech}}{T_e} \tag{35}$$

où ; N_{ech} est le nombre total de traitement des échantillons des signaux d'entrées.
En remplaçant (34) dans (35), on peut tirer la valeur de la fréquence d'échantillonnage des échantillons des signaux d'entrée selon l'équation suivante :

$$f_e = \frac{N_{ech} f_{clock}}{N_{cycle}} \tag{36}$$

5.5 Résultats de simulation et de synthèse de l'architecture du circuit logique d'implémentation

Les résultats de simulation de l'architecture du circuit d'implémentation s'obtiennent par l'intermédiaire de l'outil ModelSim 6.7, sa synthèse est optimisée selon une technologie du circuit matériel cible de type Xilinx Virtex II XC2V8000. Le traitement de la simulation de l'architecture du circuit logique d'implémentation qui est réalisé suivant une hiérarchisation des blocs des circuits logiques d'implémentation, est illustré par les schémas de la Figure 5.2, la Figure 5 3, et la Figure 5.4.

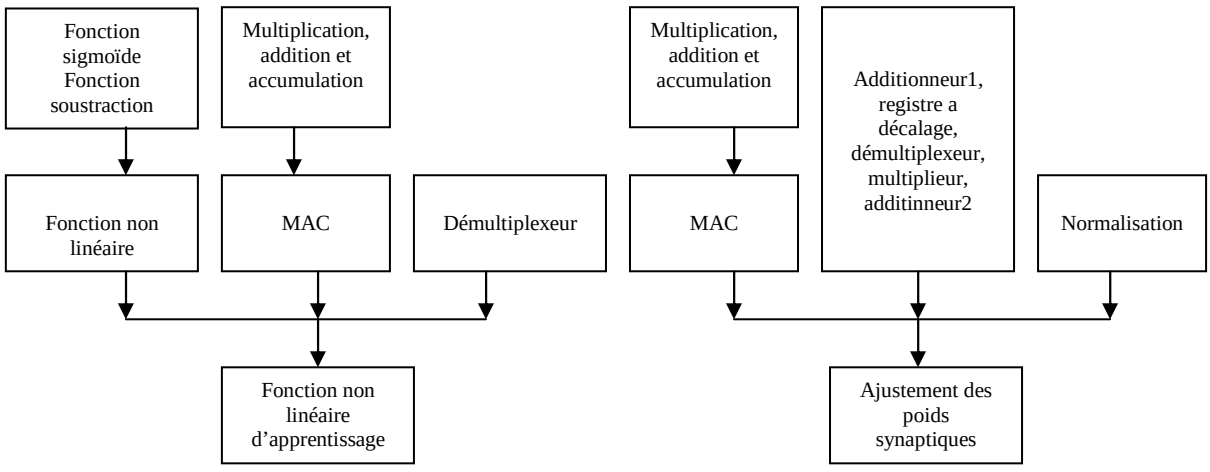


Figure 5.2 : Hiérarchie de simulation de la fonction non linéaire d'apprentissage et de l'ajustement des poids synaptiques

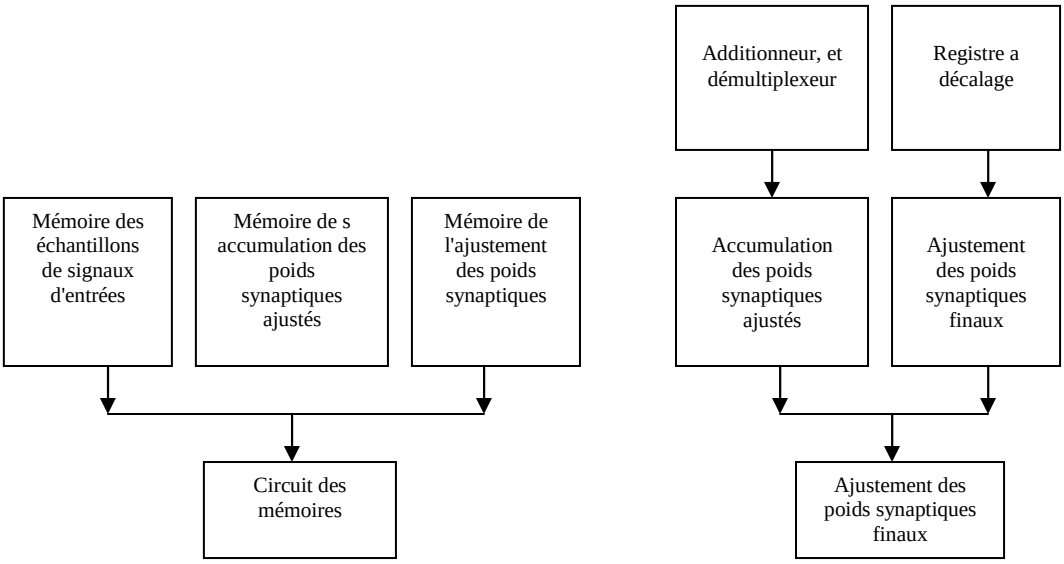


Fig.5.3 : Hiérarchie des circuits mémoires et de l'ajustement des poids synaptiques finaux

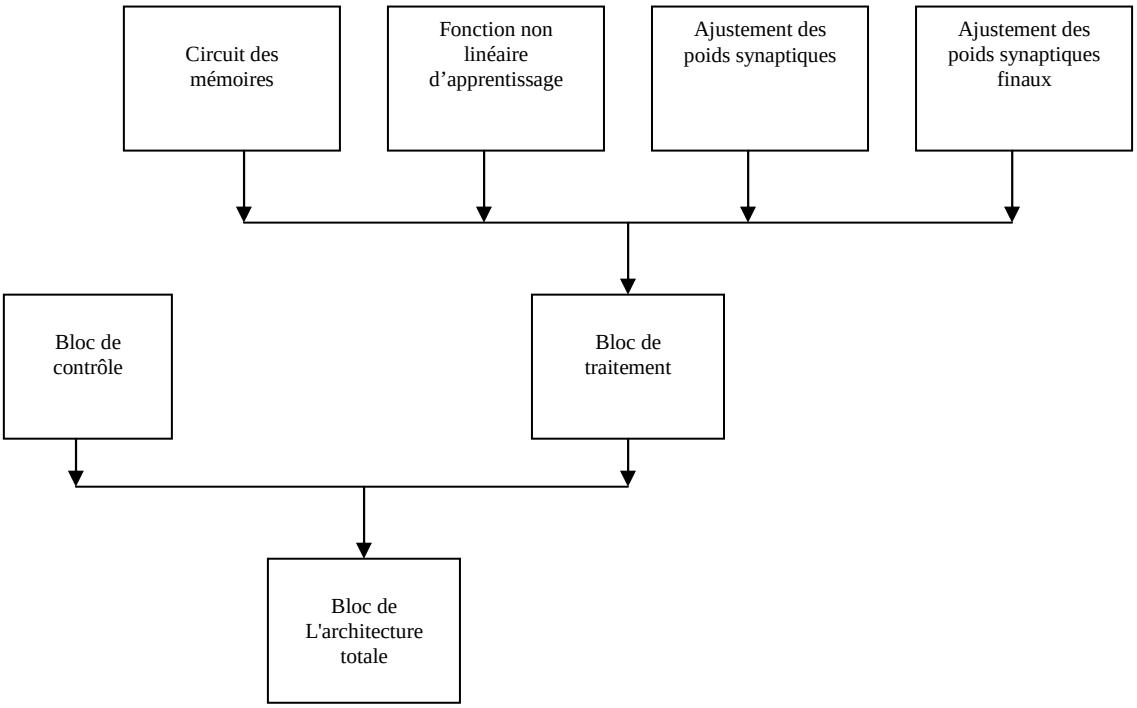


Figure 5.4 : Hiérarchie de l'architecture des circuits logique totale

5.5.1 Simulation de l'architecture du circuit d'implémentation de la fonction non linéaire d'apprentissage

La Figure 5.5, donne le schéma de simulation de la fonction sigmoïde du bloc de la fonction non linéaire d'apprentissage.

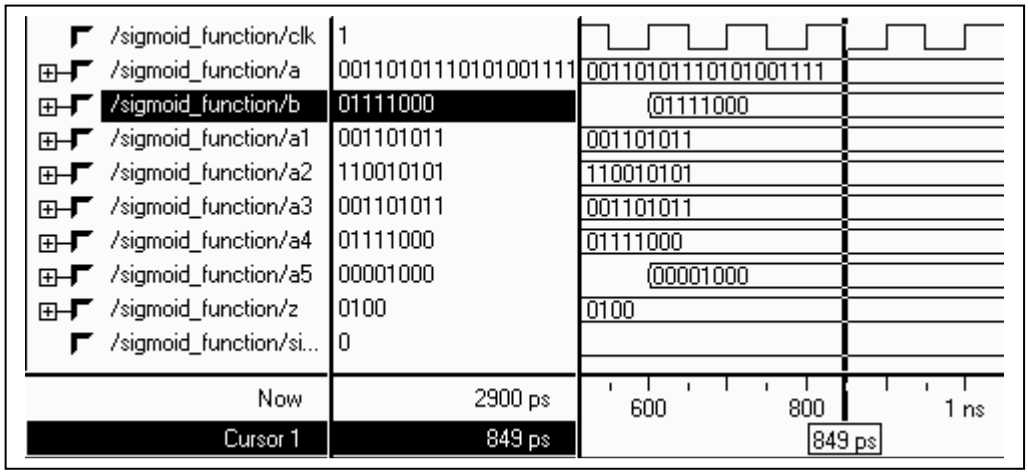


Figure 5.5 : Schéma de simulation de la fonction sigmoïde

On représente dans la Figure5.6, le schéma de simulation de l'architecture du circuit logique d'implémentation de la fonction non linéaire d'apprentissage.

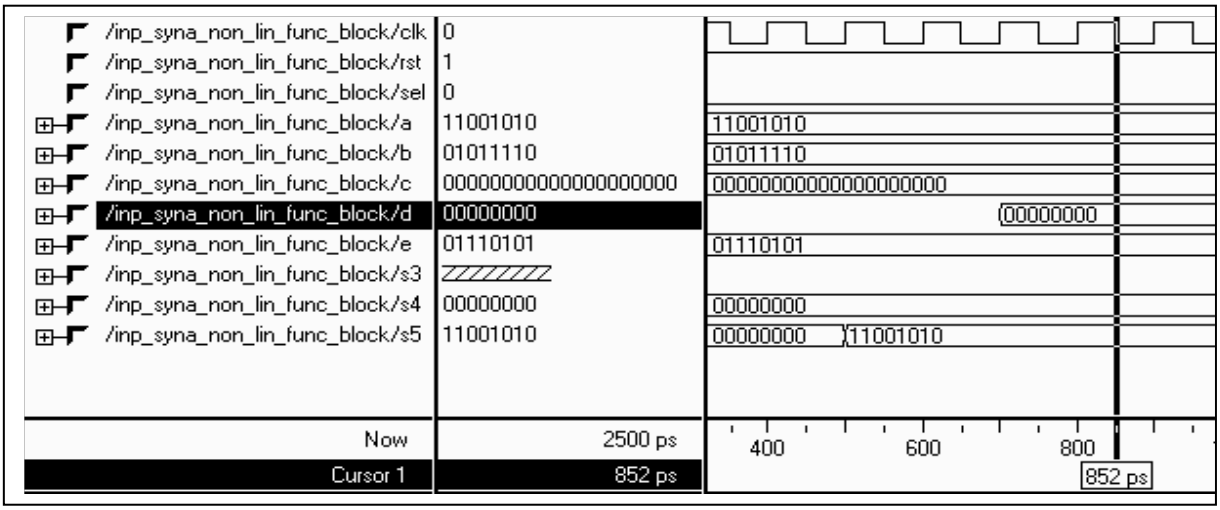


Figure 5.6 : Schéma de simulation de la fonction non linéaire d'apprentissage

5.5.2 Simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques

Le schéma de simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques est montré dans la Figure 5.7

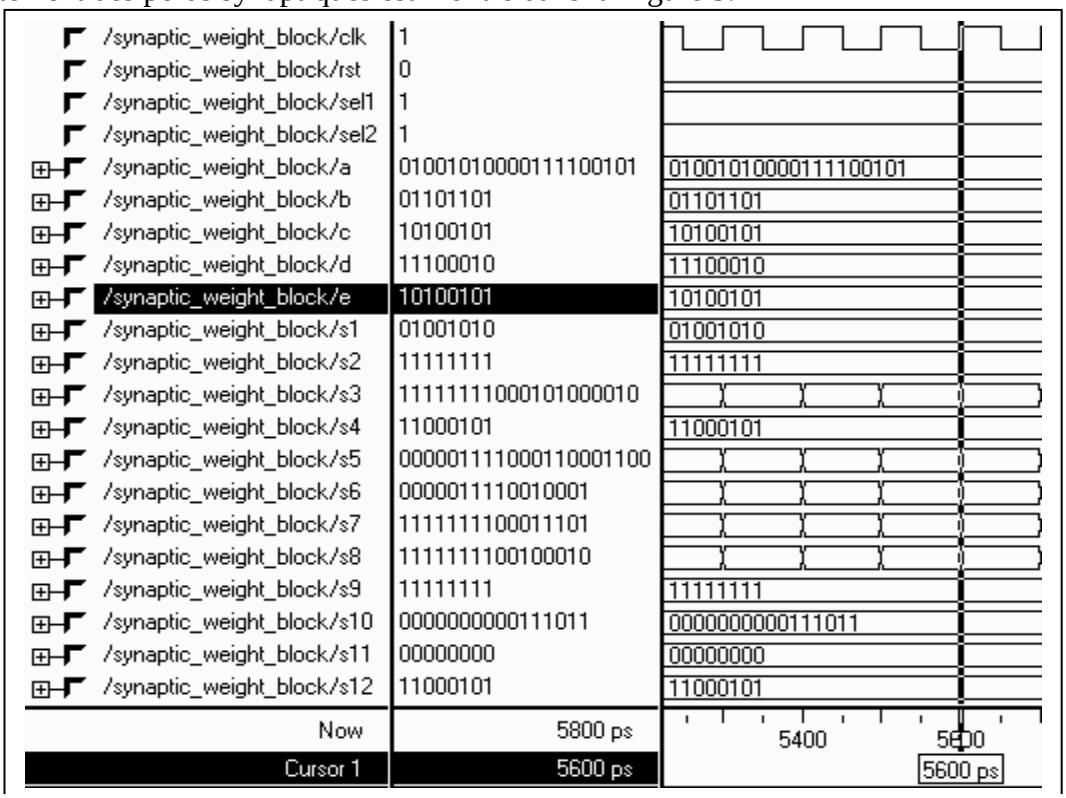


Figure 5.7 : Simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques

5.5.3 Simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques finaux

Dans la Figure 5.8, on donne le schéma de simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques finaux

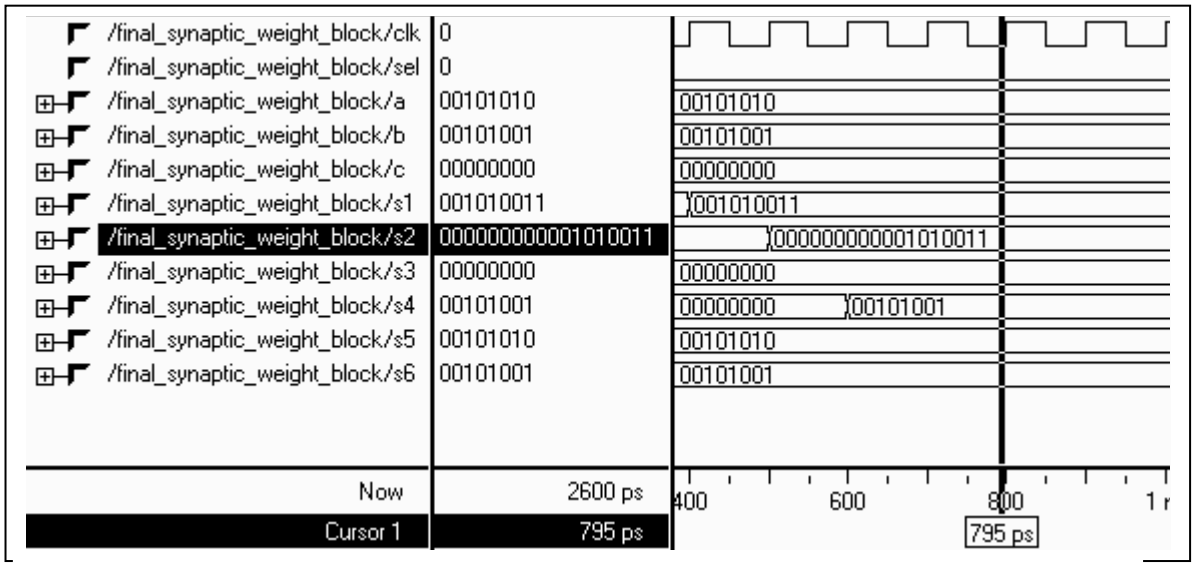


Figure 5.8. Simulation de l'architecture du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques finaux

5.5.4 Simulation de l'architecture du circuit logique d'implémentation des blocs mémoires

Le schéma de la Figure 5.9, représente les calculs de simulation de l'architecture du circuit logique d'implémentation des blocs mémoires.

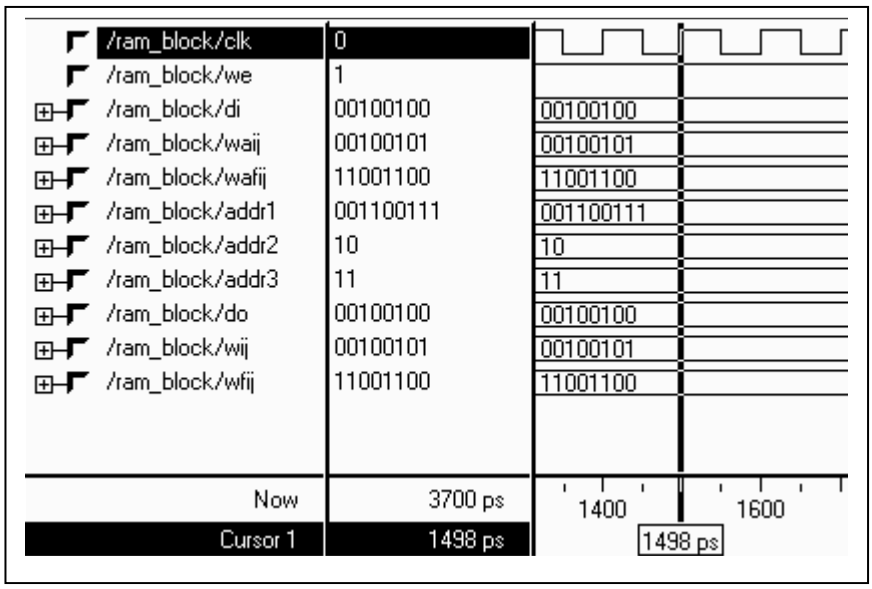


Figure 5.9 : Simulation de l'architecture du circuit logique d'implémentation des blocs mémoires

5.5.5 Synthèse de l'architecture du circuit d'implémentation du bloc de la fonction non linéaire d'apprentissage

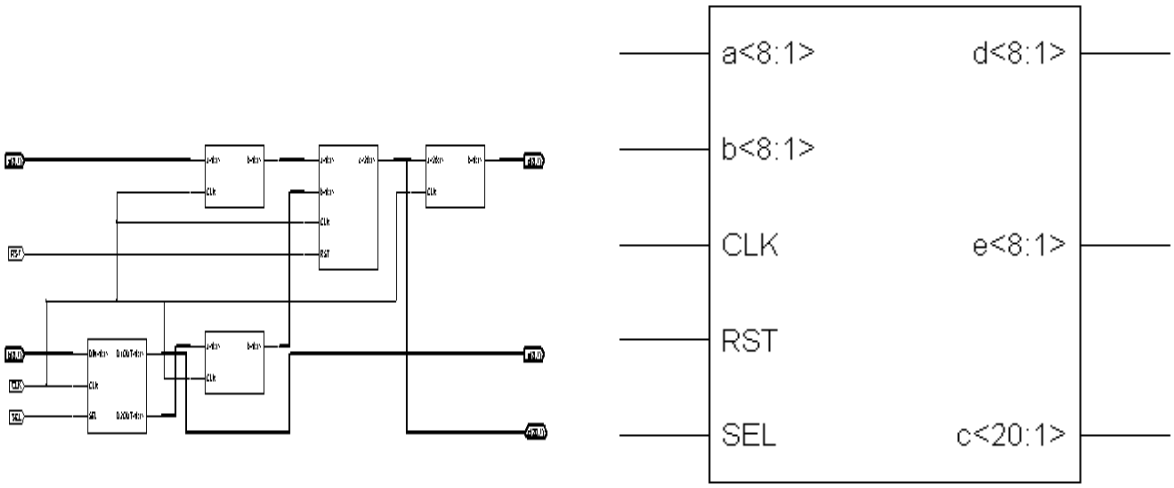
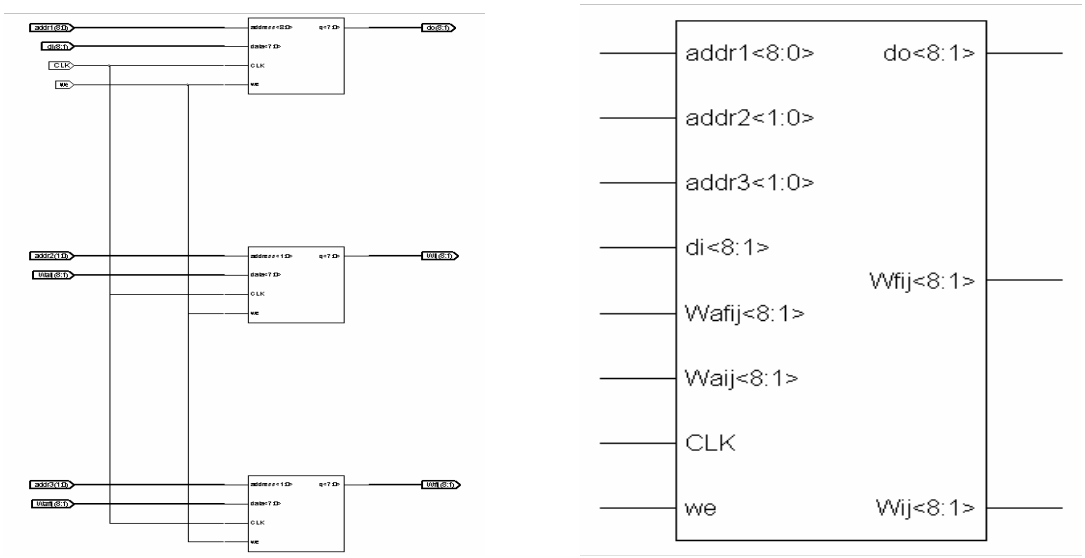


Figure 5.10 : Synthèse du circuit logique d'implémentation du bloc de la fonction non linéaire d'apprentissage

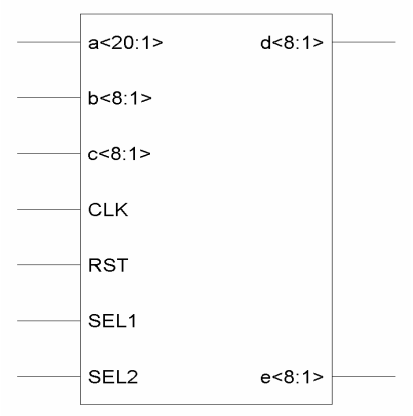
5.5.6 Synthèse de l'architecture du circuit d'implémentation du bloc des mémoires



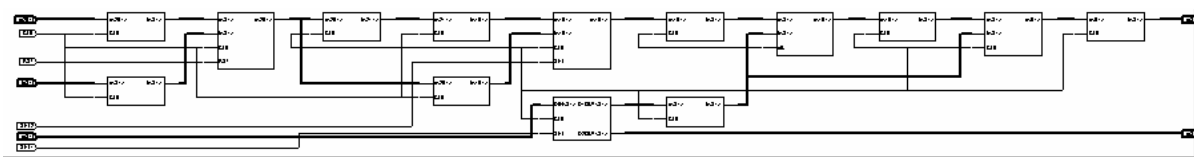
Architecture Composant

Figure 5.11. Synthèse du circuit logique d'implémentation du bloc des mémoires

5.5.7 Synthèse de l'architecture du circuit d'implémentation du bloc de l'ajustement des poids synaptiques



Composant



Architecture

Figure 5.12 : Synthèse du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques

5.5.8 Synthèse de l'architecture du circuit d'implémentation du bloc de l'ajustement des poids synaptiques finaux

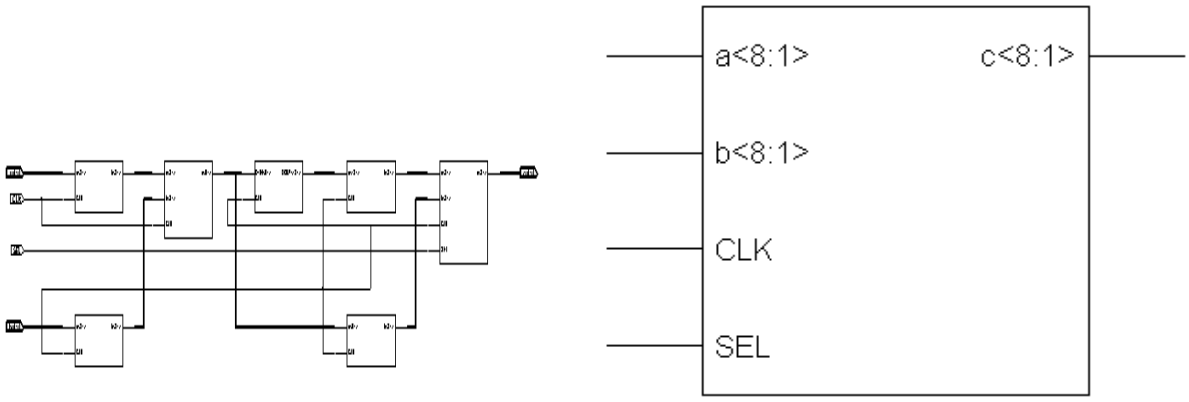


Figure 5.13 : Synthèse du circuit logique d'implémentation du bloc de l'ajustement des poids synaptiques finaux

5.6 Résultats de simulation de synthèse du circuit matériel

Le tableau 5.1 donne le résumé des résultats de simulation de synthèse du circuit Xilinx Virtex II XC2V8000.

Unite	Coût (slice)	coût (flip flops)	coût (Input LUT)	coût (IOB)	Temps (cycles)	Fréquence Max (MHz)
L'entrée synaptique et la fonction non linéaire dapprentissage	111	185	73	54	12	186.376
Poids synaptique	113	199	41	43	11	185.580

Tableau 5.1. Résultat de simulation de la synthèse du circuit Xilinx Virtex II XC2V8000

5.6.1 Résultats de synthèse du circuit matériel du bloc des mémoires

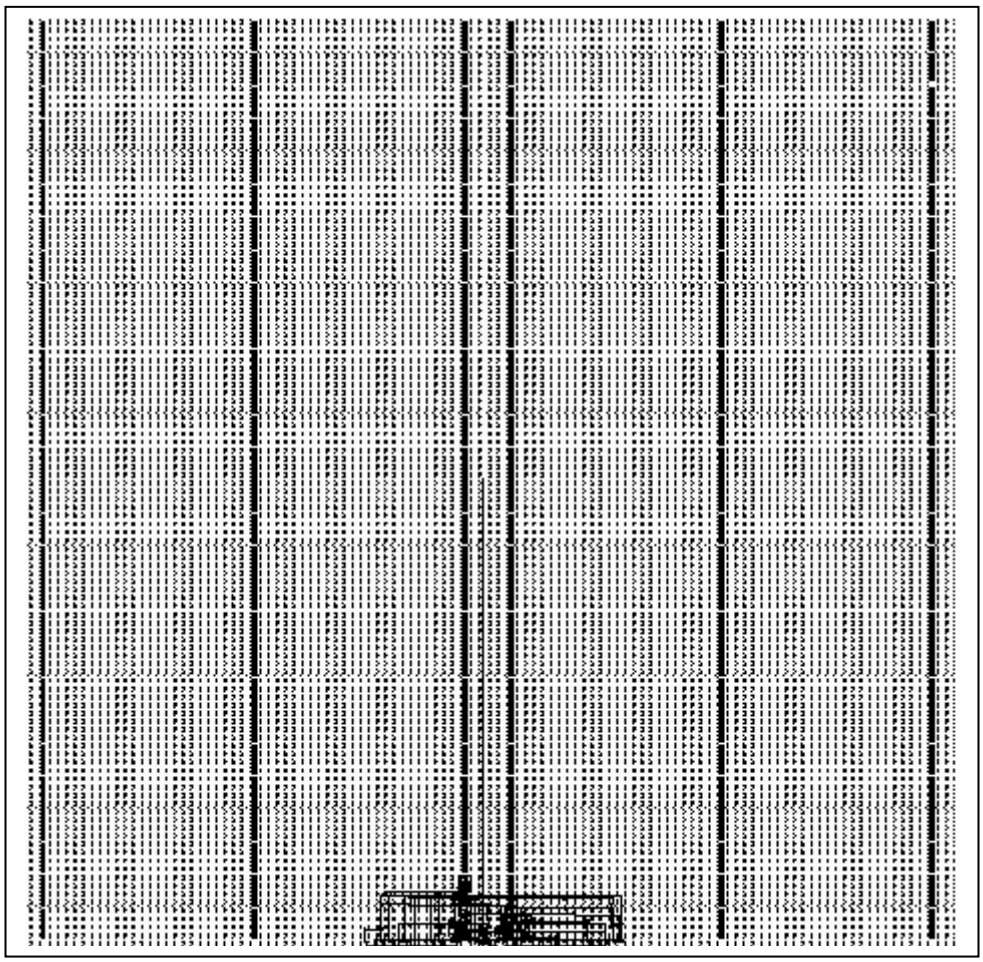


Figure 5.14 : Surface occupée par le bloc de mémoire RAM

Le schéma de la Figure 5.14, représente le placement et le routage du circuit de la fonction non linéaire d'apprentissage sur le circuit FPGA de type Virtex II XC2V8000 de Xilinx.

5.6.2 Résultats de synthèse du circuit matériel du bloc de la fonction non linéaire d'apprentissage

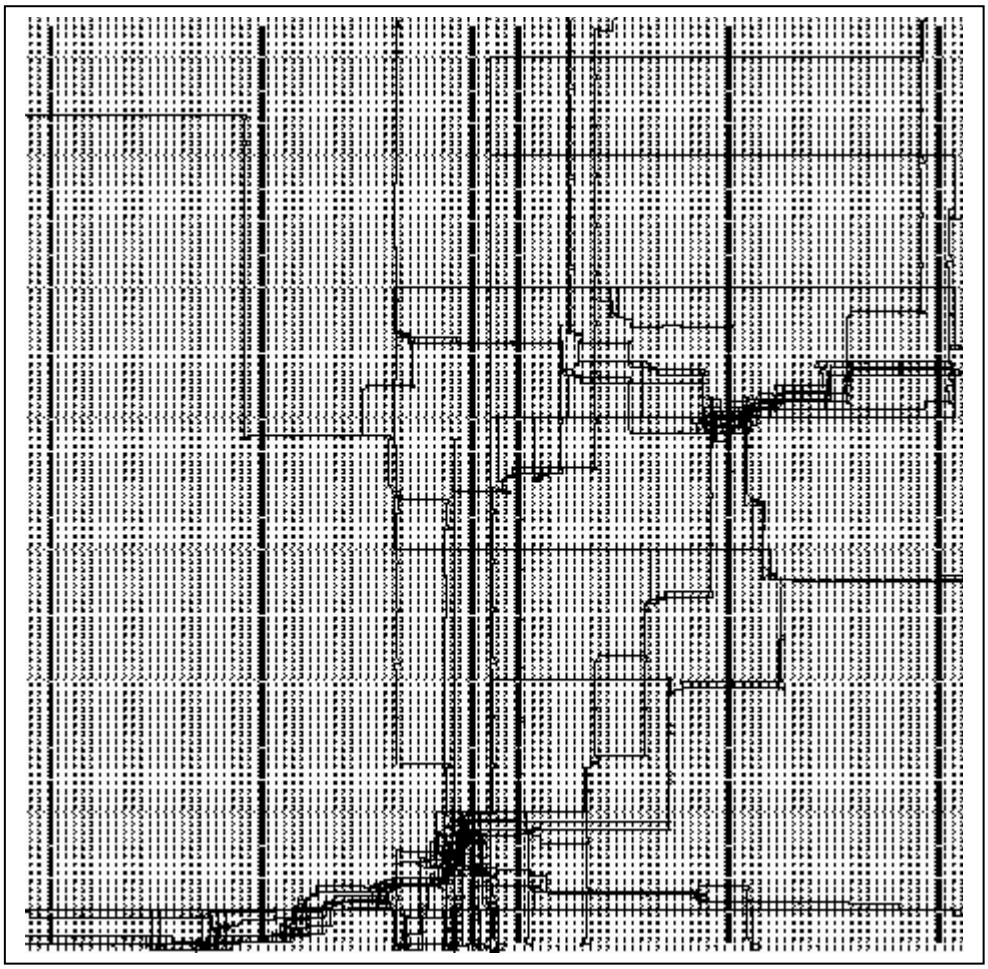


Figure 5.15 : Surface occupée par le bloc du circuit de la fonction non linéaire d’apprentissage

Le schéma de la Figure 5.15, représente le placement et le routage du circuit de la fonction non linéaire d’apprentissage sur le circuit FPGA de type Virtex II XC2V8000 de Xilinx.

5.6.3 Résultats de synthèse du circuit matériel du bloc de l'ajustement des poids synaptiques

Le schéma de la Figure 5.17, représente le placement et le routage du circuit de l'ajustement des poids synaptiques sur le circuit FPGA de type Virtex II XC2V8000 de Xilinx.

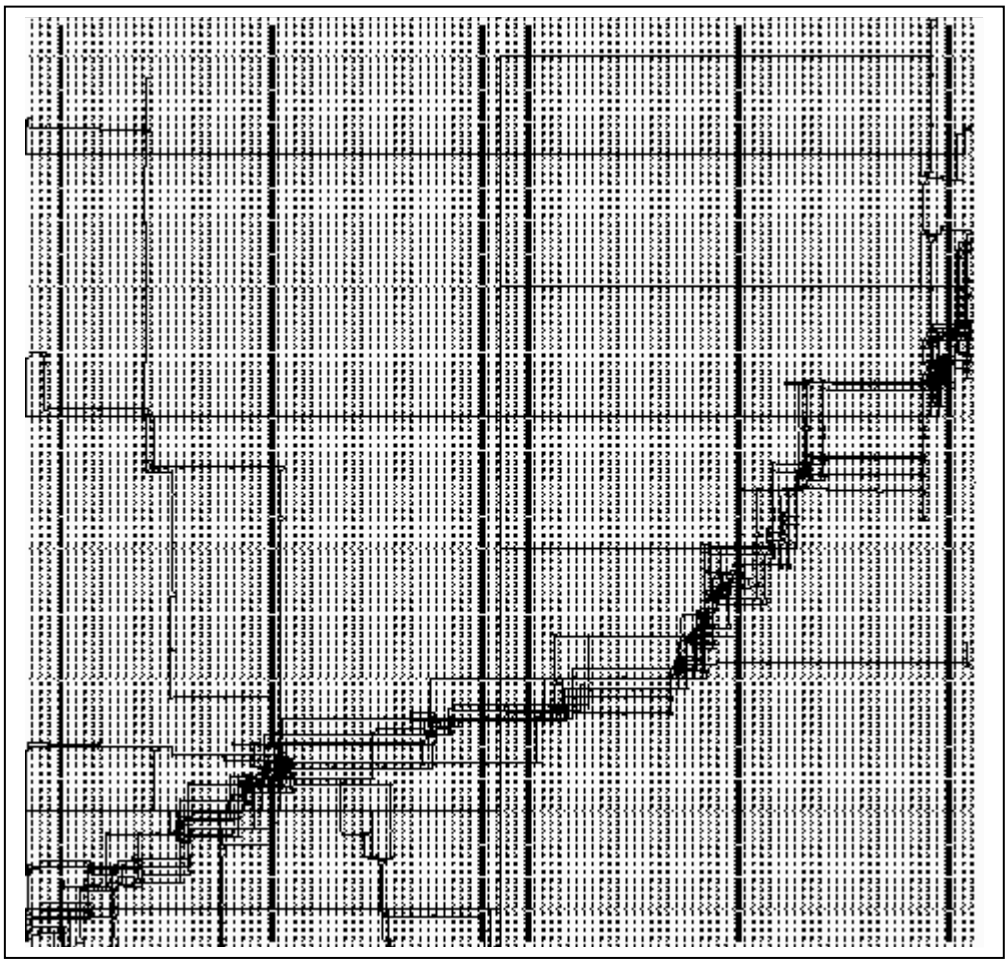


Figure 5.17 : Surface occupée par le bloc du circuit de l'ajustement des poids synaptiques

5.6.4 Résultats de synthèse du circuit matériel du bloc de l'ajustement des poids synaptiques finaux

Le schéma de la Figure 5.18, représente le placement et le routage du circuit de l'ajustement des poids synaptiques finaux sur le circuit FPGA de type Virtex II XC2V8000 de Xilinx.

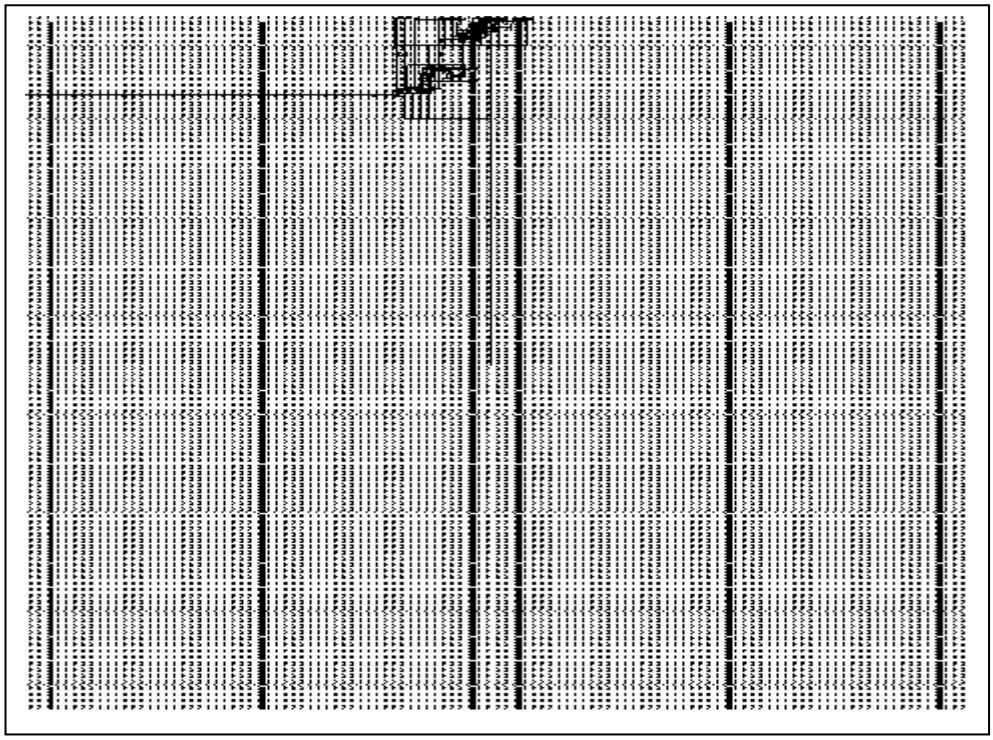


Figure 5.18 : Surface occupée par le bloc du circuit de l'ajustement des poids synaptiques finaux

5.7 Evaluation des résultats de simulation de synthèse du circuit matériel

Le tableau 5.2 montre les résultats de simulation de l'implémentation hardware sur le circuit FPGA pour d'autres travaux d'implémentation hardware qui utilise le traitement de séparation de signaux par l'algorithme ACI, [12-14] et [10]. Ces résultats de simulation matérielle ont été reportés pour permettre l'évaluation des performances des résultats de la conception du circuit matériel réalisée.

Cependant, le circuit logique d'implémentation du bloc de traitement a été décrit en VHDL dans un niveau de description structurel. D'autre part, nous avons décrit le circuit logique d'implémentation du bloc de contrôle en VHDL par un niveau de description comportemental qui s'adapte mieux aux traitements des circuits séquenceurs. D'ou, la description en VHDL de tous les blocs d'architecture d'implémentation qui a été simulée par l'outil ModelSim 6.8, synthétisés et implémentés sur circuit FPGA de type VirtexII XC2V8000 de Xilinx ont été comparés aux autres valeurs de performances des circuits matériels déjà réalisés pour voir les effets des résultats de performances sur les traitements en temps réel.

Ainsi, une simulation fonctionnelle a été effectuée sur l'architecture du circuit logique d'implémentation qui a permis de fournir des résultats en accord avec ceux effectués sur des architectures d'implémentation réalisées dans des conceptions précédentes de séparation de signaux. La valeur N=16 bits (dimension des nombres binaires dans la représentation à point fixe) a été prise pour les calculs dans la précision exigée. Le bit de poids fort a été assigné au bit signe tandis que les autres bits ont été réservés à la partie fractionnaire pour les valeurs qui ont été assignées aux échantillons des signaux d'entrées et aux signaux de l'ajustement des poids synaptiques.

En effet, le circuit matériel obtenu a permit le traitement de 2 signaux à l'entrée en séparant 2 signaux indépendants à la sortie. Pendant, la phase d'apprentissage, le processus de convergence, à utiliser le nombre environ de 100 itérations pour l'ajustement des poids synaptique qui a permit d'utiliser un nombre de l'ordre de 3220 cycles horloges pour le traitement du circuit matériel. D'où, la valeur du temps d'exécution du circuit matériel qui a été obtenu est de 17.3 µs pour le traitement de la valeur de 1000 échantillons pour chaque signal de l'entrée cadencé a un taux d'échantillonnage maximum avoisinant la valeur de 60 KHz.

Implementation	Circuit FPGA	Coût (slices)	Temps d'apprentissage (µs)	Nombre d'échantillons utilisés	Precision	Fréquence d'échantillonnage Des entrées (kHz)	Fréquence Max (MHz)
Implementation d'algorithme de séparation de signaux [12]	Xilinx Virtex-E family	–	–	6000	16 bits à point fixe	8	64.4
Implementation d'algorithme de pICA [13]	Xilinx Virtex V 1000 E	11318	–	6000	–	–	20.161
Implementation d'algorithme de FastICA [14]	Altera company	–	5.2	1000	32 bits à virgule flottante	192	50
Implementation de ICNN [10]	Xilinx Virtex XCV 812 E	12271	6.44	500	16 bits à point fixe	–	50
Implementation d'algorithme de notre conception	Xilinx Virtex II XC2V8000	1500	17.351	1000	16 bits à point fixe	57.53	185.580

Tableau 5.2**.** Résultat de simulation d'implémentation matériel sur le circuit FPGA

5.8 Conclusion

Les calculs d'optimisation effectués sur l'algorithme sont illustrés par plusieurs effets de constatation de traitement : le traitement de l'information ε, le traitement de normalisation des poids synaptiques ajustés, le traitement de l'accumulation de poids synaptiques ajustés, et le traitement de l'ajustement des poids synaptiques finaux. D'où, l'optimisation des calculs de l'algorithme (ACI) qui ont permit d'effectuer une méthodologie de conception hardware d'un seul neurone lors de la conception matérielle sur le circuit FPGA de type Virtex II XC2V8000 de Xilinx ont démontrer la validation des performances du circuit matériels en temps réel en utilisant un traitement de 2 signaux de 1000 échantillons a l'entrée pour séparer 2 signaux indépendants en sortie.

Conclusion

Dans ce travail, nous avons illustré une méthode de conception qui repose sur des calculs d'implémentation hardware principalement basés sur la recherche de la convergence des poids synaptiques en utilisant l'algorithme par l'analyse en composantes indépendantes (ACI) pour la séparation de signaux. A travers l'ajustement des poids synaptiques, nous avons réduit un nombre important de calculs utilisés par la mise en œuvre du traitement de la séparation de signaux.

Le point clé de notre approche était le développement d'une méthode de calcul à travers une méthodologie de conception visant à optimiser l'algorithme (ACI) pour permettre une architecture simple. En outre, l'architecture du circuit d'implémentation était limitée à travers la consommation de ressources matérielles, particulièrement, les circuits mémoires. Ainsi, nous avons utilisé une méthodologie de conception qui a pour objet la recherche d'une architecture d'implémentation d'un coût réduit pour diminuer la consommation de ressources matérielles dans le circuit d'implémentation hardware. A cet effet, nous avons utilisé une architecture qui consomme un nombre important de mémoires dans le circuit FPGA pour le traitement d'un nombre élevé d'échantillons des signaux d'entrées. On préserve ainsi, le nombre de ressources mémoires pour acquérir le maximum d'échantillons d'entrée et de ressources en portes logiques pour combiner l'augmentation des performances de la séparation de signaux et celles du circuit matériel d'implémentation. Par conséquent, le traitement séquentiel des neurones et celui en série des échantillons d'entrée nous ont permis d'améliorer les performances du circuit matériel et de permettre le traitement en temps réel.

Le langage VHDL a été utilisé pour la conception de l'implémentation hardware associé à des outils de logiciels de CAO dans une plate forme de développement des circuits FPGA. Le code décrit en VHDL a utilisé une description structurelle pour l'implémentation du bloc de traitement. D'autre part le bloc de contrôle spécifié par l'implémentation des circuits séquenceurs a été décrit simplement par un code VHDL à l'aide d'une description comportementale.

Les résultats de simulation obtenus sur le circuit FPGA de Xilinxs Virtex II XC2V8000, ont été comparés à ceux issus des conceptions d'implémentation hardware effectués sur des circuits FPGA par l'ACI pour la séparation de signaux. Ces résultats ont démontré une performance améliorée. Ainsi, la fréquence maximale d'échantillonnage qui avait atteint la valeur de 60 KHz avec un temps d'exécution de 17,3 ms permet un traitement de 1000 échantillons à 2 signaux d'entrée pour séparer 2 signaux de sorties.

Perspectives

Les perspectives de nos travaux de recherche porteront sur l'amélioration du temps d'exécution du traitement du circuit FPGA pour atténuer l'effet de latence des opérations séquentielles du traitement des neurones. En outre, les traitements de calculs parallèles indépendants accélèrent le temps d'exécution du circuit matériel. Pour cela, nous allons aussi effectuer une conception pour améliorer le bloc de contrôle permettant de réaliser des traitements pipelines du bloc de traitement.

D'autre part, nous procèderons à l'acquisition d'un nombre plus important d'échantillons des signaux d'entrées afin d'améliorer la qualité de traitement de l'algorithme en augmentant la fréquence des échantillons. Ainsi, ce type de traitements matériels pourront effectuer des applications de séparation de signaux qui exigent plus de performances dans le traitement en temps réel. Par conséquent, la conception de l'implémentation hardware qui permet d'augmenter les traitements parallèles dans une architecture d'implémentation présente un grand défit pour la réduction du temps d'exécution du traitement matériel dans le circuit FPGA.

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.M. Omondi and J.C. Rajapakse, ″Neural Networks in FPGAs″, Proceedings of the 9th International Conference on Neural Information Processing (ICONNIP'02) , 2002, vol.2, pp. 954-959.

[7] Jihong. Liu and Deqin liang , *″A Servey of FPGA-Based Hardware Implementation of ANNs″*, International Conference on Neural Networks and Brain (ICNN&B'05) , 2005 , vol.2, pp.915-918.

[8] Andre L.S. Braga , Carlos H.Lianos, Mauricio Ayala-Rincon and Ricardo P. Jacobi , *″VANNGen : a Flexible CAD Tool for Hardware Implementation of Artificial Neural Networks″*, Proceeding of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2005) , 2005 , 8pp.

[9] Laurent DUTRYEUX et Didier DEMIGNY, "*Logique programmable: Architecture des FPGA et CPLD, Méthodes de conception, Le langage VHDL*", Edition Eyroles, 1997.

[10] 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.

[11] 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.

[12] 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.

[13] H.T.Du and H.R.Qi, "*An FPGA Implementation of Parallel ICA for Dimensionality Reduction in Hyperspectral Images*", Geoscience and Remote Sensing Symposium, 2004, IGARSS'04. Proceedings, 2004 IEEE International, Vol.5, 3257-3260, 2004.

[14] Kuo-Kai Shyu and Ming Huan Li, "*FPGA Implementation of FastICA based on Floating-Point Arithmetic Design for Real-Time Blind Source Separation*", in Proc.2006 International Joint Conference on Neural Networks , July, 2785-2792, 2006.

[15] 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.

[16] K. Basterretxea , J.M Tarela and I. Delcomp ,*"Approximation of Sigmoid Function and the Derivative for Hardware Implementation of Artificial Neurons"*, IEE Proceedings , Circuits Devices and Systems , 2004 , vol.151, n° 1, pp.18-24.

[17] Mouloud OUNAS, Rachida TOUHAMI, and Salim CHITROUB, "Design *of Hardware Implementation Based ICA training algorithm for Blind Signal Separation"*, in proc. of Information and Communication Technologies International Symposium, ICTIS'07, April 3- 4-5, Fes, Morocco, 2007.

[18] Mouloud OUNAS, Rachida TOUHAMI, and Salim CHITROUB, "*Architecture Design of Digital Circuit based ICA training algorithm for Hardware Implementation"*, CGE'05, 16-17 avril 2007, EMP Bordj-El-Bahri, ALGER.

[19] Mouloud OUNAS, Rachida TOUHAMI, and Mustapha C.E YAGOUB, "*Low Cost Architecture of Digital Circuit for FPGA Implementation Based ICA training algorithm of Blind Signal Separation"*, Accepted paper *in* Proc. of International Symposium on Signals, Systems, and Electronics, ISSSE 2007, Montréal, Québec, Canada, 2007.

[20] Mouloud OUNAS, Salim CHITROUB, et Rachida TOUHAMI, "*Méthodologie de Conception d'une implémentation Matérielle d'un algorithme par l'Analyse en Composantes Indépendantes pour la séparation de signaux*", Article accepté, au 21ème Colloque International du Traitement du signal et des Images, GRETSI 2007, Troyes, France, 2007.

[21] M. Ounas, and R. Touhami, "*Design Methodology of Hardware Implementation Based ICA training algorithm for Blind Signal Separation"***,** Instrumentation Laboratory Workshop, Journées du Laboratoire d'Instrumentation, JLINS'07, Université USTHB, Alger, Algérie, 30 et 31 Octobre 2007.

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

[23] S.Amari, A.Cichocki, and H.H. Yang, "*Anew learning algorithm for blind signal separation*", Advances in Neural Information Processing Systems, pp. 757–763 Cambridge, MIT Press, 1996.

[24] A. Belouchrani, K.Abed-Meraim, J.F. Cardoso, and E.Moulines, "*A blind source separation technique using second-order statistics*", IEEE Transactions on Signal Processing, vol. 45, pp. 434–444, 1997.

[25] K.Matsuoka, M. Ohya, and M. Kawamoto, "*A neural net for blind separation of nonstationary signal sources"*, Neural Networks, vol. 8, no. 3, pp. 411–419, 1995.

[26] P.Comon, "*Independent component analysis: Anew concept?* ", Signal Processing, vol. 36, pp. 287–314, 1994.

[27] A. Hyvärinen and E. Oja, "*A fast fixed-point algorithm for independent component analysis*", Neural Computation, vol. 9, no. 7, pp. 1483–1492, 1997.

[28] J. Herault and C. Jutten, "*Space or time adaptive signal processing by neural network models*", In Neural Networks for Computing: AIP Conference Proceeding 251, J. S. Denker, ed. American Institute for Physics, New York, 1986.

[29] Dominique HOUZET, "*Conception de circuits en VHDL: Principes et Méthodologie*", CEPADUE-EDITION, 2000.

Annexe A

1. Les solutions ASICs :

Les circuits intégrés à applications spécifiques (ASIC) regroupent 2 catégories principales :

- les circuits semi spécifiques, pour lesquels une partie ou l'ensemble des étapes de fabrication ont été réalisées avant la conception,
- les circuits spécifiques, pour lesquels l'ensemble des étapes de fabrication seront réalisées après la conception.

1.1 Circuits semi-spécifiques

On distinguera à ce niveau deux sous catégories suivant le type de personnalisation :

- les circuits programmables – qui pourraient être aussi classés en catégorie standard, où l'ensemble des étapes de fabrication sont réalisées par le fabricant. Le concepteur fera lui même selon ses besoins la personnalisation du circuit, de manière réversible ou irréversible suivant les technologies, à partir d'outils spécifiques (logiciels, programmateur). Ils constituent une solution adaptée aux développement de prototypes, aux petites et moyennes séries d'équipements. Ils ont pour avantage d'être disponibles avant la conception, et pour inconvénients des performances en densité d'intégration et en vitesse limitées par leurs architectures et leurs technologies, ainsi que des prix unitaires élevés pour des grandes séries d'équipement (surface de silicium importante par rapport à des circuits spécifiques).
- les circuits pré diffusés (gate array), où les premières étapes de fabrication sont réalisées, préalablement à la conception. L'utilisateur va personnaliser une matrice standard caractérisée par un nombre de transistors et de plots d'entrée/sortie limité, en concevant les masques finaux de fabrication (contacts, métallisations) de manière automatique à l'aide d'une bibliothèque de cellules. Les réseaux de transistors vont être ainsi programmés et interconnectés par les niveaux de métallisation. Les étapes finales de fabrication de la puce et son encapsulation seront ensuite réalisées par le fabricant, dans des délais réduits par rapport à un cycle de fabrication complet. La topologie de ces circuits correspond à des bandes de cellules séparées par des canaux de routage figés, ou à des réseaux de transistors (sea of gates) qui permettent une meilleure densité d'intégration en optimisant les zones de routage des interconnexions à réaliser. Les limitations de cette approche sont liées au manque de flexibilité (par

rapport à l'approche présentée ci-dessous) puisque la topologie (nombre de portes, plots) est figée. Les densités d'intégration obtenues sont ; à technologie équivalente, et meilleure que pour les circuits programmables.

1.2 Circuits spécifiques

On distinguera à ce niveau, deux sous catégories suivant la méthode de conception utilisée :

- les circuits pré caractérisés (standard cell ou cell based) lorsqu'une bibliothèque de cellules est utilisée. Une bibliothèque est constituée de cellules élémentaires plus ou moins complexes prédéfinis dans des bases de données informatiques. Les caractéristiques de ces cellules sont connues (pré caractérisation par le fabricant à l'aide de simulations électriques). Cette notion de réutilisation du matériel se justifie d'une part en raison de la stabilité des procédés de fabrication et d'autre part par l'existence de moyens de CAO qui permettent une meilleure efficacité de la conception en automatisant certaines étapes (synthèse logique et topologique). La topologie de ces circuits correspond à des bandes de cellules séparées par des canaux de routage de largeurs variables (en fonction des contraintes d'interconnexion du circuit).
- les circuits sur mesure (custom ou full custom) lorsque toutes les étapes de conception électriques et topologiques (layout) sont réalisées. Chaque élément actif du circuit est optimisé pour réaliser la meilleure performance fonctionnelle et électrique et réduire la surface du silicium.

2. Les circuits programmables

Les architectures de ces circuits offrent un ensemble de ressources logiques (portes, bascules, …etc.) qui peuvent être interconnectées de différentes façons : réalisation de fonctions booléennes sous forme d'une somme limitée de monômes (circuits PAL, PLD), ou d'un réseau de cellules (FPGA).

Les FPGAs se traduit en Français par circuit pré diffusé programmable. Contrairement aux circuits pré diffusé conventionnels, les circuits pré diffusé programmable ne demande pas de fabrication spéciale en usine, ni de système de développement coûteux. Les circuits FPGA qui appartient à la famille des ASICs, se situe entre les réseaux logiques programmables et les prés diffusés. C'est donc un composant standard, combinant la densité et les performances d'un pré diffusé avec la souplesse due a la programmation des PLDs. Cette configuration évite le passage chez le fondeur et tous les inconvénients qui en découle.

2.1 Description d'un FPGA :

Le FPGA fournit le réseau de portes comme architecture, il forme une matrice carré (NxN) arrangé d'une manière part culière. La Figure A.1, présente l'architecture interne du FPGA dont il dispose un périmètre des blocs d'entrées sortie (IOB BLOCK) et des matrices d'interconnexions programmable (PROGRAMMABLE interconnect). Chaque bloc est constitué de quelques éléments pour réaliser une tache bien spécifiée

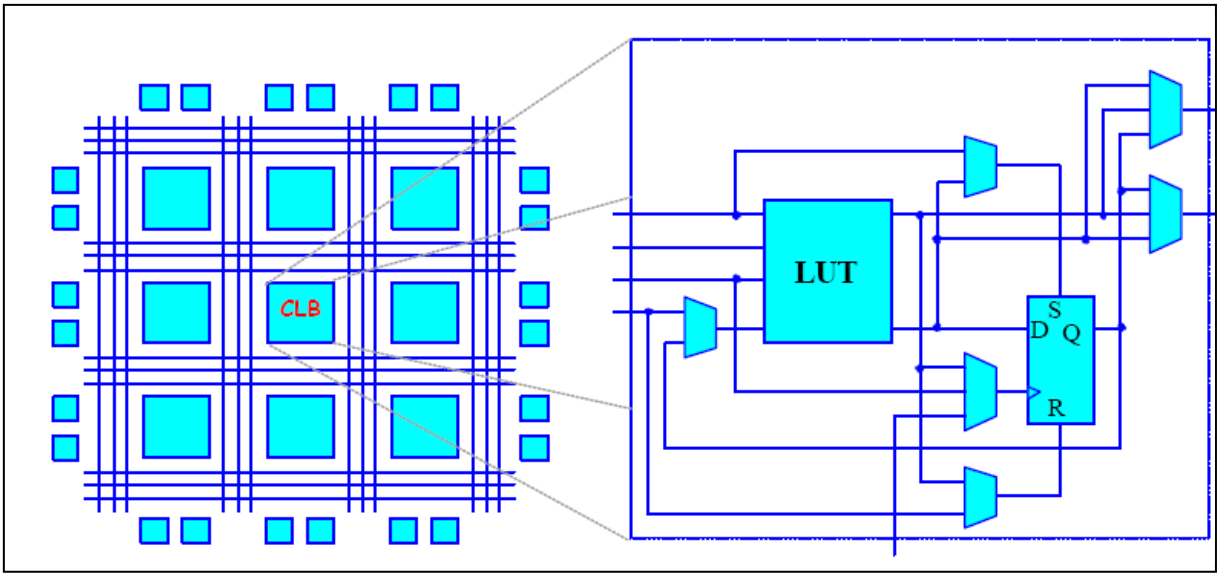


Figure A.1 : Architecture interne d'un FPGA

2.1.1 Bloc logique configurable (CLB) : c'est le cœur du circuit FPGA, il décrit la fonction qui lui est assignée (voir Figure A.2), il est constituer de :

- Générateur de fonction combinatoire
- Bascule flip-flops
- Multiplexeur de signaux de contrôle
- Ligne d'horloge

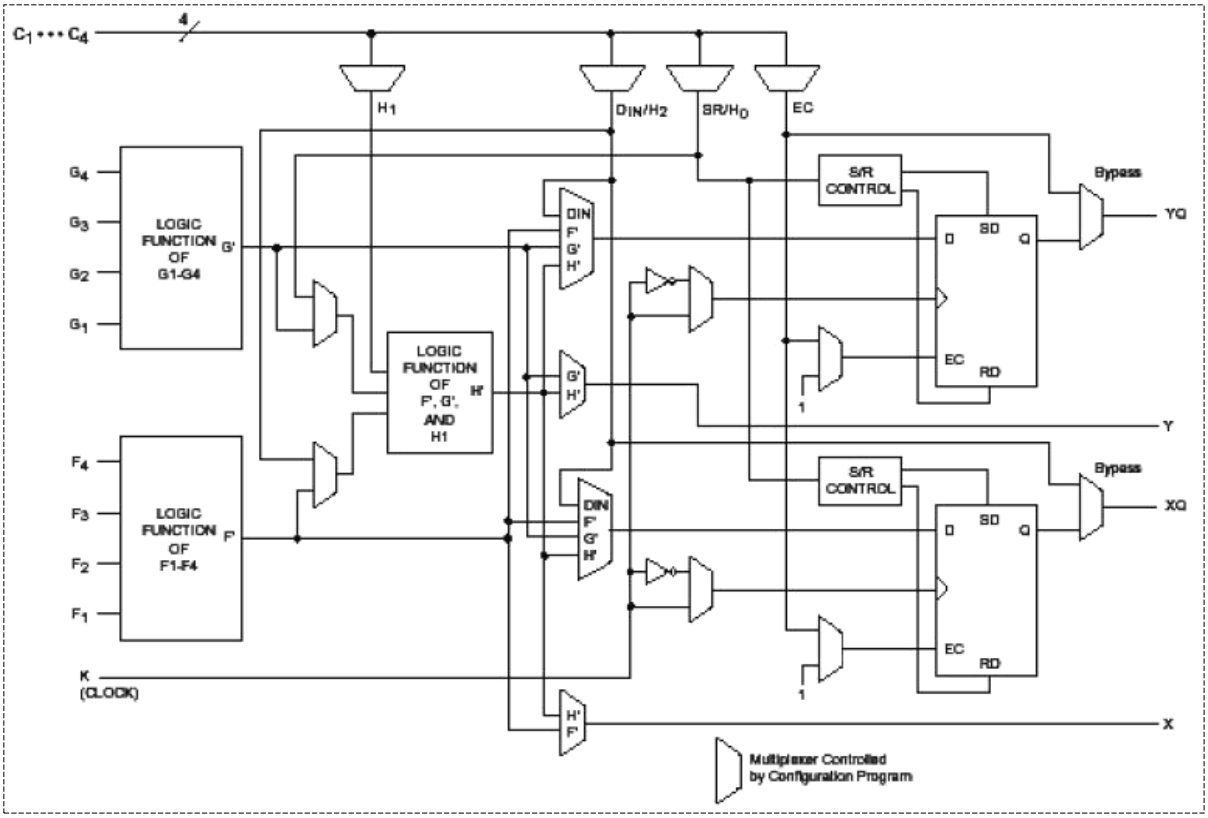


Figure A.3 : Bloc logique configurable (CLB)

2.1.2 Bloc d'entrés/sortie (IOB) : c'est le contour du circuit FPGA ou l'interface entre broche du composant FPGA et la logique interne développée a l'intérieur du circuit, il contrôle une broche du composant, pouvant ainsi la configurer en entrée, en sortie, en bidirectionnel ou en

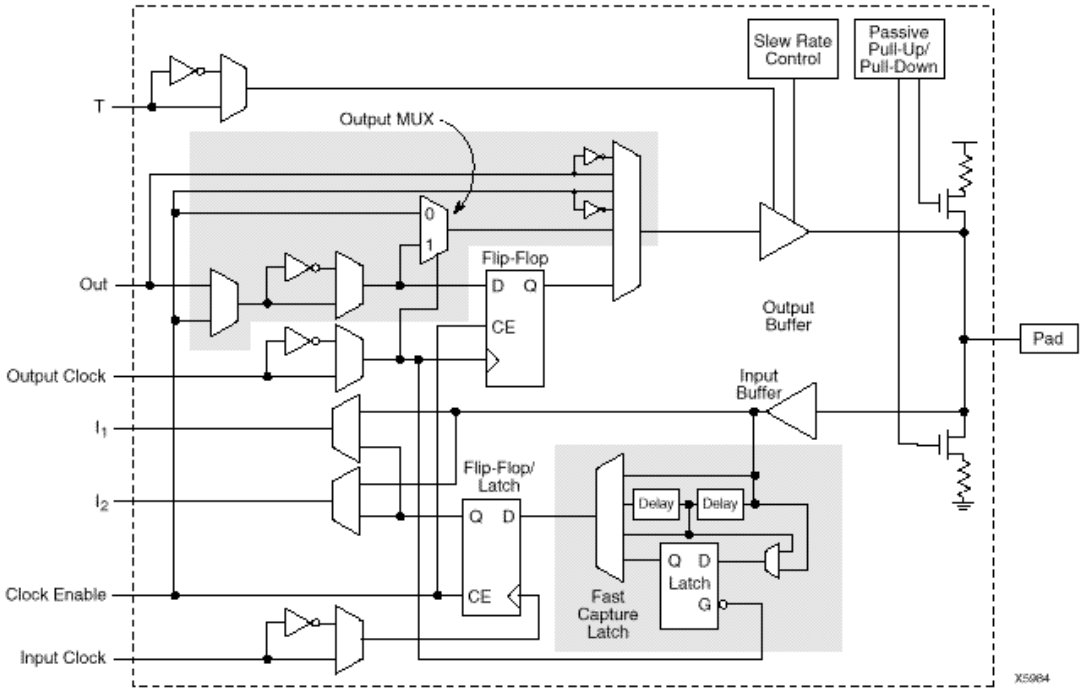


Figure A.4 : Bloc d'entrée/sortie (IOB)

2.1.3 Ligne d'interconnexions : c'est un segment de droites parallèle horizontal et perpendiculaire avec des points d'interconnexions programmables pour permettre en application le routage désiré :

- Le routage CLB associé a chaque ligne et colonne de l'assemblage CLB
- Le routage IOB connecte les entrées sorties avec les blocs logiques internes
- Le routage général qui affecte les horloges dans tout le FPGA d'une manière parfaite avec un minimum de délais

Selon la longueur de la destination des liaisons, on distingue trois sortes d'interconnexions

- Les interconnexions a usage générale : elles sont constituées d'une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes de CLB et de l'IOB (voir Figure A.5). Des aiguilleurs appelés aussi matrice de commutation sont situés à chaque intersection. Leur rôle est raccorder les segments entre eux selon diverses configurations, ils assurent ainsi la communication des signaux d'une voix sur l'autre. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre.

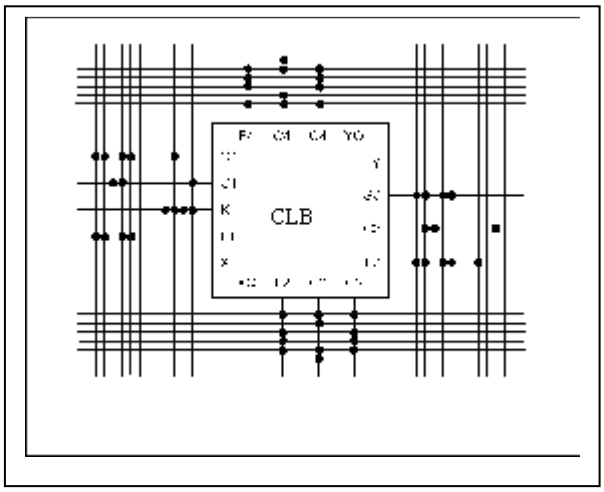


Figure A.6 : Les lignes d'interconnexions a usage général

Annexe B

1 Famille Virtex II

La famille Virtex II est le premier jeu de la plate forme FPGA, elle a été conçue pour réaliser des conceptions a faible ou grande densité d'intégration et exigent des performances élevées (elle peut atteindre jusqu'à 10 millions de portes logiques). La fréquence de son utilisation peut être portée 420 MHZ. Cette famille a onze membres (voir Tableau B.1) dans la densité de portes système s'étendant de 40Ko à 8M.

Circuit Virtex II	Nombre maximum de portes logique	Nombre de blocs logiques (CLB)	Capacité maximum de mémoire (BRAM)	Capacité Maximum de mémoire distribuée	Bloc des multiplieurs 18x18	DCMs	Nombre maximum d'entrées sorties (IOB)
XC2V8000	8M	104,832	3,024 Mbits	1,458 Mbits	160	12	1,108
XC2V6000	6M	76,032	2,592 Mbits	1,058 Mbits	144	12	1,104
XC2V4000	4M	51,840	2,060 Mbits	720 Kbits	120	12	912
XC2V3000	3M	32,256	1,726 Mbits	445 Kbits	96	12	720
XC2V2000	2M	24,192	1,008 Mbits	335 Kbits	56	8	624
XC2V1500	1,5M	17,280	864 Kbits	240 Kbits	48	8	528
XC2V1000	1M	11,530	720 Kbits	180 Kbits	40	8	432
XC2V500	500K	6,212	576 Kbits	96 Kbits	32	8	254
XC2V250	250K	3,456	432 Kbits	48 Kbits	24	8	200
XC2V80	80K	1,162	164 Kbits	16 Kbits	16	4	120
XC2V40	40K	596	172 Kbits	8 Kbits	8	4	58

Tableau B.1 : Membre de la famille Virtex II

1.1 Nomenclature d'un circuit Virtex II

Les circuits FPGAs de Xilinx sont caractérisés par une nomenclature spécifique qui définie les performances de chaque famille, cette nomenclature est illustrée dans la Figure B.1.

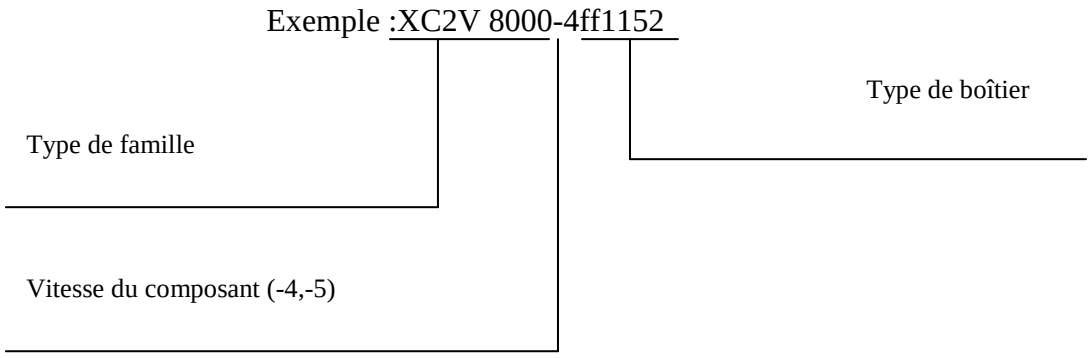


Figure B.1 : La nomenclature d'un circuit Virtex II

Ou :

XC : Xilinx Circuit.

2V : (Device type), il représente le type de famille, Virtex II.

8000 : représente le nombres de portes logique, c'est-à-dire 8M de portes logique.

4 : (speed grade), c'est la vitesse du composant suivant la technologie utilisée dans la fabrication du circuit.

ff1152 : (Package type), représente le type de boîtier.

1.2 Architecture du circuit Virtex II

L'architecture du circuit Virtex II est montrée sur la Figure B.2, elle est constituée de quatre blocs principaux qui sont :

- Bloc logique configurable (CLB)
- Bloc mémoire a 18Kbits (BLOC Select RAM).
- Bloc Multiplieur 18x18 (Multiplier).
- Bloc directeur de pendule a lecture digitale (DCM).

La connexion entre ces blocs est assurée d'une manière programmable grâce à des ressources d'interconnexions configurable GRM ‘’General Routing Matrix ‘’.

1.2.1 Les CLB (Configurable Logic Block)

Les CLBs sont implémentés sous une matrice (NxM) sur un circuit FPGA, ils incluent toutes la logique nécessaire pour la conception des circuits combinatoires et séquentiels.

Chaque CLB est composé de quatre slices (voir Figure B.3) et trois portes à trois états, chaque slice contient :

- Deux générateurs de fonction (F et G).
- Deux éléments de stockage.
- Des portes arithmétiques et logiques
- Une large capacité de fonction
- Une chaîne en cascade horizontale (porte OU).

Les générateurs de fonctions F et G sont configurables comme des LUTs à quatre entrées, comme des registres à décalage ou comme des mémoires SRAM.

1.2.2 Les IOBs (Input/output bloc)

Les blocs (IOB) fournissent une interface entre les broches externes du circuit et la logique interne. Deux IOBs peuvent être comme une paire différentielles. Celle-ci est souvent connectée a une matrice d'interconnexions.

1.2.3 Le Multiplieur 18x18 bits

La famille Virtex II incorpore des blocs multiplieurs 18x18 bits. L'implémentation d'un multiplieur sur cette famille peut être effectuée selon deus manières, la première est l'utilisation directe des blocs multiplieurs, la seconde est la configuration des CLB en multiplieur. Cette tache est réalisée grâce a un outil nommé Corgen IP qui nous permet de générer des modules pré optimisés. Celle-ci est jugée moins efficace par rapport à la première, vue que l'utilisation des CLB présente une consommation d'énergie.

La disposition des blocs multiplieurs se trouve implémentée juste à coté blocs Select RAM. Cette technique permet d'augmenter les performances d'une application si les opérandes proviennent d'une mémoire.

Ces blocs multiplieurs permettent d'effectuer une multiplication sur des opérandes signés ou non signés et avoir et avoir un résultat et avoir un résultat sue 36 bits, ils peuvent aussi avoir des entrées sorties régulières.

1.2.4 Le DCM (Digital Clock Manager)

Le Virtex-II inclut dans son architecture un nombre important de blocs mémoires de 18 K bits. Ceci augmente l'espace mémoires. Chaque Bloc Select RAM a double port est de taille

se 2x18 bits. Ces derniers peuvent avoir indépendamment les signaux d'horloges et de commandes. D'une manière générale les blocs Select RAM sont implémentés suivant les configurations suivantes :

- Lire après écriture (Read After Write).
- Lire avant écriture (Read Before Write).
- Lire seulement (Read Only).

Ces blocs mémoires peuvent être utilisés comme des RAM a simple ou double port, et ils supporte plusieurs configurations.

Ces configurations sont déterminées par un compromis entre la taille de la donnée et le nombre d'adresse ligne.

Liste des Abréviations

ACI: Analyse en Composantes Indépendantes.

ASIC: Application Specific Integrated Circuit.

AAA: Adéquation Algorithme et Architecture.

CAO: Conception Assisté par Ordinateur.

CLB : Configurable Logic Block.

CPLD: Complexe Programmable Logic Device.

DSP: Digital Signal Processor.

DCM : Digital Clock Manager.

DEMUX : Demultiplexeur.

FPGA : Field Programmable Gate Array.

IOB: Input Output Bloc

ISE: Integrated Software Engineering

LUT: Look Up Table.

MUL : Multiplieur.

MUX : Multiplexeur.

PAL : Programmable Array Logic

PLD: Programmable Logic Device.

RAM: Random Access Memory.

SRAM: Select Random Access Memory

VHDL: VHSIC Hardware Description Language.

VHSIC: Very High Speed Integrated Circuit.

VLSI: Very Large Scale Integration.