جامعة أبو بكر بلقايد – تلمسان

Université Abou Bakr Belkaïd de Tlemcen

Faculté de Technologie

Département de Génie électrique et Electronique

Laboratoire de Recherche de Génie Biomédical

MEMOIRE DE PROJET DE FIN D'ETUDES

pour obtenir le Diplôme de

MASTER en GENIE BIOMEDICAL

Spécialité : **Electronique Biomédicale**

présenté par : Fandi Tadj Eddine.

Simulation d'un classifieur neurenal sur FPGA

Soutenu le 30 juin 2013 devant le Jury

M.	Bechar Hacène	*MAA*	Université de Tlemcen	Président
M.	Chikh Med Amine	*Prof*	Université de Tlemcen	Encadreur
M.	Benhadada Omar	*MAA*	Université de Tlemcen	Examinateur
Mme	Baba Hamed Amel	*Doctorant*	Université de Tlemcen	Co-Encadreur

Année universitaire 2012-2013

Remerciement

Je tiens tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, je tiens à remercier mon encadreur Mr CHIKH M.A Professeur à l'Université de Tlemcen, de ses précieux conseils et son aide durant toute la période du travail.

Je tiens aussi a remercier Mme BABA HAMED .A doctorante à l'Université de Tlemcen, en tant que co-encadreuse pour sa patience et son soutien qui m'a été précieux afin de mener mon travail à bon port.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions:

Mes remerciements à Mr BECHAR. H maître de Conférences à l'Univérsité de Tlemcen, qui m'a fait l'honneur de présider le jury aussi de l'attention qu'il a porté à notre travail et ses judicieux conseils.

Nous exprimons notre haute considération au membre de jury Mr Benhadada .O *d'avoir accepté l'examinassions de notre travail.*

Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Table de matière.

Table de figures.

Table de tableaux.

Liste des acronyms

ASIC: Application-Specific Integrated Circuit.
CAO : Conception Assisté par Ordinateur.
CIL : Cellules d'Interconnexions Locales.
CIG : Cellules d'Interconnexions Globales.
CLB: Configurable Logic Bloc.
EPLD :Erasable Programmable Logic Device.
FPGA: Field Programmable Gate Array.
HDL : Hardware Description Language.
IC: Integret circuit.
lOB: Input Output Bloc.
LCA: Logic Cell Array.
LUT: Look up table.
NRE : Non Recurring Expenses.
PAL : Programmable array logic.
PLD: Programmable Logic Device.
RAM: Random Access Memory.
ROM: Read Only Memory.
SRAM: Static Random Access Memory.
USB : Universal Serial Bus.
VHSIC: Very High Speed Integrated Circuit.
VHDL: VHSIC Hardware Description Language.

Introduction générale.

Depuis les années 60, la Loi de Moore, qui prédit que la complexité en termes de transistors des circuits intégrés double tous les deux ans, reste vérifiée. Les circuits reconfigurables de type FPGA (*Field Programmable Gate Array*) n'ont pas échappé à cette loi. Depuis les premiers FPGA, développés comme une évolution naturelle des CPLD (*Complex Programmable Logic Devices*), ces circuits n'ont pas cessé de gagner en complexité et intègrent désormais jusqu'à un milliard de transistors pour les générations les plus récentes. Cette augmentation du niveau d'intégration s'est traduite par une croissance similaire de la puissance de calcul de ces circuits. Les FPGA, aux débuts restreints au simple rôle de *glue logic*, ils ont été ensuite utilisés pour faire du prototypage rapide d'ASIC (*Application Specific Integrated Circuits*) et trouvent depuis quelques années leurs places dans de nombreux domaines d'applications.

Même s'ils restent moins performants que de véritables circuits intégrés ASIC, leurs faibles coûts unitaires, leurs flexibilités et surtout le parallélisme à grain fin lié à leurs structures en font des cibles idéales pour des applications de calcul intensif. Ces applications, qui présentent généralement un fort parallélisme de données, comme le cas des réseaux de neurones, ils font partie d'un champ de recherche qui s'attaque à comprendre et à réaliser des systèmes capables de minimiser le comportement du cerveau. Ceci est fait dans le but de mieux comprendre le cerveau lui-même, mais aussi d'utiliser ce savoir dans le domaine de l'ingénierie pour résoudre des problématiques dans les domaines de la classification et la prédiction pour le développement des systèmes d'aide au diagnostique médical.

Pour répondre aux différents besoins notamment la conception des systèmes d'intelligence artificielle, le recours aux outils de Conception Assistée par Ordinateur (CAO) est plus qu'indispensable. L'avantage d'utiliser ces outils est de faire une conception matérielle et logicielle simultanément afin de réduire le temps de développement et d'augmenter la fiabilité par le test des prototypes virtuels avant la réalisation sur un circuit intégré.

Nous avons ciblé comme application dans ce travail l'implémentation d'un classifieur neuronal sur FPGA pour la reconnaissance du diabète.

Problématique :

L'objectif de cette étude est la modélisation d'un réseau de neurones artificiel en établissant une décomposition sous forme d'operateurs matériels simples à implanter sur un FPGA pour la reconnaissance du diabète. Notre contribution consiste à proposer une architecture flexible et fiable pour résoudre les problèmes d'intelligence artificielle à base des réseaux de neurones tout en réduisant le temps de simulation. Ceci nous permettra par la suite d'établir une prédiction médicale pour le diabète à travers les descripteurs de cette maladie.

Nous proposons cette procédure qui va classifier le mieux possible les descriptions sur les cas connus, mais surtout, qu'elle classifie bien les descriptions correspondantes aux nouveaux patients.

Etat de l'art :

Les réseaux de neurones sont des outils très puissants dans le domaine de l'intelligence artificielle, cette puissance se caractérise par l'utilisation des algorithmes généralement consommateur en termes de ressources mémoires, ce qui nécessite d'avantage des calculs et par conséquent plus de temps d'exécution. D'où la nécessité de proposer une architecture qui se caractérise par un calcul parallèle qui va nous réduire le temps d'exécution en donnant plus de flexibilité à notre réseau. Ce qui nous ramène à proposer une architecture matérielle implémentée sur les circuits reconfigurables FPGA afin de satisfaire les contraintes d'un cahier de charge.

Plan du mémoire :

Notre mémoire est scindé en trois chapitres : La première partie de ce document est dédiée aux circuits FPGA et leurs applications dans le domaine médical. Nous exposons une présentation de ces circuits et leurs outils de développement par ordinateur ainsi qu'une comparaison avec d'autres circuits. La deuxième partie sera consacrée pour les réseaux de neurones et principalement le perceptron multicouches ainsi que l'algorithme de la rétro propagation. Le troisième chapitre s'intéresse aux différents aspects de la conception de notre classifieur neuronal. Nous utiliserons le langage de description matérielle VHDL pour une modélisation structurelle des composants et un environnement logiciel pour la conception. Cet environnement de la firme XILINX est nommé XILINX ISE 10.1 qui contient les outils de CAO. Nous utilisons aussi le Système générateur sous l'environnement Simulink de MATLAB pour la modélisation.

Nous clôturons notre mémoire par une conclusion générale et des perspectives pour les futurs travaux.

Chapitre 1 : LES CIRCUITS LOGIQUES PROGRAMMABLES FPGA ET LEURS APPLICATIONS DANS LE DOMAINE MEDICAL

Introduction :

Dans de nombreux circuits, la technologie sous-jacente est basée sur un IC (circuit intégré), et un circuit électronique complet sera composé d'un certain nombre de circuits intégrés, avec d'autres composants de circuits tels que les résistances et les condensateurs[1]. Le dictionnaire anglais d'Oxford définit la technologie comme "l'application des connaissances scientifiques à des fins pratiques, en particulier dans l'industrie"[2]. Pour nous, cela vaut pour le matériel électronique sous-jacent et le logiciel qui peut être utilisé pour concevoir un circuit pour un besoin donné. Un ensemble de contraintes pour les circuits numériques sont développés, et le rôle à designer est d'arriver à une solution qui répond idéalement à toutes ces contraintes.

Les premiers circuits intégrés reconfigurables de type FPGA (*Field Programmable Gate Array*) ont été commercialisés par la société Xilinx en 1985. Depuis cette date, d'autres concurrents sont apparus sur un marché qui n'a pas cessé de croître. Au fur et à mesure que la complexité des FPGA s'est développée, leurs possibilités d'emploi se sont accrues, jusqu'à concurrencer sérieusement les circuits spécifiques pour des petits volumes de production (jusqu'à quelques milliers de composants). C'est pourquoi nous envisageons de plus en plus de développer l'utilisation des FPGA dans le cadre des applications spécifiques. Ils apportent une grande facilité d'emploi et permettent une réduction du coût de développement de l'électronique embarquée.

[1] Iron Grout, "Digital system design with FPGA and CPLDs", 2008, p: 3.
[2] MacMillen, D., et al., ''An Industrial View of Electronic Design Automation,'' IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 19, No. 12, December 2000, p: 1428.

A- Les circuits logiques programmables FPGA :

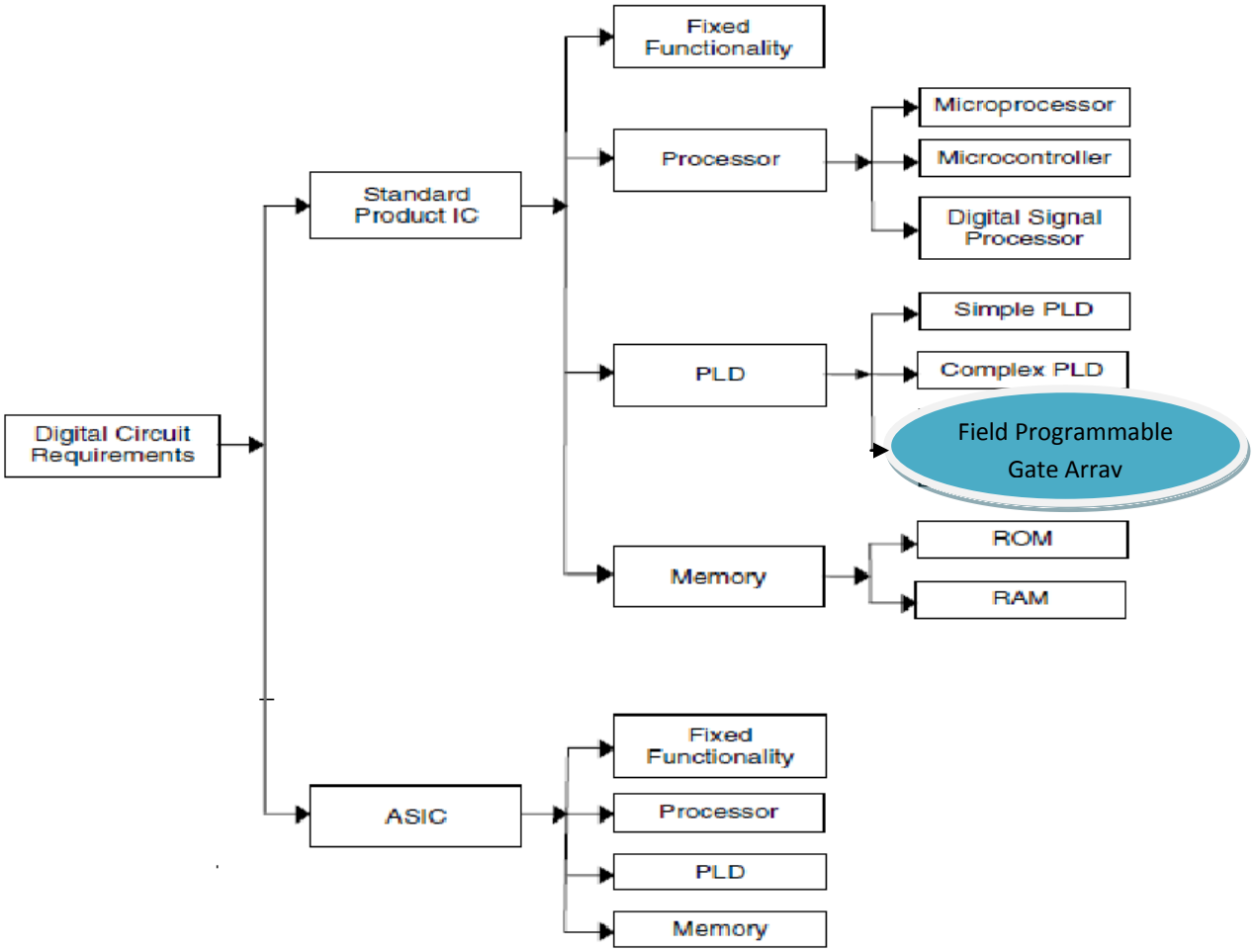


Figure 1 : Classification des circuits numériques.

1-Définition d'un FPGA :

FPGA (*Field Programmable Gate Arrays*) est un circuit intégré composé d'un réseau de cellules programmables. Chaque cellule est capable de réaliser une fonction désirée.

2- Architecture interne des FPGA :

Un FPGA appelé aussi LCA (*Logic Cell Array*) signifiant réseau de cellules logiques, c'est un composant électronique qui contient un grand nombre d'éléments logiques programmables reliés entre eux par à une matrice de routages programmables. Les FPGA sont composés de blocs logiques

programmables *CLB* et de blocs d'entrées/sorties *IOB* positionnées sur la périphérie du composant.

La structure d'un FPGA diffère d'un constructeur à un autre mais elle garde la même architecture globale illustrée par la figure suivante.

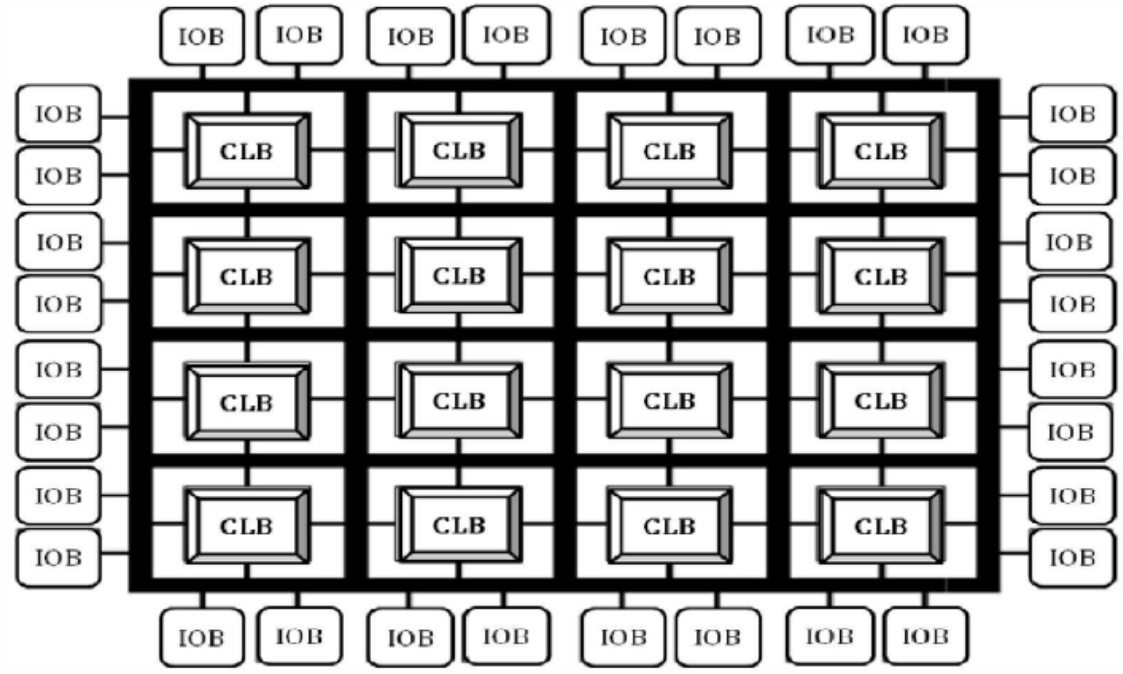


Figure 2: Structure globale d'un FPGA.

L'architecture d'un FPGA pour la famille XILINX se présente sous forme de deux couches :

- ✓ une couche active appelée circuit configurable,
- ✓ une couche de configuration.

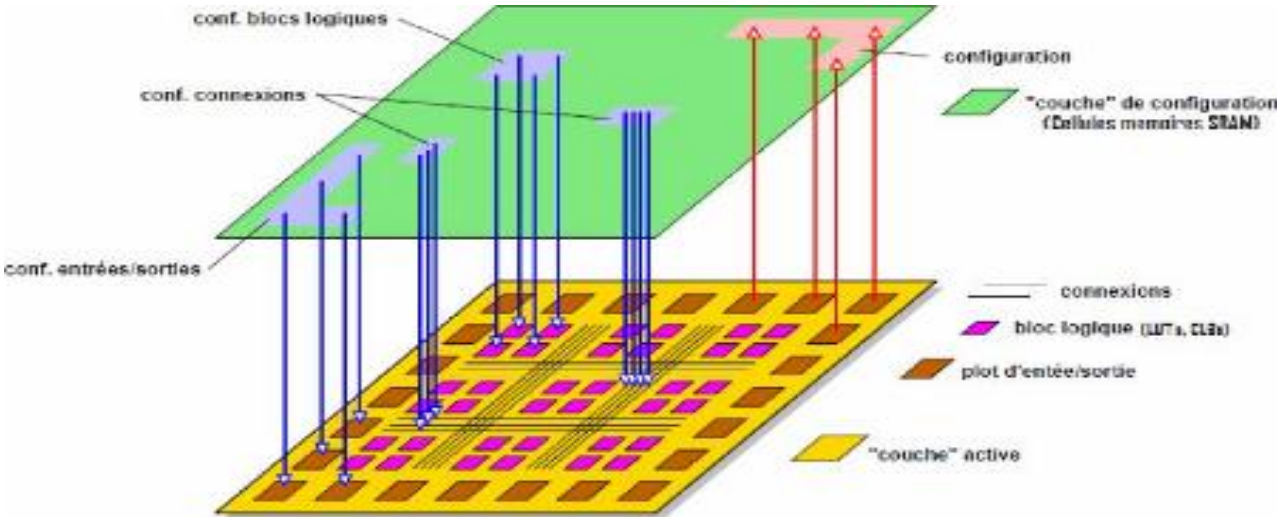


Figure 3: Architecture d'un FPGA.

2.1. La couche active :

Une couche active ou couche logique qui comprend des ressources logiques, des entrées/sorties, des ressources de routage, et éventuellement des blocs dédiés.

2.1.1. Les blocs logique programmables « CLB » :

Les cellules CLB nous permet d'implémenter des fonctions logiques combinatoires ou séquentielles complexes car chaque CLB est constitué d'une partie combinatoire et d'une partie séquentielle. Chaque fonction est décomposée en petites fonctions booliennes qui peuvent être contenues par des cellules *SLICES* .Ces dernières comportent des *LUT* pour la partie calculatoire qui sont des petites mémoires de 2^n mots de 1 bit, adressées par n bits provenant de la matrice de routage et une ou des bascules (généralement de type D « flip-flop ») contrôlées par un signal horloge pour la partie de mémorisation. Chaque CLB est composé de quatre slices répartis sur deux colonnes, chaque slice étant composé à son tour de deux cellules logiques (*Logic Cell*) disposées en colonne.

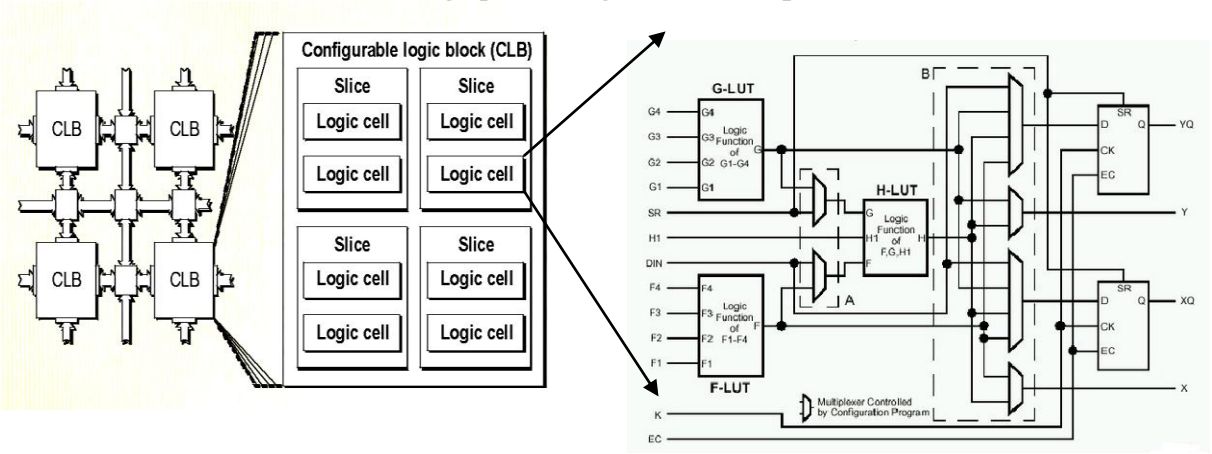


Figure 4: Architecture d'un CLB et de sa cellule logique de type SPARTAN.

2.1.2- Les blocs d'entrée/sortie IOB (Input/Output Blocks) [3]:

Les blocs d'entrées/sorties sont positionnés à la périphérie du circuit FPGA. Ces IOB permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant, et Chaque IOB peut

[3] Baba Hamed Amel, Conception d'un contrôleur vidéo en technologie FPGA, 2010, p : 36.

être configuré en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance).

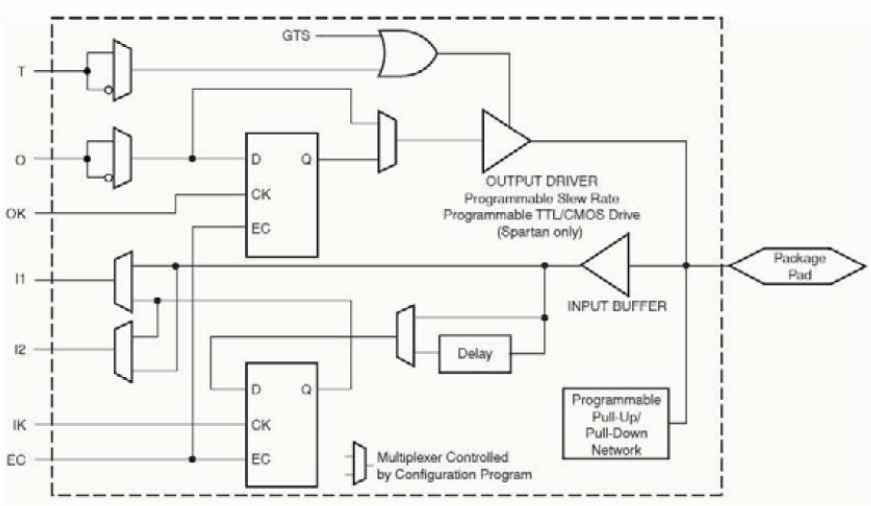


Figure 5: Schéma d'un bloc d'entrée/sortie (SPARTAN).

- Configuration en entrée :

Premièrement, le signal d'entrée traverse un buffer qui, selon sa programmation, peut détecter soit des seuils TTL, soit des seuils CMOS. Il peut être routé directement sur une entrée directe de la logique du circuit FPGA ou sur une entrée synchronisée. Cette synchronisation est réalisée à l'aide d'une bascule de type D, le changement d'état peut se faire sur un front montant ou descendant. De plus, cette entrée peut être retardée de quelques nanosecondes pour compenser le retard pris par le signal d'horloge lors de son passage par l'amplificateur. Le choix de la configuration de l'entrée s'effectue grâce à un multiplexeur (*Program Controlled Multiplexer*). Un bit positionné dans une case mémoire commande ce dernier.

- Configuration en sortie :

Nous distinguons les possibilités suivantes :
- Inversion ou non du signal avant son application à l'IOB.
- Synchronisation du signal sur des fronts montants ou descendants d'horloge.
- Mise en place d'un "pull-up" ou "pull-down" dans le but de limiter la consommation des entrées/sorties inutilisées.
- Chaque sortie peut délivrer un courant de 12mA. Ainsi toutes ces possibilités permettent au concepteur de connecter au mieux une architecture avec les périphériques extérieurs.

2.1.3. Les interconnexions [4]:

Dans les FPGA on peut distinguer deux types d'interconnexions (locale et globale). La composante locale a pour rôle d'interconnecter les éléments des blocs logiques et d'assurer la communication entre les blocs logiques directement adjacents sans avoir recours à la composante globale. La deuxième composante assure la circulation des données à l'échelle du FPGA entre les blocs logiques éloignés.

Au niveau local, on trouve donc des interconnexions configurables au sein des blocs logiques et entre les cellules logiques élémentaires. Réalisé au moyen de multiplexeurs, de buffers trois états et de portes passantes. On trouve également un groupe d'interconnexions vers les blocs logiques directement adjacents selon le même principe. Il s'agit d'interconnexions courtes présentant de faibles retards si on les compare aux interconnexions à longue distance.

Au niveau global, on trouve un réseau d'interconnexions horizontales et verticales à l'échelle du circuit situé entre les blocs logiques. L'échange des données entre les blocs logiques et ces interconnexions globales se fait par l'intermédiaire de Cellules d'Interconnexions Locales (CIL). L'aiguillage des données au sein du quadrillage d'interconnexions globales est assuré par des Cellules d'Interconnexions Globales (CIG) placées à chacune de ces intersections selon le schéma synoptique de la figure suivante :

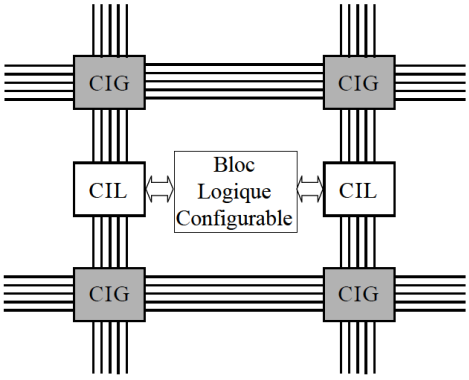


Figure 6: Réseau d'interconnexions global.

Ces cellules (CIL et CIG) sont constituées principalement de multiplexeurs et de portes de transmission à un ou deux transistors. On trouve également de nombreux buffers et buffers trois états afin de régénérer régulièrement les signaux le long des interconnexions.

[4] Jean-Max DUTERTRE, Circuits Reconfigurables Robustes, 2002, p : 13-14.

Il existe également un ou plusieurs réseaux d'interconnexions configurables dédiées aux signaux d'horloge. Ils comportent principalement des inverseurs et des buffers trois états.

2.2. Une couche de configuration :

La couche de configuration appelée mémoire de configuration permet de programmer électriquement les caractéristiques des ressources de la couche active. En effet, toutes les ressources logiques du FPGA sont contrôlées par le contenu de la «mémoire de configuration» (chez Xilinx cette mémoire est à base de cellules SRAM[1] ((*Static Random Access Memory*) volatiles, le FPGA doit donc être reconfiguré à chaque mise sous tension). Leur contenu fixe l'équation des LUTs, le routage des signaux, les entrées/sorties et leurs tensions ainsi que les paramètres de toutes les autres ressources du FPGA.

La cellule SRAM consiste en deux inverseurs CMOS connectés en boucle pour former un bistable. L'état de cette cellule peut être modifié par un signal électrique externe (ligne B). La cellule RAM est une structure de stockage volatile. La figure 7 représente une cellule SRAM à 5 transistors (a) et une cellule SRAM à 6 transistors (b). Malgré son coût, C'est la méthode utilisée dans les FPGA les plus performants à ce jour.

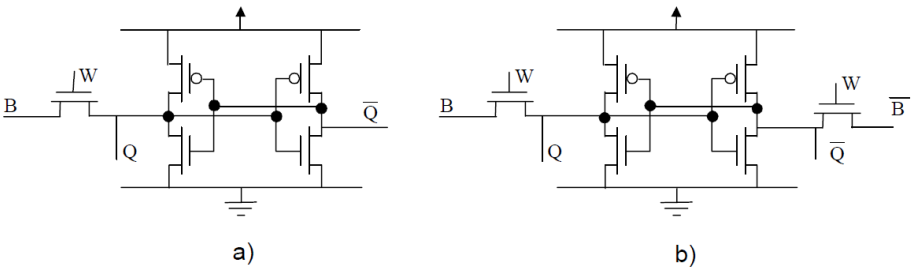


Figure 7: Cellule SRAM à 5 (a) et à 6 (b) transistors.

3. Méthodologie de conception :

3.1. Les outils de CAO (Conception Assistèe par Ordinateur) pour la configuration d'un FPGA[5] :

Le rôle principal confié aux outils de *CAO* se résume en 4 étapes qui sont : la description, la simulation, la synthèse, le placement et le routage et en dernier la configuration du FPGA. Un design peut être conçu à l'aide d'un éditeur schématique lors de la conception des circuits simples ou d'un outil de programmation utilisé pour les circuits complexes.

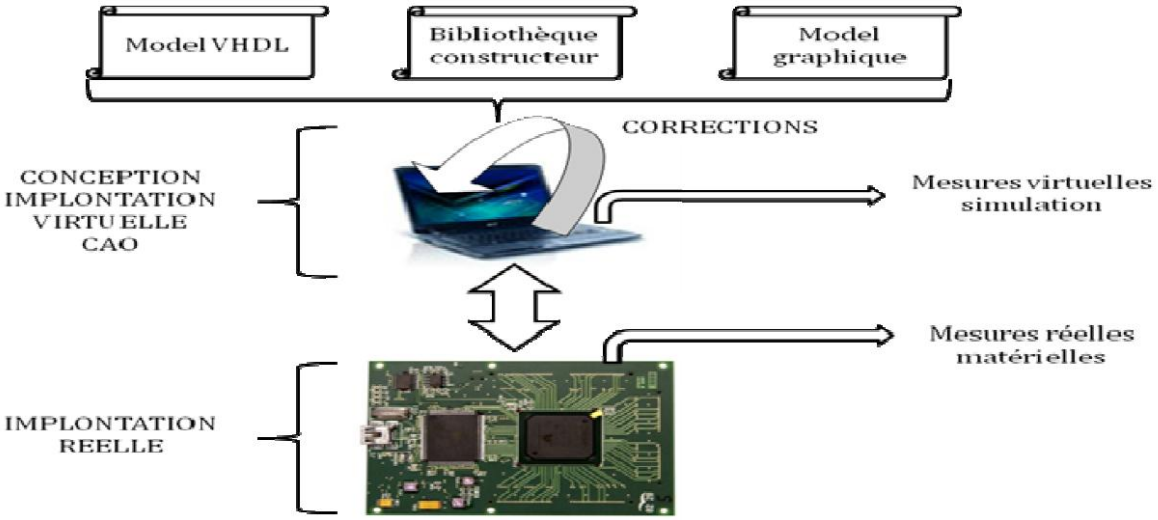


Figure 8: Mode d'exécution matériel des outils de CAO.

3.1.1 Spécification du design :

Pour réaliser un circuit, il faut tout d'abord envisager l'architecture globale de ce circuit et spécifier les trois éléments suivants :

- Le nombre de broches d'entrées/sorties et leurs localisations dans le composant *FPGA.*
- La spécification de la fréquence d'horloge du système.
- La spécification de la mémoire requise pour l'application.

[5] Zahir Ait Ouali, Application des FPGA à la commande d'un moteur asynchrone, p :20-21.

3.1.2. Développement du design :

- Spécification de la méthodologie de design (Outil de développement utilisé).
- La saisie du circuit Codage *RTL (VHDL, Verilog ...)*
  - Graphique *(Machine à états).*
  - Saisie HDL *(Hardware Description Language)* .
- La simulation (Prés et Post synthèse).

3.1.3. Synthèse :

L'outil de synthèse a pour objectifs de minimiser la surface de silicium, le temps de propagation ainsi que la consommation. Cet outil permet de convertir la représentation du design à partir du code *HDL* fourni pour produire une représentation au niveau de portes logiques. Cette phase s'occupe de déterminer quelles sont les structures susceptibles pour répondre à un cahier des charges étudié et de produire un code sous forme d'un fichier.

3.1.4. Placement et routage :

Le placement et le routage sont réalisés en définissant les chemins qui relient l'ensemble des blocs logiques choisies pour notre application à travers un algorithme de routage qui est sensé de faire l'aiguillage des données qu'il reçoit vers leurs destinations par action sur les nœuds de routage.

Plusieurs traitements sont nécessaires pour obtenir un fichier de configuration:

- *Partitionnement:* Les équations logiques spécifiques de notre application sont regroupées en un autre ensemble équivalent d'équations. Chaque équation de ce nouvel ensemble peut être implémentée dans un seul bloc logique du composant *FPGA.*

- *Placement:* Des blocs logiques sont sélectionnés et affectés au calcul des nœuds du réseau booléen.

- *Routage:* Les ressources d'interconnexion sont affectées à la communication de l'état des nœuds du réseau vers les différents blocs logiques.

- *Génération des données numériques de configuration:* Les informations abstraites de routage, de placement et les équations implantées dans les blocs sont transformées en un ensemble de valeurs binaires, fournies sous forme d'un

fichier appelé « bitstream » qui va être envoyé vers le FPGA via une interface de configuration.

3.1.5. Intégration et implémentation :

L'implémentation est la réalisation proprement dite qui consiste à mettre en œuvre l'algorithme sur l'architecture du circuit configurable cible, c'est-a-dire à compiler, à charger, puis lancer l'exécution sur un ordinateur ou calculateur. C'est une étape de programmation physique et de tests électriques qui clôturent la réalisation du circuit. La figure suivante résume un peu l'ensemble de ces étapes.

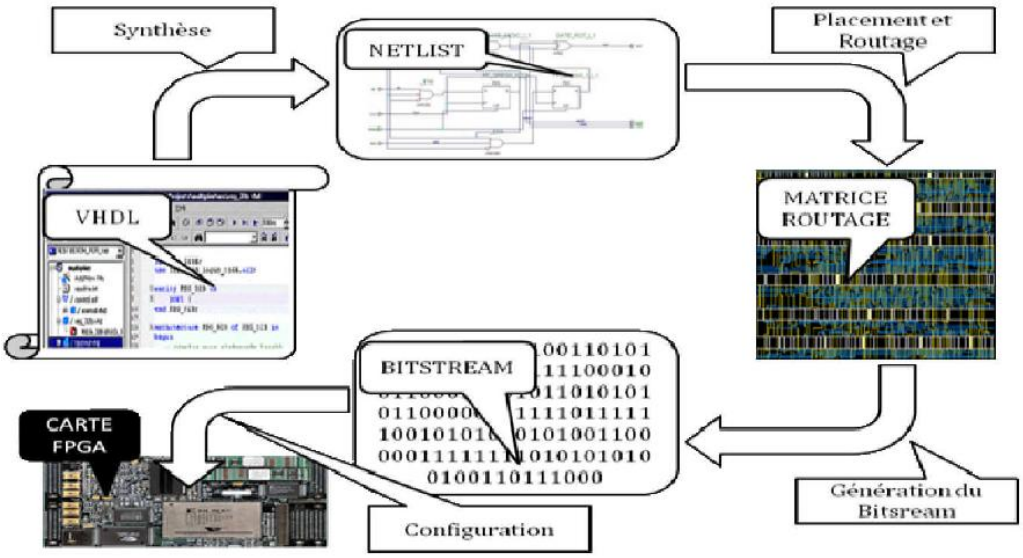


Figure 9: Cycle de programmation d'un FPGA en utilisant les outils de CAO.

3.2. Le langage VHDL :

3.2.1. Historique :

Dans les années 80, le département de la défense aux Etats-Unis fait un appel d'offre pour avoir un langage de description matérielle numérique unique. Le langage VHDL (*VHSIC **H**ardware **D**escription **L**anguage*) est inventé pour répondre à ces critères. Ce langage se base sur le VHSIC (***V**ery **H**igh **S**peed **I**ntegrated **C**ircuit*) qui est un projet de recherche mené par le groupement IBM/Texas. Ce langage est ouvert au domaine public en 1985 et deviendra une norme en 1987 sous la dénomination de IEEE 1076- 1987. Des changements minimes se feront pour la seconde normalisation en 1993 et il portera le nom VHDL'93. La dernière évolution de la norme est la norme IEEE 1076-2001.

Cependant, la norme IEEE 1076-2001 ne permet pas seule la description mixte (numérique et analogique). Le Verilog et le VHDL ont des capacités techniques équivalentes[6]. Et donc le choix du langage revient à l'utilisateur et aux outils qui les métrisent et par fois ce choix s'impose à travers les logiciels de simulation et de synthèse disponibles.

3.2.2. Description :

L'ambition des concepteurs du langage est de fournir un outil de description qui permet de créer des modèles de simulation. Initialement réservé au monde des circuits numériques, VHDL est en passe d'être étendu aux circuits analogiques.

Deux intérêts majeurs du langage sont :

- Des *niveaux de description* très divers: VHDL permet de représenter le fonctionnement d'une application tant du point de vue système que du point de vue circuit, en descendant jusqu'aux opérateurs les plus élémentaires. A chaque niveau, la description peut être structurelle (portrait des interconnexions entre des sous-fonctions) ou comportementale (langage évolué).

- Son aspect « *non propriétaire* »: le développement des circuits logiques a conduit chaque fabriquant à développer son propre langage de description. VHDL est en passe de devenir un langage commun aux nombreux outils de CAO, indépendants ou liés à des producteurs de circuits, des outils d'aide à la programmation.

4. Architecture : les tendances principales :

4.1. Architecture reconfigurable à grain fin :

Les premiers circuits reconfigurables (FPGA) ont été basés sur des éléments de traitement et des communications reconfigurables à grain fin[7]. L'élément source de ce traitement reconfigurable à grains fin est la LUT (Look Up table). Cette architecture à base de LUT est la plus utilisé commercialisé

[6] D.SMITH « HDL Chip Design : A pratical Guide for Designing, Synthesis & Simulating Asics &FPGAs using VHDL or Verilog » Doone Pubns . ISBN : 0965193438.
[7] Xun ZHANG, Contribution aux architectures adaptatives : étude de l'efficacité énergétique dans le cas des applications à parallélisme de données, 2009, page : 23.

avec une tendance allons de 3 à 6 entrées, ce qui permet de réaliser des fonctions combinatoires simples. A chaque élément configurable est associée une mémoire de configuration. Les technologies basées sur les mémoires SRAM permettent la reconfiguration dynamique par chargement de différents *bitstreams* (*bitstreams*: est un fichier qui permet la configuration du FPGA).

La LUT permet de stocker la table de vérité de la fonction à implémenter dans la cellule comme illustré dans la figure suivante :

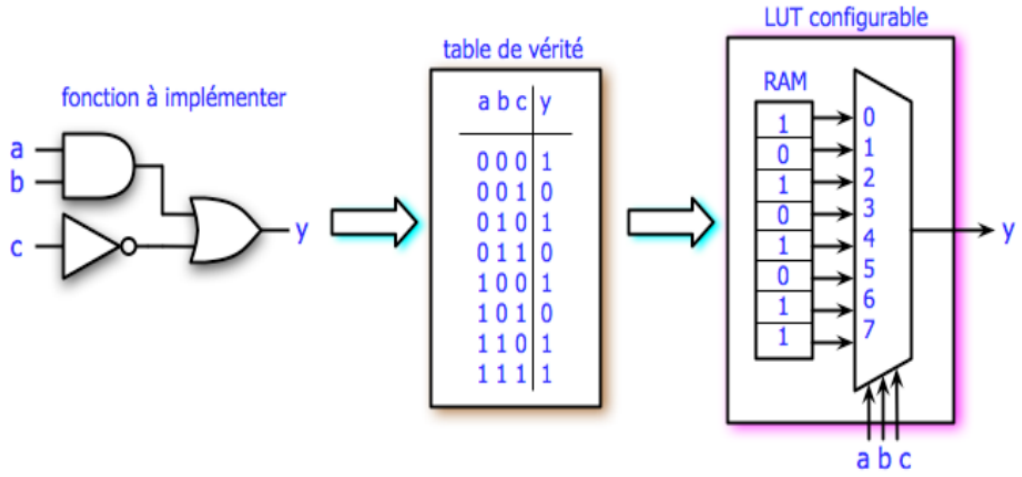


Figure 10: Model d'implémentation des fonctions dans les LUT.

Cette architecture peut implémenter n'importe quel circuit numérique vue sa grande flexibilité de reconfiguration. Un seul CLB peut réaliser des fonctions de grandes tailles, par exemple, il peut contenir des RAM distribuées de 256 bits, registre à décalage de 128 bits et fonction à 8 entrées. Cependant, l'inconvénient de cette architecture est l'occurrence d'occuper plus de surface, et de consommer plus par rapport à des structures à grain épais.

Cette structure permet de configurer les cellules logiques du circuit FPGA en fonction de différents besoins de l'application, pour spécialiser chaque tâche au maximum ce qui en général améliore l'efficacité.

4.2. Architecture reconfigurable à grain épais :

Les circuits reconfigurables sont évolués depuis quelques années des grains fins à des structures à grain plus épais qui peuvent être des fonctions simples arithmétiques et logiques voir des processeurs élémentaires plus complexes. Alors que les architectures à grain fin sont plus adaptées aux

traitements aux niveaux bits, les architectures à gros grains sont plus adaptées aux traitements des données sur des mots de plus grandes tailles[7] .

Pour développer cette architecture les concepteurs des FPGA ont envisagé des stratégies qui réduisent la taille de configuration et par ailleurs pour la reconfiguration dynamique, dont les interconnexions et les opérateurs sont configurables. Ceci à pour effet d'optimiser aux niveaux des bits, pour améliorer les performances de traitement arithmétique.

Les systèmes reconfigurables à grain épais se présentent comme un bon compromis entre FPGAs et circuits spécifiques (ASICs). Ce type d'architecture est basé sur une adaptation au niveau du jeu d'instructions.

4.3. Architecture reconfigurable hétérogène :

Afin de trouver un compromis entre les différentes caractéristiques, des architectures hétérogènes ont été conçue notamment par des fabricants des FPGAs tels que Xilinx et Altera qui proposent différentes familles de circuits intégrant plus ou moins de blocs spécialisés (DSP, mémoire, processeurs)[8]. Dans les circuits FPGAs actuellement commercialisés, on rencontre de plus en plus un mélange de granularité. En effet, certains éléments de calcul à grains épais sont embarqués dans des FPGAs, par exemple des multiplieurs, des modules "DSP", ou encore des processeurs sous forme d'IP ou en dur (PowerPCs pour Xilinx[9])[7].

La structure d'architecture reconfigurable hétérogène est composée de différents types d'éléments de calcul à grain fin et/ou à grain épais. L'adaptation est réalisée par le choix de type d'élément de calcul et les interconnexions entre ces éléments. Le type d'élément peut être changé en utilisant la reconfiguration partielle[10]. L'interconnexion entre les éléments peut être également reconfigurée.

[7] Xun ZHANG, Contribution aux architectures adaptatives : étude de l'efficacité énergétique dans le cas des applications à parallélisme de données, 2009, page : 28.

[9] Altera INC. : Stratix ii devic handbook,. Rapport technique, Altera,San Jos,Calif, USA, 2005.

[7] Xun ZHANG, Contribution aux architectures adaptatives : étude de l'efficacité énergétique dans le cas des applications à parallélisme de données, 2009, page : 31.

5. Comparaison entre les FPGA et les autres circuits spécifiques[11] :

Il existe une variété de circuits spécifiques dans le marché et donc le choix entre les différentes technologies devient une étape délicate et primordiale, car elle conditionne non seulement la conception mais aussi toute l'évolution du produit à concevoir. De plus, elle détermine le coût de réalisation et de la rentabilité économique du produit. Ceci implique qu'il faut faire une étude comparative des circuits existant sur le marché pour faire ce choix, en se basant sur plusieurs critères dont on peut citer le rapport coût / souplesse d'utilisation et les quantités à produire.

5.1 Comparaison entre les PLD et les ASIC :

Pour faire le choix entre les ASIC et les PLD il faut donc cité les avantages et les inconvénients de l'un par rapport à l'autre.

5.1.1. Avantages des PLD par rapport aux ASIC :

- Entièrement programmables par l'utilisateur,
- Leur flexibilité de reconfiguration en temps réel, ce qui facilite la mise au point et garantit la possibilité d'évolution,
- les délais de conception sont réduits.
- Leurs coûts sont faibles en petites et moyennes productions.

5.1.2. Inconvénients des PLD par rapport aux ASIC :

- Ils sont moins performants en termes de vitesse de fonctionnement (d'un facteur 3),
- le taux d'intégration est moins élevé (d'un facteur 10 environ),
- La programmation coûte les 2/3 de la surface de silicium.
- Le coût de l'ASIC est faible par rapport aux PLD en grande production (quoique les choses évoluent très rapidement dans ce domaine, notamment dans la compétition entre FPGA et pré diffusés). Au delà d'une certaine quantité, l'ASIC est forcement plus rentable que le PLD, dont on va décrire dans ce qui va suivre le seuil limite de cette quantité entre les ASIC et les PLD notamment pour les FPGA.

5.2 Comparaison entre les FPGA et les EPLD :

Si on tranche dans notre choix en faveur des PLD, il faut savoir si on doit utiliser un EPLD (*Erasable Programmable Logic Device*) ou un FPGA. En réalité, le choix est assez facile à faire. Car chaque circuit a son domaine d'utilisation, que ce soit pour les FPGA qui sont des pré diffusés (utilisation des fonctions logiques ou arithmétiques complexes). Par contre, les EPLD ressemblent dans leurs domaines d'utilisation plutôt à celui des PAL, par exemple la configuration des machines d'état complexes.

5.3 Seuil de rentabilité entre un FPGA et un ASIC :

Après avoir opté dans notre choix pour les FPGA par rapport aux autres circuits intégrés, et après la mise en œuvre de notre système, la production devienne la prochaine étape dans le cycle de développement. Avec le taux d'intégration qui devient de plus en plus important, les FPGA sont devenues un moyen très intéressant en fabriquant des petites et moyennes quantités, mais en grandes productions les ASIC deviennent plus importants en termes de coût comme illustré dans la figure 11. La question qui se pose est : combien d'unités doit-on produire pour que l'ASIC soit plus rentable que le FPGA ? Ou quel est le seuil de production entre les FPGA et les ASIC?

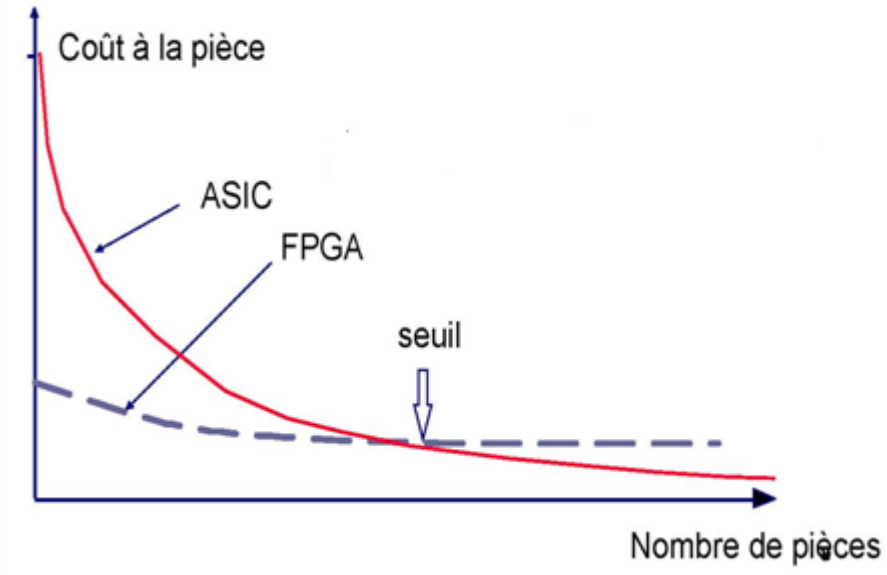


Figure 11: Seuil de rentabilité entre les FPGA et les ASIC.

La détermination du coût d'un circuit intégré se résume aux nombres de puces que l'on peut fabriquer sur une tranche de silicium c.à.d. sur la surface de la puce. Pour cela, il faut étudier les éléments qui agissent sur la taille de la puce, qui se réduise aux nombres d'entrées/ sorties et le nombre de portes

suffisantes pour réaliser une fonction logique. Lors des années 80 et avec l'apparition des FPGA, la production de ces circuits était très coûteuse par rapport aux ASIC à fonctionnalité identique. Mais aux cours des années, le nombre des broches ou de portes intégrées dans les circuits a augmenté comme illustré dans la figure 11, et par conséquent, les tailles des puces tendent à être fixées de plus en plus par les E/S. c'est ainsi que la balance s'inverse pour les FPGA.

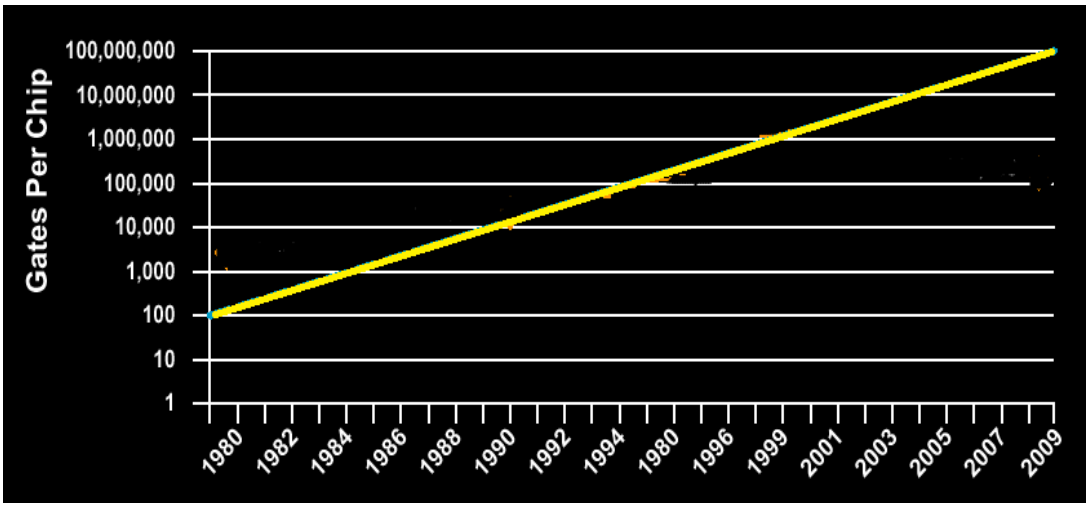


Figure 12: L'évolution d'intégration des portes sur puce aux cours des années.

Une analyse rapide peut donner un ordre de grandeur du seuil de rentabilité entre un FPGA et un ASIC. Prenons comme exemple un boîtier de 10.000 portes. L'étude se base sur des données fournies par la société d'études de marché ***DATAQUEST*** en 1995. La formule de base du seuil de rentabilité est la suivante [11]:

Seuil de rentabilité = NRE + (développement et outils) + (X unités * prix à l'unité)

NRE (Non Recurring Expenses) : les frais fixes de mise en œuvre.

On obtient pour les ASIC et les FPGA les deux formules suivantes :

ASIC = $25 000 (NRE) + $79 000 (développement et outils) + (X unités * $13)

FPGA = 0 NRE + $25 000 (développement et outils) + (X unités * $79)

[11] C.ALEXANDRE, Circuits logiques programmables, 2005, page :25-26-27.

Il n'y a pas de NRE pour un FPGA. Les NRE sont imputés à chaque fois que l'on fait appel à un fondeur. A partir des 2 équations ci-dessus, le seuil de rentabilité est atteint pour 1 196 unités. Le FPGA devient plus cher à produire qu'un ASIC au-delà de 1 196 unités. En effet, il existe d'autres facteurs qui influent grandement sur le seuil de rentabilité :

• ***Le « time to market » (*temps de mise sur le marché*):*** C'est le temps écoulé entre le début de l'étude et la phase de production. Prendre du retard sur le lancement d'un produit sur le marché, en raison d'un cycle de développement et de mise au point très longue, a des effets négatifs en termes de rentabilité. Le cycle moyen de développement d'un FPGA est de 11 semaines, il passe à 32 semaines pour un ASIC.

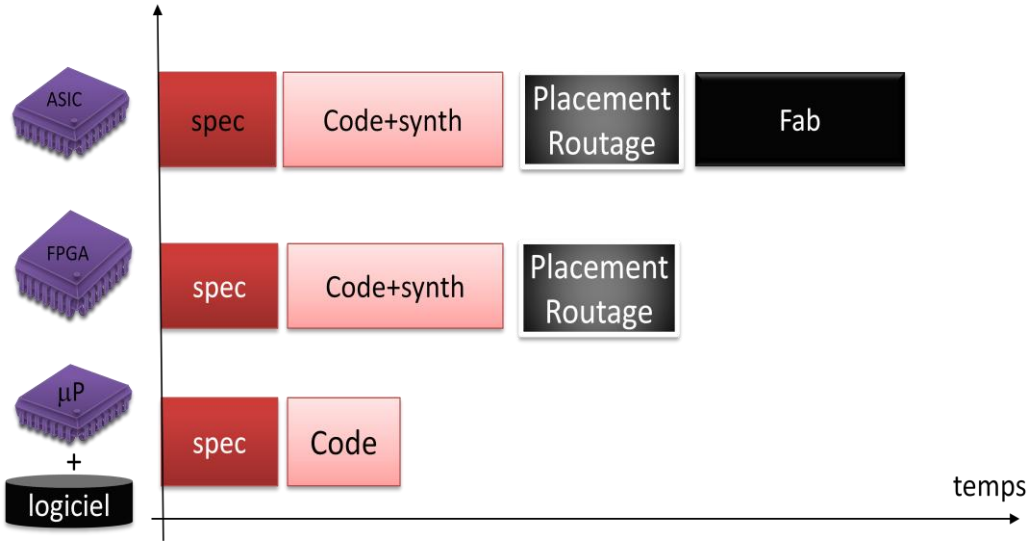


Figure 13: Le temps de mise en œuvre des circuits sur le marché.

Les chiffres permettant de quantifier les seuils de rentabilité entre ASIC et les FPGA. L'ordre de grandeur du seuil de rentabilité est le suivant :

Jusqu'à 5000 pièces	Plus de 5000 pièces
FPGA	ASIC

6. Avantages et inconvénients des FPGA :

Avantages	Inconvénients
-Technologie « facile» à maîtriser. -Temps de développement réduit. -Reprogrammable. -Idéal pour le prototypage. -Coût peu élevé. -Parallélisme de traitement. -Flexibilité et la possibilité de réduire fortement les délais de développement et de commercialisation. -La reconfiguration, parfois en temps réel.	- Performances non optimisées. -Temps de réponse long par rapport aux *ASIC*.

B- Quelque application dans le domaine médical :

Ultrasons, rayons X, CT, et les applications PET requirent toutes les opérations de calcul intensif pour des algorithmes tels que la transformée de Fourier rapide (FFT) en utilisant le parallélisme de pipeline personnalisé. L'amélioration des performances de l'algorithme est réalisée par le déchargement à partir du processeur hôte sur un FPGA.

Les FPGA sont reconfigurables par logiciel. Cet avantage permet à un concepteur de gagner le temps de développement en démontrant un traitement basé sur le matériel tout en préservant la possibilité de reprogrammer le FPGA pour accueillir des modifications qui sont nécessaires après spécification initiale. Bien que les conceptions FPGA peuvent être complexes et modulaires, les cartes à base de FPGA réalisent une infrastructure de communication, de connectivité, d'interface de bus d'E /S et de mémoire SRAM. Le développement de ces composants internes peut être fastidieux et distrayant.

Les FPGA ont rapidement gagné en popularité pour les applications médicales. En ce qui concerne l'imagerie médicale, les FPGA sont principalement utilisés dans la détection et la construction de l'image. L'application de détection implique les systèmes embarqués, avec des exigences de performance en temps réel et des défis importants de l'interface matérielle. Reconstruction de l'image, en revanche, est semblable à un problème de calcul de haute performance.

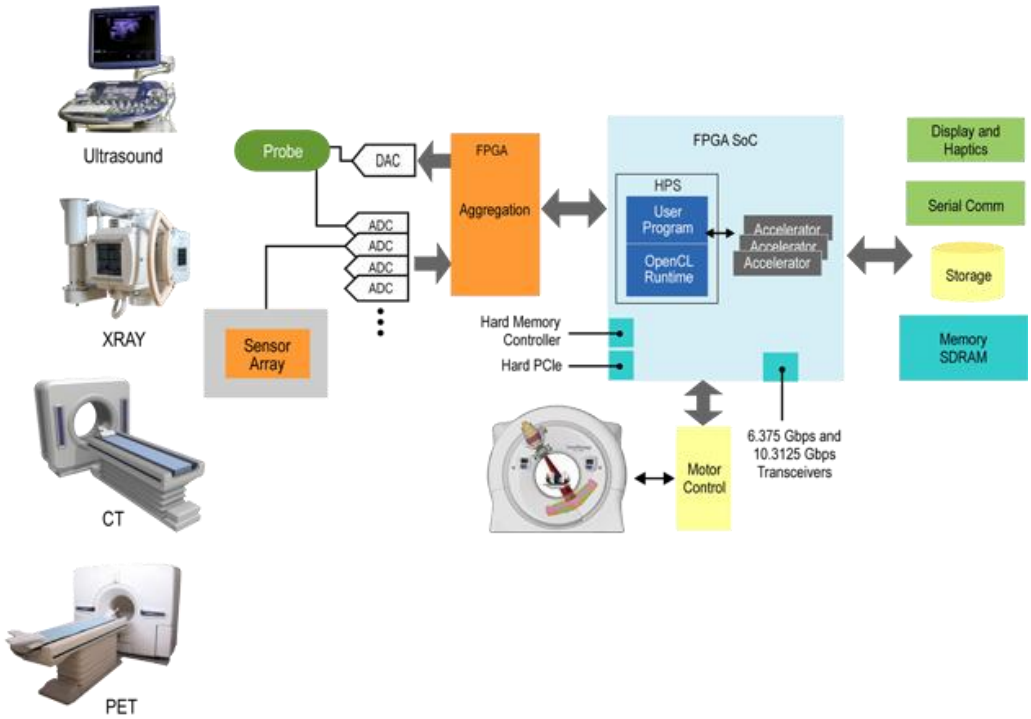


Figure 14: Utilisation des FPGA dans le domaine médical.

1. Domaine d'Imagerie médicale « Radiologie » :

Sous le titre du *« Développement d'une communication temps réel sur FPGA » / X-Ray Génération*, la société GE (Générale Electrique) qui est unique en son genre par son expertise de la médecine moléculaire, à accélérer le futur de la médecine et révolutionne la prise en charge du patient par l'aide donné aux professionnels de la santé de tel façon à prévoir les pathologies potentielles avant même qu'elles ne se déclarent. Le département de développement des générateurs dans cette entreprise conçoit, définit et développe le sous-système destiné à fournir l'énergie et la haute tension nécessaires au tube à rayons x. Les générateurs sont utilisés dans une large gamme d'appareils d'imagerie médicale, tels que des scanners, des appareils de mammographie, radiographie….

Une génératrice haute tension est notamment constitué d'un module « HV-tank » Porté à la haute tension, entre 80 kV et 140 kV. Le « HV-tank » Intègre diverses fonctionnalités, comme par exemple la régulation de courant du filament du Tube permettant la création des rayons X. Le «HV-tank » embarque à ce titre une carte électronique digitale à base de FPGA, permettant l'implémentation de diverses régulations hardwares. Cette carte de contrôle, portée à la haute-tension, dialogue avec les autres cartes intelligentes du scanner en basse-tension, grâce à un lien de communication propriétaire dont le hardware est compliqué.

2. Imagerie médical « endoscopie » :

Les systèmes endoscopiques nécessitent une fonctionnalité croissante. Beaucoup de dispositifs endoscopiques sont considérés comme des systèmes à haute définition (HD). Cependant, les chirurgiens continuent de demander des plates-formes de définition encore plus élevés que la technologie récente. De nombreuses tendances de l'équipement de systèmes endoscopiques investie pour augmenter les performances de transformation, qui à son tour soutien les algorithmes d'imageries les plus avancées. Comme qualité d'image s'améliore, la technologie des capteurs s'améliore avec, et par conséquent les systèmes devront traiter plus de données d'image de la tête de caméra et mettre en œuvre des fonctions de plus en plus compliquées dans leurs traitements.

Les concepteurs ont besoin pour mettre en œuvre leurs fonctions dans des appareils de traitement à haute performance, tels que les FPGA, qui permettent des performances évolutives à travers les générations de plates-formes. Les FPGA ont tendance à conduire la courbe de processus pour les nœuds technologiques parce que chaque génération des dispositifs pouvant intégrer d'avantage les ressources de traitement (par exemple, des éléments logiques (LE), traitement (DSP) des ressources signaux numériques avec un soutien en virgule fixe ou flottante, une mémoire sur puce, et plus vite support d'E/S). En outre, la structure de FPGA intrinsèque se prête à traitement en parallèle, ce qui est avantageux pour les algorithmes d'imageries.

La souplesse des FPGA dans leurs reprogrammations donne un bon dispositif d'accompagnement pour les capteurs d'image parce que les appareils peuvent traiter les non-linéarités inhérentes. En autre, les FPGA ont des capacités d'E/S hautes performances qui permettent un soutien énorme par les interfaces périphériques tels que SDI, DVI, SAS / SATA et USB.

3. La chirurgie :

Dans le domaine de la chirurgie, la tendance est de standardiser sur des blocs de conception FPGA à base d'innover une vision 3D et un contrôle précis de robotique assisté, dans le but à aboutir à une chirurgie minimalement invasive, ce qui signifie beaucoup moins de traumatisme et une récupération plus rapide pour les patients que la chirurgie conventionnelle ouverte. En combinant les avantages des FPGA à la précision, la dextérité, le contrôle de la robotique, le système chirurgical va bénéficier d'un plus large éventail de spécialités. A partir de la chirurgie cardiaque et générale, la plate-forme a permis à des

méthodologies pour une variété de chirurgie comme l'urologie, pédiatrie, gynécologie, colorectal, et même cérébrale.

Figure 15: La robotique utilisée dans la chirurgie.

C- Conclusion :

Nous avons vu dans ce chapitre que le développement d'une commande implémentable sur FPGA est une tache très complexe, car elle exige une connaissance et une maîtrise très avancé des technologies relatives à ce type de circuit et de leurs environnements de développements. Malgré ces inconvénients, il reste très intéressant d'exploiter ce type de circuit car il nous offre des performances qu'on les trouve pas dans les autres circuits. On peut les résumer à leur puissance de calcul parallèle qui nous permet d'économiser le temps, en plus de leur flexibilité de configuration et reconfiguration, ils possèdent une gestion dynamique de connectivité à leur environnement, ce qui permet non seulement d'implémenter les différents algorithmes adéquat à des applications très pointues (dans le domaine médical), mais tout en gardant une grande flexibilité de communication à des environnements très variés en conservant toujours l'efficacité énergétique et les performances nécessaires.

CHAPITRE2 : Réseau de neurones.

Dans ce chapitre nous présentons des réseaux de neurones, le principe de calcul neuronal et la modélisation d'un neurone avec quelques fonctions d'activation. Cette présentation est suivie de quelques indications générales sur la méthodologie de développement d'un réseau de neurones ; nous introduisons les notions de base dont nous aurons besoin par la suite de notre étude. Ensuite, nous présentons un model de réseau utilisé pour le traitement des données que nous utilisons dans notre application pour la classification du diabète. Nous indiquons l'algorithme qui nous semble le plus approprié en tenant compte des contraintes de notre application. Enfin, nous terminons par quelques avantages et inconvénients liés aux réseaux de neurones.

Introduction :

Les travaux sur les réseaux de neurones artificiels dénommés « réseaux neuronaux », ont été motivé dès leurs créations par la reconnaissance que le cerveau humain calcule d'une manière totalement différente de l'ordinateur numérique classique. Le cerveau est un ordinateur très complexe, non linéaire et parallèle (système de traitement des informations). Il a la capacité d'organiser ses constituants structurels, appelés neurones, de manière à effectuer certains calculs (par exemple, la reconnaissance des formes, la perception et le contrôle moteur) beaucoup plus rapide que l'ordinateur numérique le plus performant en existence aujourd'hui[12] .

Les neurones sont considérés comme le support physique de l'intelligence. Un réseau neuronal est l'association de plusieurs éléments de base nommé neurone formel. Les principaux réseaux se distinguent par l'organisation de ces neurones en couches, complets ou autres, c'est-à-dire leur architecture, son niveau de complexité catégorisé par le nombre de neurones, la présence ou non de boucles de rétroaction dans le réseau, par le type des neurones, leurs fonctions de transition ou d'activation et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques.

Les réseaux de neurones ont la capacité de stocker de la connaissance empirique dans les poids synaptiques et de la rendre disponible à l'usage, grâce à des algorithmes d'apprentissage automatiques, on peut régler un réseau de neurones pour lui faire accomplir des tâches qui relèvent de l'intelligence artificielle.

[12] Simon Haykin, Neural networks, Second Edition,1999, page :23.

1- Du neurone biologique au neurone artificiel :

1.1 Le neurone biologique :

La Cellule est l'élément fondamental du tissu nerveux. Ce sont les neurones qui constituent l'unité fonctionnelle du système nerveux. Le relais qui assure la transmission de l'influx nerveux est la synapse.

En biologie, la plupart des neurones sont constitués de trois parties : les dendrites qui collectent les informations en provenance des autres neurones (stimuli externes) ; le soma qui traite ses entrées et renvoie une impulsion en sortie ; et finalement l'axone, par lequel le signal sortant est transmis aux autres neurones. La connexion entre deux neurones est appelée la synapse. Il existe deux sortes de synapse : les synapses électriques (minoritaires) et les synapses chimiques (majoritaires).

Du point de vu fonctionnel, il faut considérer le neurone comme une entité polarisée, c'est-à-dire que l'information ne se transmet que dans un seul sens : des dendrites vers l'axone.

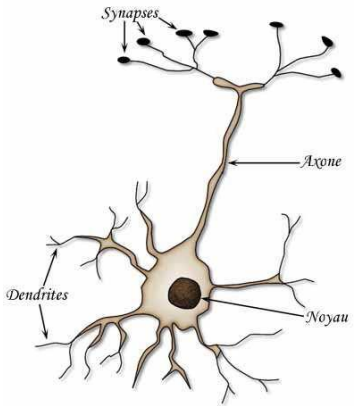


Figure 16: Neurone biologique.

1.2- Le neurone artificiel :

Tout comme le neurone biologique, le neurone artificiel est un processus de calcul simple, qui fait la somme pondérée des signaux d'entrées en provenance de neurones appartenant à un niveau situé en amont, et renvoie en sortie une fonction de cette somme. Cette fonction f est définie lors de la création du neurone, elle dépendra fortement de l'objectif désiré. L'information stockée à l'intérieur d'un neurone est appelée poids «w_i » qui représente la force de la connexion entre neurone.

On pourra décrire dans le tableau suivant la transition entre le neurone biologique et le neurone formel :

Neurone biologique	*Neurone artificiel*
Synapses	Poids de connexion
Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

Tableau 1: Transition du neurone biologique au neurone formel.

2-Neurone formel :

2.1 Définition :

Le neurone formel est une approximation très grossière de l'opération du neurone biologique. Le neurone créé par McCulloch et Pitts[13] est un neurone utilisant une fonction binaire.

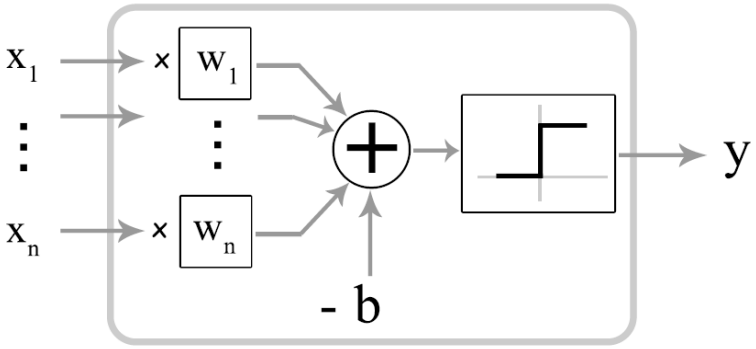


Figure 17: Neurone artificiel ou formel.

Un neurone formel doit être capable de :
- Recevoir en entrée les différentes informations provenant des neurones environnants.
- Analyser ces informations, de manière à envoyer en sortie une réponse.
- Ajuster cette réponse avant de l'envoyer aux neurones suivants.

[13] McCullochW. and PittsW. L.R. A logical calculus for the ideas immanent in nervous activity. *Bull.Math. Biophysics*, 1943, pages 115.133.

Il est donc tout naturel d'assimiler un neurone à un triplet (*poids* , biais , fonction d'activation f) :
- On multiplie chaque valeur d'entrée par la composante des *poids* correspondante.
- On compare la valeur obtenue à une valeur de référence : le biais, ce qui revient à soustraire le biais.
- Enfin, on applique la fonction d'activation à cette différence.

2.2-Fonction d'activation :

Cette fonction permet de définir l'état interne du neurone en fonction de son entrée totale, citons à titre d'exemple quelques fonctions souvent utilisées :

2.2.1-Fonction binaire a seuil :

Nous allons citer dans cette partie quelques exemples des fonctions binaires à seuil.

Fonction Heaviside définie par :

$$h(x)=\begin{cases}1 & \text{si } x \geq 0\\ 0 & \text{sinon}\end{cases}$$

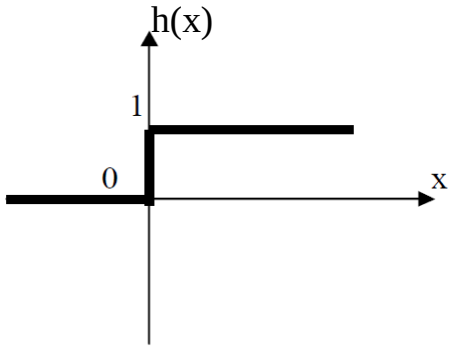


Figure 18: Fonction Heaviside.

Fonction Signe définie par :

$$Sgr(x)=\begin{cases}+1 & \text{si } x \geq 0\\ -1 & \text{sinon}\end{cases}$$

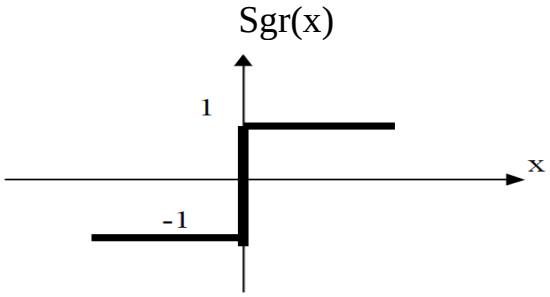


Figure 19: Fonction Signe.

2.2.2 -Fonction linéaire ou identité:

Si la fonction identité a l'avantage d'être simple, il n'a que peu de rapport avec la réalité (le signal de sortie est non borné, linéaire par rapport aux signaux d'entrées, ce qui ne correspond pas du tout au fonctionnement des neurones biologiques). Mais elle est généralement utilisée dans la couche d'entrée du réseau.

$$F(x) = x$$

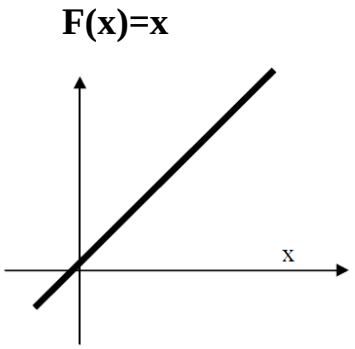


Figure 20: Fonction identité.

2.2.3-Fonction linéaire à seuil ou multi-seuils:

$$F(x) = \begin{cases} x & x \in [\,u, v\,] \\ v & \text{si } x \geq v \\ u & \text{si } x \leq u \end{cases}$$

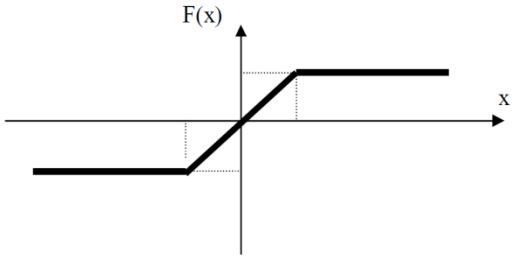


Figure 21: Fonction linéaire à seuil.

Cette fonction représente un compromis entre la fonction linéaire et la fonction à seuil : entre ses deux barres de saturation, elle confère au neurone une gamme de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de réponse du neurone.

2.2.4- La fonction sigmoïde :

$$f(x) = \frac{1}{1 + e^{-x}}$$

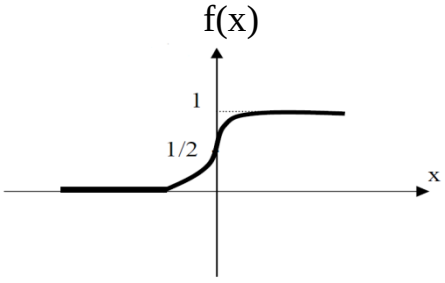


Figure 22: Fonction sigmoïde.

3-Procédure de développement d'un réseau de neurones:

Le développement et la mise en œuvre d'un réseau de neurone se résument en quelques étapes : la première est le choix de la base de donnée ou la collecte des données, cette étape est primordiale car son objectif est de rassembler un nombre de données suffisants pour la subdivisé en deux, une pour l'apprentissage et l'autre pour le test. Elle doit être avec le minimum de bruit possible pour constituer une base représentative des données susceptibles d'intervenir en phase d'utilisation du système neuronal.

La seconde étape consiste à effectuer une phase d'analyse de données de manière à déterminer les caractéristiques discriminantes pour détecter ou différencier ces données. Dans cette phase l'obtention de ces caractéristiques nous permet à la fois de réduire le temps de simulation à travers la réduction de la taille du réseau, et de réduire le temps d'apprentissage. Elle permet aussi d'agir sur les performances du système (pouvoir de séparation, taux de détection).

Cette étude statistique permet de dévoiler la redondance qui existe dans notre base, et de déterminer le nombre de classe auxquelles ces données appartiennent dans un problème de classification.

La troisième étape est la subdivision ou la séparation de la base en deux parties distinctes, la première pour effectuer l'apprentissage et la seconde pour la phase de test. Il n'y a pas de règles pour déterminer ce partage de manière quantitatif. Il résulte souvent d'un compromis en tenant compte du nombre de données dont on dispose et du temps pour effectuer l'apprentissage. Cependant, chaque base doit satisfaire aux contraintes de représentativité de chaque classe de données et doit généralement refléter la distribution réelle, c'est à dire la probabilité d'occurrence des diverses classes.

La quatrième étape est le choix du type de réseau à implémenter. Dans la littérature, on trouve une variété de réseaux qui possède chacun des avantages et des inconvénients, donc il faudra bien se placer et de faire un choix optimal pour répondre aux différentes contraintes. Notre choix dépend de plusieurs critères, par exemple, si nous avons un problème de classification, de la nature des données à étudier, d'éventuelles contraintes temporelles et autres….

On proposera quelques types des classifieurs et leurs modes de classification appropriés dans le tableau suivant :

Mode Apprentissage	*Règle d'apprentissage*	*Architecture*	*Algorithme*	*Objectif*
Supervisé	Correction d'erreur	Perceptron simple ou Multicouche	Perceptron, Rétro-propagation. Adaline. Madaline.	Classification. Approximation de Fonction. Prédiction. Contrôle.
	Bolzmann	Récurrente	Apprentissage de Bolzmann	Classification.
	Hebb	Multicouches non bouclés	Analyse de discriminent linéaire.	Analyse de donnée. Classification.
	Par compétition	A compétition	LVQ$^{\alpha}$	Catégorisation au sein d'une classe. Compression de donnée.
		ART	ARTMap	Classification. Catégorisation au sein d'une classe.
Non supervisé	Correction d'erreur	Multicouche non bouclés	Projection de Sammon	Analyse de donnée.
	Hebb	Non bouclé ou à compétition	Analyse en composant principale	Analyse de donnée ou compression de donnée.
	Par compétition	A compétition	VQ$^{\beta}$	Catégorisation. Compression de données.
		Carte de Konnen	SOM$^{\gamma}$	Catégorisation. Analyse de données.
		ART	ART-1, ART-2	Catégorisation.
Mixte	Correction d'erreur et par compétition	RBF	RBF	Classification. Approximation de fonction. Prédiction. Contrôle.

Tableau 2: Quelques modèles de classification[14].

$^{\alpha}$ Algorithm learning vector quantization.

$^{\beta}$ Algorithm Vector Quantization

$^{\gamma}$ Algorithm Self Organizing Map.

[14] Rachid Ladjaj, 2002/2003, Ingénieur 2000, Filière informatique et réseau. Site internet : http://www-igm.univ-mlv.fr/~dr/XPOSE2002/Neurones/index.php?rubrique=Apprentissage

La cinquième étape exige la normalisation des données, qui consiste à faire une phase de prétraitement à travers la détermination des maximum des différents paramètres de notre base, ensuite, de normaliser cette dernière à travers ces valeurs.

La 6^{éme} étape est l'apprentissage du réseau choisi. Plusieurs types d'apprentissage peuvent être adaptés au même type de réseau, le but derrière ce choix est à la fois les performances et la rapidité de convergence. Le plus souvent d'abord on utilise un apprentissage structurel, si ce n'est pas le cas, l'obtention des paramètres architecturaux optimaux se fera par comparaison des performances obtenues pour différentes architectures testées de réseaux de neurones, ensuite on fait l'apprentissage paramétrique. Le critère d'arrêt de l'apprentissage est souvent calculé à partir d'une fonction de coût, en caractérisant l'écart entre les valeurs de sortie obtenues et les valeurs de références.

La dernière étape est la validation du réseau entrainé à travers une base réservé pour le test, différent de celle utilisée pour l'apprentissage. A travers ce test on fait extraire les performances du système, si ces performances ne sont pas satisfaisantes, il faudra soit modifier l'architecture du réseau, soit modifier la base d'apprentissage et de test.

4- L'apprentissage :

L'apprentissage consiste à adapter la connaissance du RNA au problème posé. La connaissance étant définie par l'ensemble des pondérations sur les liens du réseau ainsi que par sa topologie, l'apprentissage est donc étroitement liée au réglage adéquat des poids d'interconnexion, ainsi qu'à l'adaptation de la topologie[15].

4.1- L'apprentissage supervisé:

L'apprentissage est dit supervisé lorsque la base de donnée est constituée de couples de valeur (entrée - sortie désirée). Le but de cet apprentissage est de déterminer le vecteur des poids w et dans certain cas même le vecteur biais du réseau capable de mettre les informations ou résultat obtenu en correspondance avec le désirée, le réseau s'adapte par comparaison entre le résultat qu'il a

[15] Jean-Luc Bloechle, Réseau de Neurones Artificiels pour la classification des fontes Arabes et la distinction entre la langue Arabe et les langues Latines**,** Université de Fribourg, page :7, Juin 2003.

calculé, en fonction des entrées fournies, et la réponse attendue en sortie. Ainsi, le réseau va se modifier jusqu'à ce qu'il trouve la bonne sortie, c'est-à-dire celle attendue, correspondant à une entrée donnée.

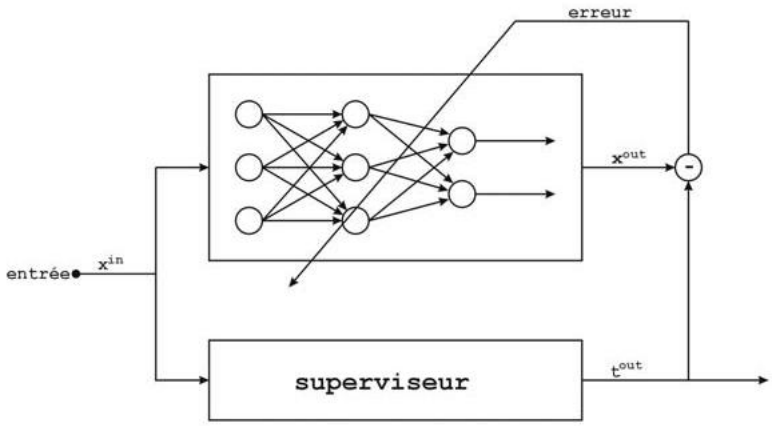


Figure 23: Apprentissage supervisé.

4.2- Le perceptron multicouche:

Enthousiasmé par la réussite d'une règle d'apprentissage directement inspirée des mécanismes observés dans le model biologique sur un neurone formel, F. Rosenblatt proposa qu'un perceptron puisse résoudre n'importe quel problème de classification en un temps d'apprentissage fini [Rosenblatt, 1962]. Malheureusement, après quelques recherches, M. Minsky et S. Papert prouvèrent que cette assertion était trivialement fausse, ce qui eu pour effet de jeter un discrédit sur le perceptron qui aura mis presque dix ans à se dissiper [Minsky et Papert, 1969]. En fait, le perceptron se heurtait tout simplement aux mêmes limitations que son concurrent ; une limitation qui n'avait finalement rien à voir avec les performances de leur règle d'apprentissage respective mais qui était d'origine purement architecturale[16] .

La parade a été de mettre plusieurs perceptrons en réseau, formant ainsi le perceptron multicouches. Toutefois, cette opération fut ici bien plus facile puisque la règle d'apprentissage de Hebb se généralise à un réseau entier sans poser de difficulté particulière : qu'il s'agisse de résultats intermédiaires ou non, la valeur de chaque poids synaptique du réseau est augmentée à chaque fois que l'entrée et la sortie correspondante sont activées en même temps. Bien entendu, l'efficacité de cette règle, qu'elle soit appliquée à un perceptron isolé ou à un réseau complet, n'en est pas démontrée pour autant.

[16] Yohann BÉNÉDIC, APPROCHE ANALYTIQUE POUR L'OPTIMISATION DE RÉSEAUX DE NEURONES ARTIFICIELS**,** 2007, page : 18-19.

L'intérêt des neurones artificiels, son possibilité d'être mis en réseau, est discutable [Minsky et Papert, 1969]. C'est la raison pour laquelle leur véritable essor ne vient qu'à la fin des années quatre-vingt, lors de la mise au point d'un algorithme d'apprentissage spécialement pensé à l'échelle d'un réseau : la rétro propagation du gradient [Werbos, 1990, 1994 ; Mc Clelland et Rumelhart, 1988 ; Parker, 1985 ; Le Cun, 1987]. Cette technique détermine en particulier la façon dont l'erreur observée en sortie doit être propagée dans les couches internes, comme illustré dans la figure suivante.

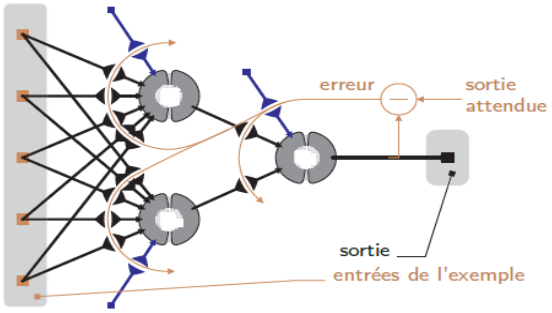


Figure 24: Apprentissage d'un perceptron multicouche par rétro propagation de l'erreur.

4.3- L'algorithme de la rétro propagation de l'erreur en appliquant la descente de gradient :

La rétro-propagation est actuellement l'outil le plus utilisé dans le domaine des réseaux de neurones vue son efficacité prouvée dans plusieurs domaines. C'est une technique de calcul qui peut être appliquée à n'importe qu'elle structure de fonctions dérivables.

On va considérer pour ce qui va suivre un réseau constitué d'une couche d'entrée composée de n neurones, une couche cachée de m neurones, et d'une couche de sortie constituée de « L » neurones. Leurs poids respectifs sont $W_{i\,J}$ entre la couche d'entrée et la couche cachée, et $W_{J\,K}$ entre la couche cachée et la couche de sortie. Nous avons (n*m) connexions entre la couche d'entrée et la couche cachée, et (m*L) connexions entre la couche cachée et la couche de sortie.

Les fonctions d'activation sont respectivement la fonction identité pour la couche d'entrée, et la fonction sigmoïde pour la couche cachée et la couche de

sortie. Les sorties de la couche d'entrée sont symbolisées par la lettre X, pour la couche cachée par la lettre Y, et pour la couche de sortie par la lettre S.

Toutes les données revendiquées précédemment sont illustré dans la figure suivante :

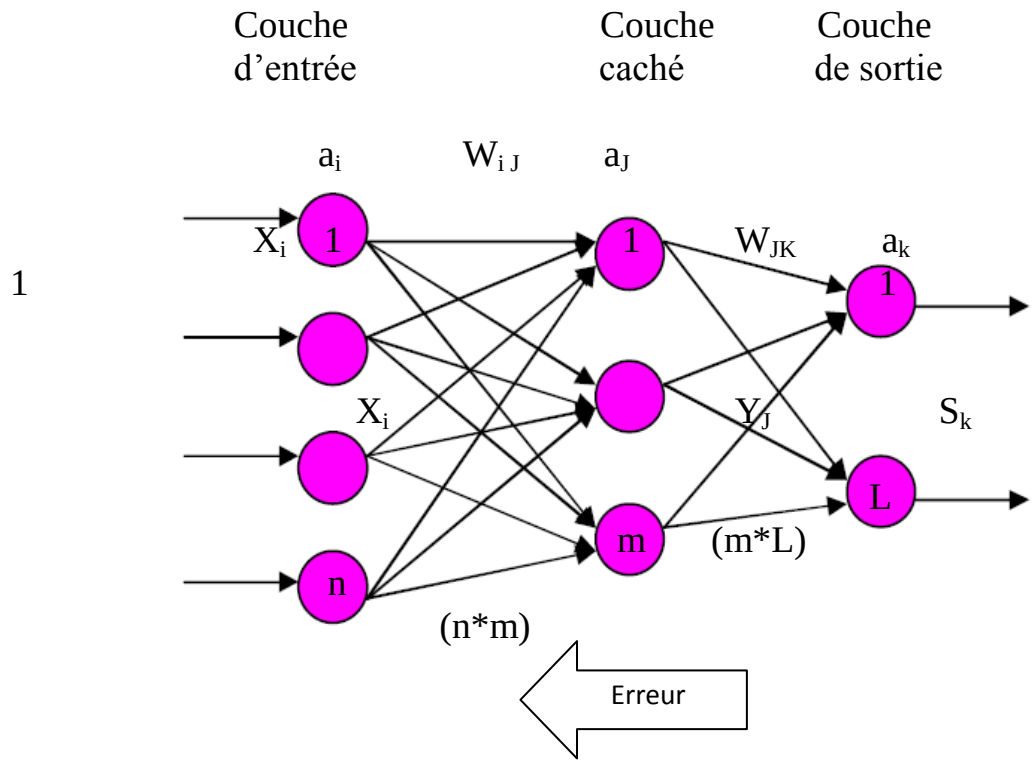


Figure 25: Exemple d'un réseau pour la rétro propagation de l'erreur.

La figure 26 illustre l'algorithme de la rétro propagation de l'erreur pour le réseau précédemment décri et représenté par la figure 10. L'algorithme commence par la propagation des entrées X_i de la couche d'entrée vers la couche de sortie, en passant par la couche cachée. Une fois la sortie calculée, elle doit être comparée avec la sortie désirée, si une erreur est commise, on doit la rétro-propager de la sortie vers l'entrée en modifiant les poids et les biais de tous le réseau de façon à se qu'il s'adapte et apprend pour obtenir la sortie désirée comme il est décri dans l'algorithme.

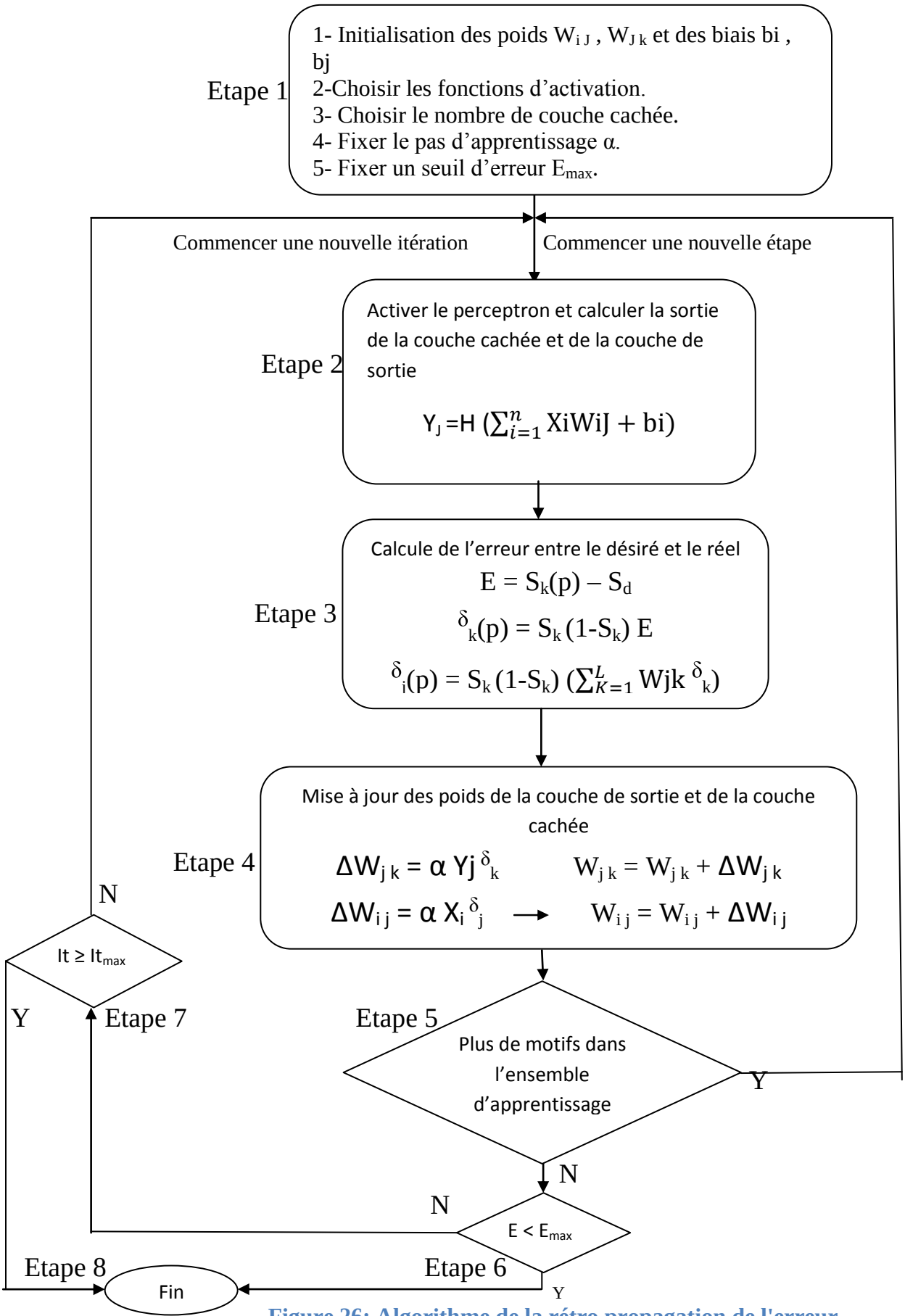


Figure 26: Algorithme de la rétro propagation de l'erreur.

Résumé de l'algorithme de rétro propagation

Etape 1 :

1- Initialisation des poids $W_{i\,J}$, $W_{J\,k}$ et des biais bi , bj
2-Choisir les fonctions d'activation.
3- Choisir le nombre de couche cachée.
4- Fixer le pas d'apprentissage α.
5- Fixer un seuil d'erreur E_{max}.
6- Fixer le nombre maximum d'itération It_{max}.

Etape 2 :

Activer le perceptron en appliquant un vecteur d'entré et calculer la sortie de chaque couche.

Etape 3 :

Calculer les termes d'erreur de signal de la couche de sortie et des couches cachées.

Etape 4 :

Mise à jour des poids de la couche de sortie et des couches cachées.

Etape 5 :

Tester s'il y a plus de motifs dans l'ensemble d'apprentissage, aller à l'étape 2. Sinon aller à l'étape 6.

Etape 6 :

Si la condition de convergence vers le seuil fixé est atteinte, aller à l'étape 8. Sinon, aller à l'étape 7.

Etape 7 :

Si la condition sur le nombre d'itérations est atteinte, aller à **l'étape 8,** sinon aller **à l'étape 2** et commencer une nouvelle itération.

Etape 8 :

Fin.

5- Considérations pratiques :

Les poids synaptiques sont considérés comme le support de l'intelligence, ils doivent être initialisés à de petites valeurs aléatoires. D'autre part la valeur du taux d'apprentissage α a un effet significatif sur les performances du réseau, s'il est petit l'algorithme converge lentement, par contre, s'il est grand l'algorithme risque de générer des oscillations. Il doit être compris entre 0 et 1 pour assurer la convergence optimale de notre algorithme.

Il n'existe pas de règles permettant de déterminer le nombre de couches cachées dans un réseau donné, ni le nombre de neurones dans chacune d'elles, ceci revient à l'utilisateur de les fixer généralement après plusieurs tests. Mais il faut faire très attention il ne faut pas augmenter ces nombre à des chiffres très grands, car plus ces valeur augmentent plus elle rend leurs implémentation plus complexes et leurs coûts plus importants.

Théoriquement, l'algorithme ne s'arrête que lorsque le réseau atteint le minimum d'erreur, correspondant à un gradient nul, ce qui n'est jamais rencontré en pratique. C'est pourquoi on fixe les conditions d'arrêt d'un apprentissage qui sont le seuil d'erreur, et un nombre d'itération maximal.

6 - Avantages et inconvénients :

L'intérêt des réseaux de neurones artificiels réside dans le parallélisme de leur structure, leur capacité d'adaptation ainsi que dans leur mémoire distribuée. La rapidité d'exécution est une qualité importante et elle justifie souvent le choix d'implanter un réseau de neurones. Ces qualités ont permis de réaliser avec succès, plusieurs applications de classification, de filtrage, de compression de données, contrôleur, etc…Ces propriétés sont à la source de la capacité de généralisation d'un réseau et donc de son aptitude à adopter un comportement correct.

Les réseaux de neurones présentent des inconvénients :

- Un réseau de neurones ne dispense pas de bien connaître son problème, de définir ses classes avec pertinence, de ne pas oublier les variables importantes.
- Un réseau de neurones est une « boîte noire » qui n'explique pas ses décisions. Il est également hasardeux de conclure ou de créer des règles sur le fonctionnement et leur comportement ce qui rend leur interopérabilité très difficile même impossible.

Les réseaux de neurones ont une très bonne prédiction statistique (ayant la capacité de s'accommoder de valeurs très bruitées ou même manquantes), mais ils sont complètement impossibles à inspecter. La perte partielle de compréhension est compensée par la qualité des prédictions.

7- Conclusion :

Dans ce chapitre, nous avons présenté les méthodes utilisées lors du développement de notre système, ainsi que la méthode la plus utilisée pour l'apprentissage des réseaux de neurones statiques, qui est l'algorithme de la rétro propagation qui repose simplement sur le calcul par la descente du gradient, appliquée à toute fonction dérivable. Nous avons cité aussi quelques considérations pratiques qui peuvent nous aider à développer de façon optimale notre système.

Chapitre 3 : Modélisation et implémentation du classifieur sur FPGA.

1-Introduction :

Dans ce chapitre, notre travail consiste dans un premier temps à effectuer la phase d'apprentissage, et à extraire les poids finaux et les biais, ensuite les utilisés dans l'architecture qu'on va proposer pour l'implémenté sur un FPGA.

2- Base de données :

Dans le cadre de notre étude, nous utilisons la base de données médicale réelle (Indians Diabetes Pima). L'ensemble de données a été choisi du dépôt d'UCI[17] qui réalise une étude sur 786 femmes Indiennes Pima, originaires du continent asiatique. Implantées dans les régions montagneuses du Mexique, en gardant une alimentation traditionnelle elles comptent une proportion de diabétiques de l'ordre de 10%. Ce taux élevé témoigne d'une prédisposition génétique. Ces mêmes femmes Pima, qui ont stoppé leurs migrations en Arizona, États-Unis, adoptant un mode de vie occidentalisée, développent un diabète dans presque 50% des cas. Parmi ces 786 femmes, nous utilisons uniquement 392, qui ont des paramètres complets contrairement au reste à savoir les 140, celles-ci présentaient des données manquantes principalement sur la pression artérielle.

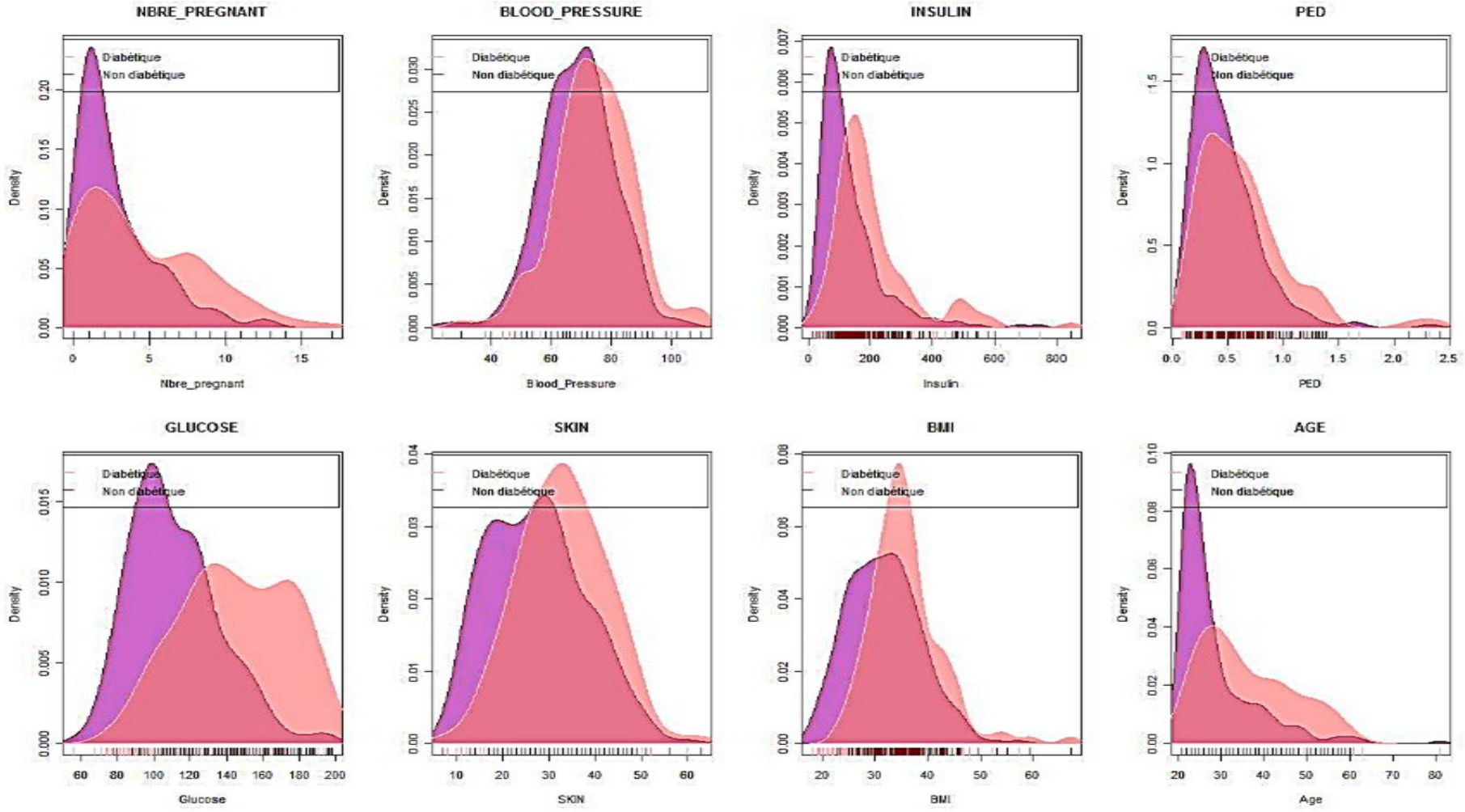
Le diagnostic est une valeur binaire variable «classe» qui permet de savoir si le patient montre des signes de diabète selon les critères de l'Organisation Mondiale de la Santé. Les huit descripteurs cliniques sont :

1. Npreg : nombre de grossesses,
2. Glu : concentration du glucose plasmatique,
3. BP : tension artérielle diastolique,
4. SKIN : épaisseur de pli de peau du triceps,
5. Insuline : dose d'insuline,
6. BMI : index de masse corporelle,
7. PED : fonction de pedigree de diabète (l'hérédité).
8. Age : âge.

La figures 30 montre la répartition des diabétiques et non diabétiques des paramètres de la base de données :

[17] Data base PIMA :http ://www.cs.umb.edu/ rickb/files/UCI/diabetes.arff

Figure 27: Représentation de la base de données : distribution des diabétiques et non diabétiques dans les paramètres de la base.



Les figures 30 nous montrent clairement la répartition des diabétiques et les non diabétiques suivant les variations des paramètres. Nous prenons à titre d'exemple le Glucose, on remarque que pour la majorité des non diabétiques ils ont une valeur du glucose inférieure à 140 mg/dl (selon les normes dictées par l'OMS pour le test d'intolérence au glucose), et un nombre important de diabétiques dont le taux de glycémie varie dans une plage qui est supérieur à 140mg/dl. La partie de chevauchement entre les deux crée confusion et prend des différences pour les experts, d'où la nécessité d'un système automatique.

3- Le choix de l'architecture et du type d'apprentissage :

Dans ce travail, on est en face à un problème de classification supervisée, nous considérerons une structure classique de réseau neuronal appelée PMC (Perceptron Multicouches). Ce dernier consiste en une succession de couches constituées d'unités neuronales, les quelles possèdent une fonction d'activation, nous avons choisie les fonctions identité et sigmoïde respectivement pour la couche d'entrée et les autres couches (cachées et de sortie). Chaque neurone, à l'intérieur d'une couche, reçoit des signaux provenant de la couche précédente, les multipliés par leurs poids et effectue un calcul somme, puis calcul la fonction de transfert, et transmet le résultat à la couche suivante. Parmi les caractéristiques du perceptron multicouches, il n'existe pas d'interconnexions entre les neurones situés à l'intérieur d'une même couche : les activations des différents neurones sont seulement propagées de la couche d'entrée vers la couche de sortie à travers tous les neurones constitutifs du réseau. La couche d'entrée collecte les variables d'entrée c.à.d. leur nombre est proportionnel au nombre des variables ou des paramètres, tandis que la couche de sortie produit les résultats.

Le PMC est une structure très couramment utilisée vue sa simplicité. On détermine, selon les besoins, le nombre de couches cachées, le nombre de neurones par couche cachées et la fonction d'activation de chaque couche. Ce qui donne une bonne flexibilité au réseau. Le choix final considéré pour notre application en ce qui concerne le nombre de couche et le nombre de neurone dans chaque couche est obtenu après plusieurs tests en modifiant plusieurs fois l'architecture et de prendre la structure qui présente un compromis entre les différentes performances du réseau (Taux de classification, Sensibilité, Spécificité) cette phase de simulation est effectué à travers l'environnement MATLAB. L'algorithme adopté pour la phase d'apprentissage est l'algorithme de la rétro propagation (voir le chapitre 2) afin de modifier les poids synaptiques et les biais.

La qualité de l'apprentissage augmente avec la taille de l'ensemble d'apprentissage, de même, la précision de l'estimation augmente avec la taille de l'ensemble test. Après avoir subdivisé notre base de données en deux sous ensembles, le premier contient 2/3 de la base pour effectuer l'apprentissage et le seconde contient 1/3 pour le test.

4- Conception du classifieur neuronal :

Ils existent à l'heure actuelle deux catégories d'outils de conception. Ceux qui sont issus du monde de la modélisation (Matlab pour la modélisation mathématique) et ceux issus de la CAO électronique. Pour une description au niveau matériel, les outils de conception CAO sont encore les outils les plus compétents pour réaliser une synthèse fiable. Toutefois, les outils de modélisation permettent de définir, comme l'outil Simulink de Matlab, un fichier VHDL pour le placement et le routage vers la cible FPGA.

4.1. Conception par les outils de CAO :

Avec la miniaturisation continuelle de la technologie des semi-conducteurs, les circuits électroniques sont devenus trop complexes pour être conçus à la main ce qui a résulté la nécessité des outils informatiques adéquats et c'était la naissance des outils de CAO[18]. Les concepteurs réalisent et testent les circuits sur ordinateur avant de lancer la fabrication. On peut distinguer trois types d'outils de CAO selon les modes de descriptions trouvés dans ces derniers:

-Un outil de CAO utilisé que pour la synthèse et la simulation (comme MODELSIM).
-Un outil de CAO qui utilise le mode schématique et textuel (comme ISE de XILINX).
-Un outil de CAO qui peut utiliser le mode schématique, textuel et permet même le dessin du masque (*layout*) (comme CADENCE).

4.1.1- Représentation des données :

Dans un premier temps, il faut savoir comment présenter les données (les nombres signés et réels). Pour la quantification en virgule fixe le nombre de points et la distance entre les points est arbitraire, mais le plus souvent nous choisissons un ensemble symétrique par rapport à l'origine.

[18] Si Mahmoud KARABERNOU, « *Méthodologie De Conception Et Langages De Description De Matériel* », Ecole Nationale Supérieure De L'Electronique Et De Ses Applications (ENSEA), Septembre 2009.

Évidemment, le nombre de valeurs représentables détermine le nombre de bits qui sont nécessaires pour leur représentation[19].
La façon la plus naturelle est de réserver un bit de signe et le reste pour représenter le module comme fraction binaire. C'est la représentation par signe et par module (Figure 31).

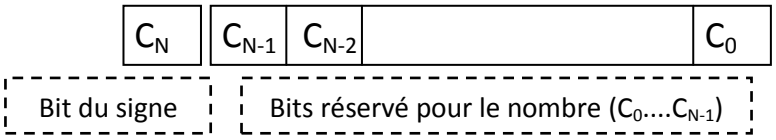


Figure 28: Représentation d'un nombre en binaire.

4.1.2-L'architecture d'un neurone :

4.1.2.1- Modélisation d'un neurone :

Pour la modélisation d'un neurone, deux opérations principales sont effectuées par ce dernier, la multiplication et l'accumulation. Donc cette structure comporte un multiplieur et un accumulateur par neurone. Les entrées des neurones de la couche précédente entrent dans le neurone en série et elles sont multipliées par leurs pondérations correspondantes. Les valeurs multipliées seront additionnées dans un bloc accumulator, ensuite le résultat obtenu est transféré vers la fonction d'activation.

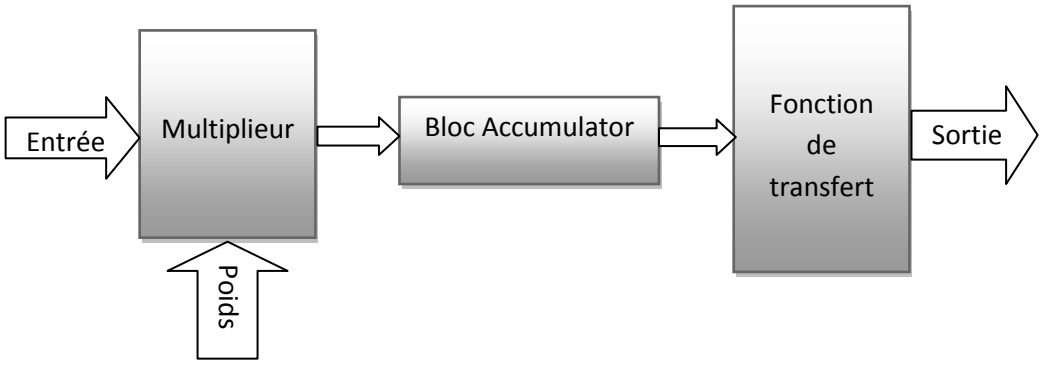


Figure 29: Modélisation d'un neurone.

[19] Jean-Luc Beuchat, *Etude Et Conception D'opérateurs Arithmétiques Optimisés Pour Circuits Programmables*, THÈSE No 2426 (2001).

4.1.2.2- Architecture matérielle d'un neurone :

L'architecture matérielle d'un neurone est constituée à base d'un multiplieur qui fait la multiplication entre l'entrée du neurone et son poids respectif. Les poids sont stockés dans une ROM et à travers un adressage le poids correspondant va être sélectionné et envoyé vers le multiplieur. Au démarrage la valeur du biais stockée dans le registre va être chargée dans l'accumulateur à travers un multiplexeur manipulé par une ligne de commande appelée « Load ». Après cette opération, on va changer le niveau logique de la ligne « Load » pour permettre le passage effectué en stockant le résultat dans l'accumulateur. Une boucle de retour est misse en œuvre pour de nouvel addition.

La ligne de commande « Enable »va être activée pour faire fonctionner l'accumulateur et la 2ème ligne « Out-Enable » va être tous le temps actif afin de permettre d'effectuer à chaque fois l'opération d'addition et de passer le résultat à la couche suivante. Le processus est synchronisé sur un signal d'horloge, le nombre de cycles d'horloge pour qu'un neurone termine son travail doit être égal aux nombres de connexions de ce neurone à la couche précédente. Cette architecture de neurones est illustrée dans la Figure 33.

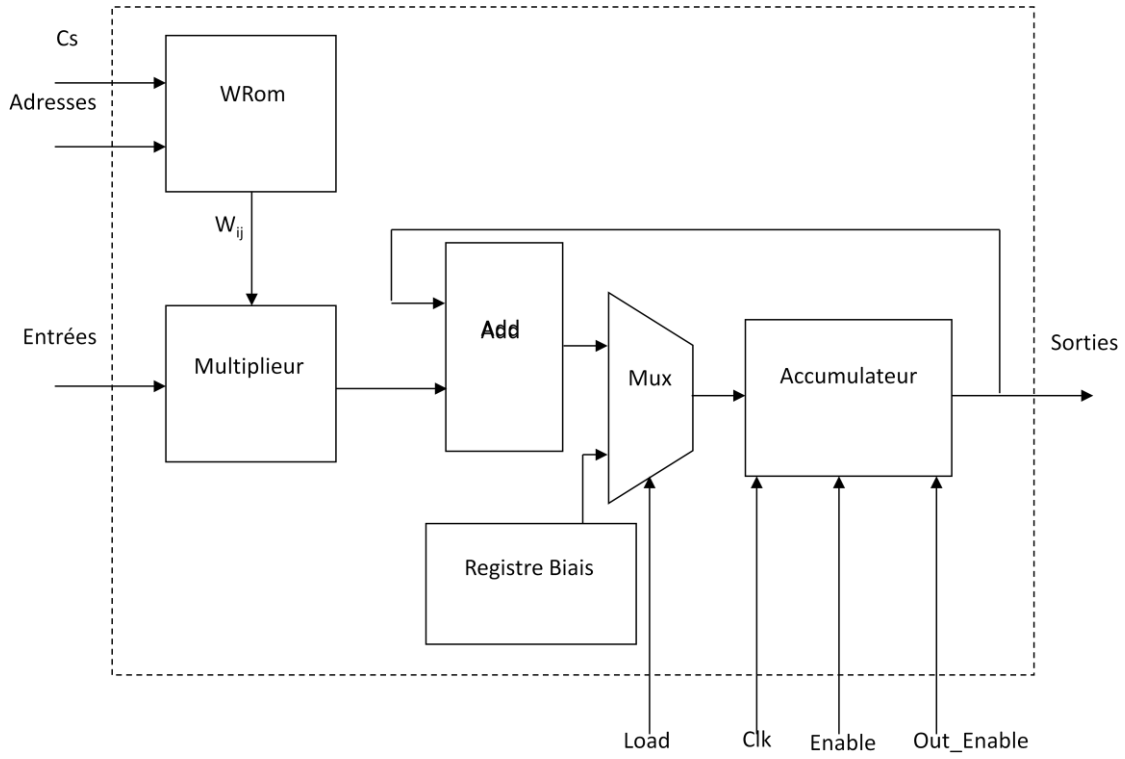


Figure 30: Architecture matérielle d'un neurone.

4.1.3- Architecture matérielle globale du réseau de neurone :

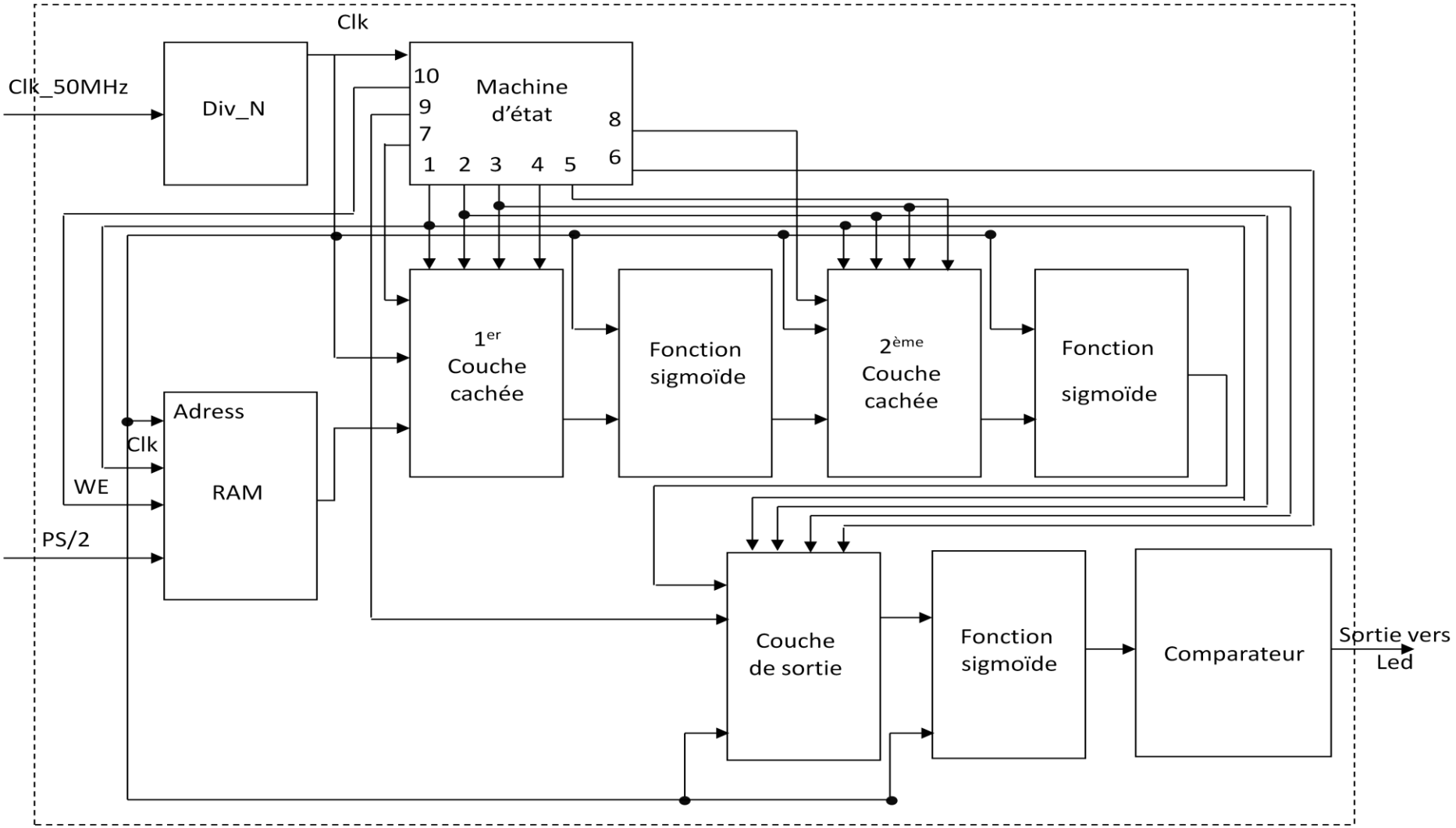
Notre structure est composée d'une couche d'entrée de 8 neurones (les huit paramètres de la base de données PIMA), deux couches cachées chacune de 7 et 5 neurones respectivement (obtenus après plusieurs tests) et une couche de sortie qui contient un seul neurone (la classe).

L'architecture proposé dans cette étude et après effectué plusieurs apprentissages sous l'environnement Matlab l'architecture adopté et qui possède un compromit entre les différents performances, une architecture de deux couches caché qui possèdent respectivement 7 et 5 neurones, pour la première et la deuxième couche .En utilisant les 8 paramètres nous somme parvenus à un taux de classification de 86.15%, avec une sensibilité de 87.91%, et une spécificité égale à 82.05%.

Chaque couche de ce réseau dispose d'une entrée, qui est connectée à tous les neurones de cette couche, et chaque sortie passe successivement à la fonction sigmoïde en série en activant le signal (Out_Enable), les neurones dans une seule couche fonctionnent en parallèle.

L'architecture globale de notre réseau de neurone est illustrée par la Figure34 :

Figure 31: Architecture globale d'un réseau de neurone.



L'architecture matérielle contient un bloc de diviseur de fréquence qui va résulter une horloge adapté à chaque bloc, une Ram pour mémoriser les entrées (les 8 descripteurs de la base de données PIMA) qui seront positionnées selon des adresses, un bloc de la 1er couche cachée (contient 7 neurones), un bloc pour la 2ème couche cachée (contient 5 neurones) et un bloc de la couche de sortie qui contient un neurone, des blocs de la fonction d'activation sigmoïde misent après chaque couche (travaux en cours), un comparateur qui va comparer la valeur de sortie (le sujet est sain ou diabétique).

Les signaux de commande dans l'architecture sont générés par une machine d'état, ce bloc contrôle l'ensemble des opérations du réseau :

- La sélection de la lecture (entrées provenant d'un clavier relié avec la carte NEXYS2 par le contrôleur PS/2) ou de l'écriture de la RAM [We (10)].
- La sélection des ROM qui se trouvent dans chaque neurone [Cs (3)].
- L'enchainement des adresses [(1)] pour la RAM ainsi que pour les ROM.
- Le chargement des biais dans les accumulateurs de chaque neurone en activant la ligne de commande du multiplexeur (2 vers 1) [Load (2)].
- L'activation de la 1ère couche cachée après 8 (le nombre de neurones d'entrée) cycles d'horloge [Enable_1(4)], de la 2ème couche après 7 cycles (le nombre de neurones de la 1ère couche cachée) [Enable_2 (5)] et de la couche de sortie après 5cycles d'horloge (le nombre de neurones de la 2ème couche cachée) [Enable_3 (6)].
- Garder les signaux [Out_Enable1 (7), (8), (9)] à un état actif pour pouvoir effectuer des opérations d'additions.

Cette machine d'état est conçue à l'aide d'un code générique VHDL (afin qu'il puisse être appliqué facilement à d'autre configuration de réseau).

4.2. Conception par l'outil de modélisation (System Generator):

Système générateur (*System Generator*) est un outil de modélisation au niveau de Matlab Simulink pour la conception des systèmes implémentés sur FPGA. Il dispose d'une bibliothèque de blocs très variés, qui peut être compilé automatiquement dans un FPGA. Le générateur du système peut générer des représentations équivalentes de la conception, au même niveau ou de niveau inférieur. Par exemple, du domaine fonctionnel, il peut générer une représentation structurelle, un HDL ou un fichier netlist, ou une représentation physique comme un flux de bits de configuration FPGA.

Il peut également générer un module équivalent de haut niveau qui effectue une fonction spécifique dans les applications externes au système générateur [20].

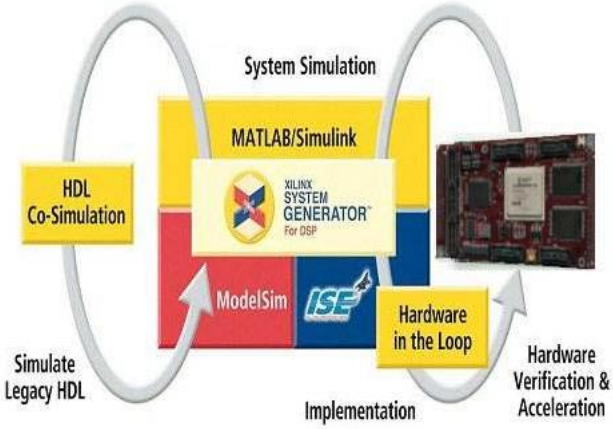


Figure 32: Système générateur.

4.2.1. La modélisation du classifieur neuronale :

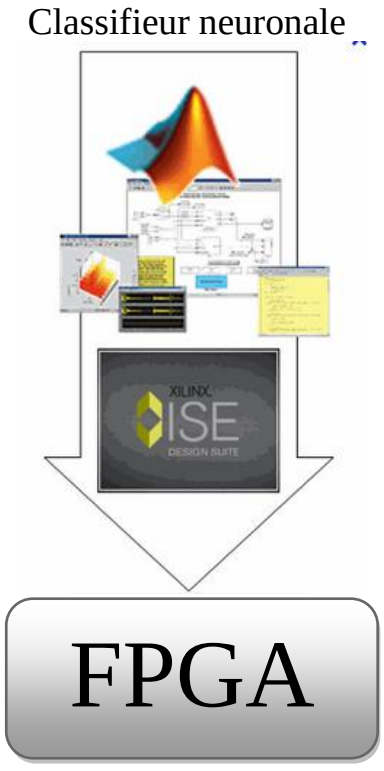


Figure 33: Implémentation du classifieur sur FPGA en utilisant System Generator.

[20] Xilinx, System Generator for DSP User Guide, Release 10.1, March 2008.

Une fonctionnalité très utile de Matlab Simulink utilisée dans ce travail est la possibilité de l'utiliser dans le développement d'un classifieur neuronal, l'utilisation des additionneurs, multiplicateurs, des portes logiques et des registres, ainsi que des composants prédéfinie pour générer une variété de fonctions et de sous programme modélisé en langage Matlab. Tous ces composants de base sont disponibles dans une bibliothèque fournie avec Simulink. Comme illustré dans la figure suivante :

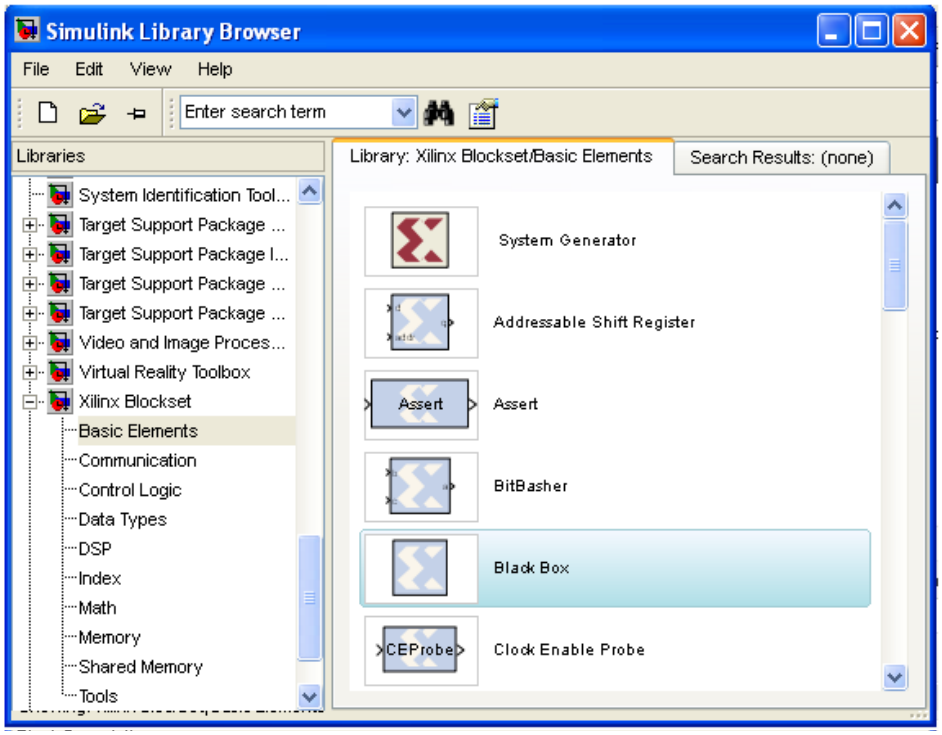


Figure 34: bibliothèque de système générateur sous simulink Matlab.

Une liste de boîte à outil professionnelle actuellement disponible peut être obtenue, cette liste est loin d'être statique car chaque année plusieurs boîtes à outils sont crée. Par exemple, il ya des boîtes à outils pour : les systèmes de communication, systèmes de contrôle, conception dans le domaine fréquentiel, et traitement de signal. Avec les composants et leur disposition facilement accessible pour l'utilisateur, il est possible de créer des modèles afin de procéder à une modélisation de conception appropriée et une étude de simulation.

5.2.1.1. La modélisation d'un neurone :

Pour réduire le temps de simulation au maximum il faut garantir un fonctionnement totalement parallèle de notre classifieur, nous avons modélisé un neurone par l'architecture suivante:

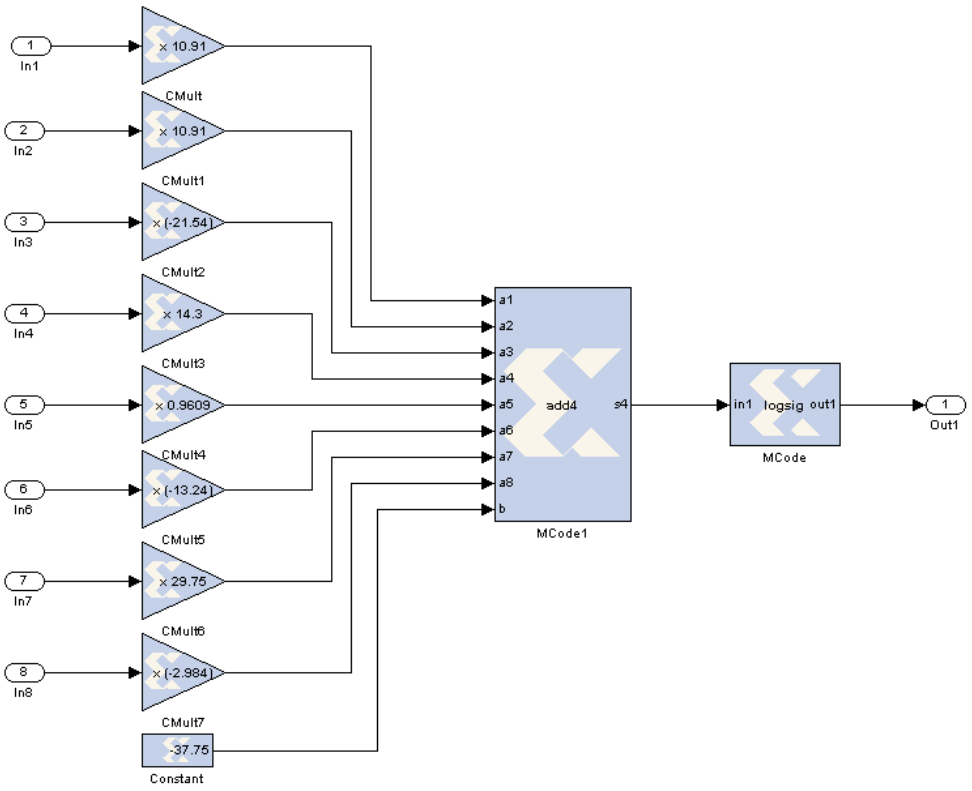


Figure 35: Un neurone modélisé avec le système générateur.

Dans cette architecture nous avons modélisé un neurone qui se situe dans la première couche cachée en utilisant trois blocs élémentaires suivants : un '*CMulte*', une '*constante*' et un '*Mcode*'. Ces blocs se trouvent dans la bibliothèque '*Xilinx Blockset*' de simulink. Dans chaque bloc existe un certain nombre de paramètres à préciser.

Le bloc '*CMulte'* est utilisé comme un support pour les poids synaptiques, la '*constante'* est utilisé pour charger le biais. Dans ces deux blocs il faut préciser un certain nombre de paramètres qui sont les valeurs des poids et des biais codées en décimale, le nombre de bits utilisé pour le codage binaire et la position de la virgule comme illustré dans la figure ci-dessous.

Pour notre application un codage de 16 bits est utilisé répartie comme suit : 7 bits pour les chiffres après la virgule, 8 bits pour la partie entière et 1bit pour le signe. Notant bien que les concepteurs du système générateur ont préféré d'utiliser un codage binaire en virgule fixe par rapport à la virgule flottante.

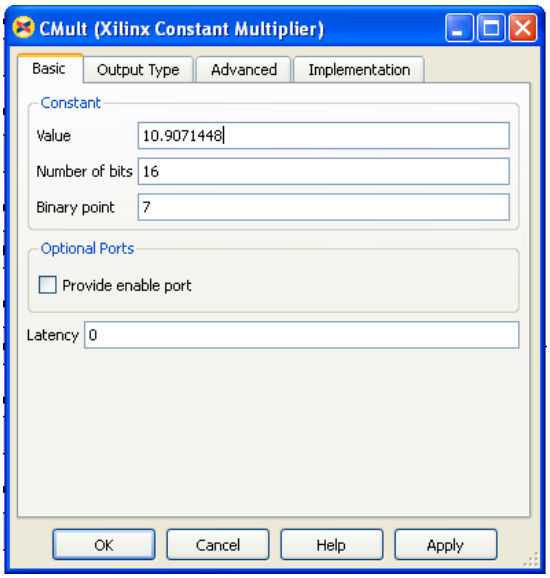


Figure 36: Onglet de paramètres pour le bloc CMult.

Le bloc 'MCode' possède une grande flexibilité car il permet à l'utilisateur non seulement de choisir une fonction prédéfinie dans l'environnement Matlab et en introduisant le nom de cette fonction, mais il permet aussi d'importer un sous programme de calcul personnalisé programmé en langage Matlab comme le montre la figure suivante.

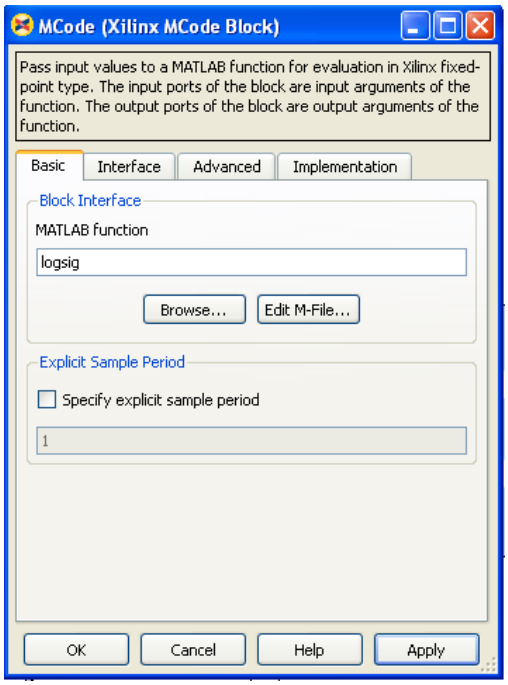


Figure 37: Onglet des paramètres pour le bloc MCode.

Dans notre application nous avons utilisé ce bloc comme un additionneur entre les valeurs pondérées et les biais, comme nous l'avons aussi adopté pour la fonction sigmoïde.

4.2.1.2. Architecture globale du classifieur:

Pour la classification du diabéte par les réseaux de neurones, nous avons proposé une architecture qui se compose de 8 neurones pour la couche d'entrée , 7 neurones pour la premiére couche cachée, 5 neurones dans la deuxiéme couche cachée et un neurone de sortie. La modélisation de la couche d'entrée est simple puisqu'elle posséde une fonction d'activation identité (F(x)= x), donc il suffie d'introduire les paramètres de notre base de données PIMA normalisée avec des bloc '*from file*'.

Pour pouvoir introduire ces paramètres dans notre réseau il faut les convertire en binaire, pour cela le système générateur nous offre le bloc '*Gateway In*' qui va faire cette conversion en virgule fixe comme le montre la figure suivante.

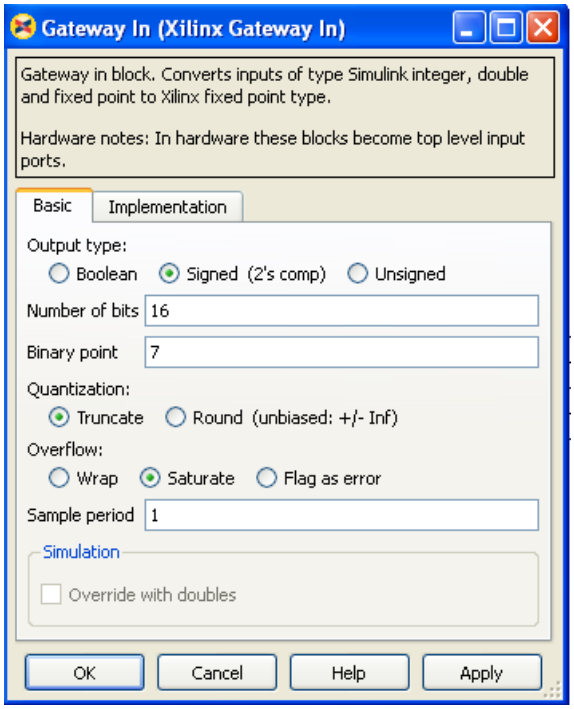


Figure 38: Onglet des paramètres pour le bloc Gateway In.

Une fois les données sont à l'intérieure de notre réseau à travers la couche d'entrée représenté par la couleur violet, ils vont se propager vers la première couche cachée, ensuite la deuxième couche et enfin la couche de sortie qui sont représentées dans la figure suivante respectivement par les couleurs : vert, bleu, et rouge. Il faut que chaque entrée doit être relié aux neurones de la couche suivante à travers une broche bien déterminée car elle doit se multiplier par leur poids respectifs.

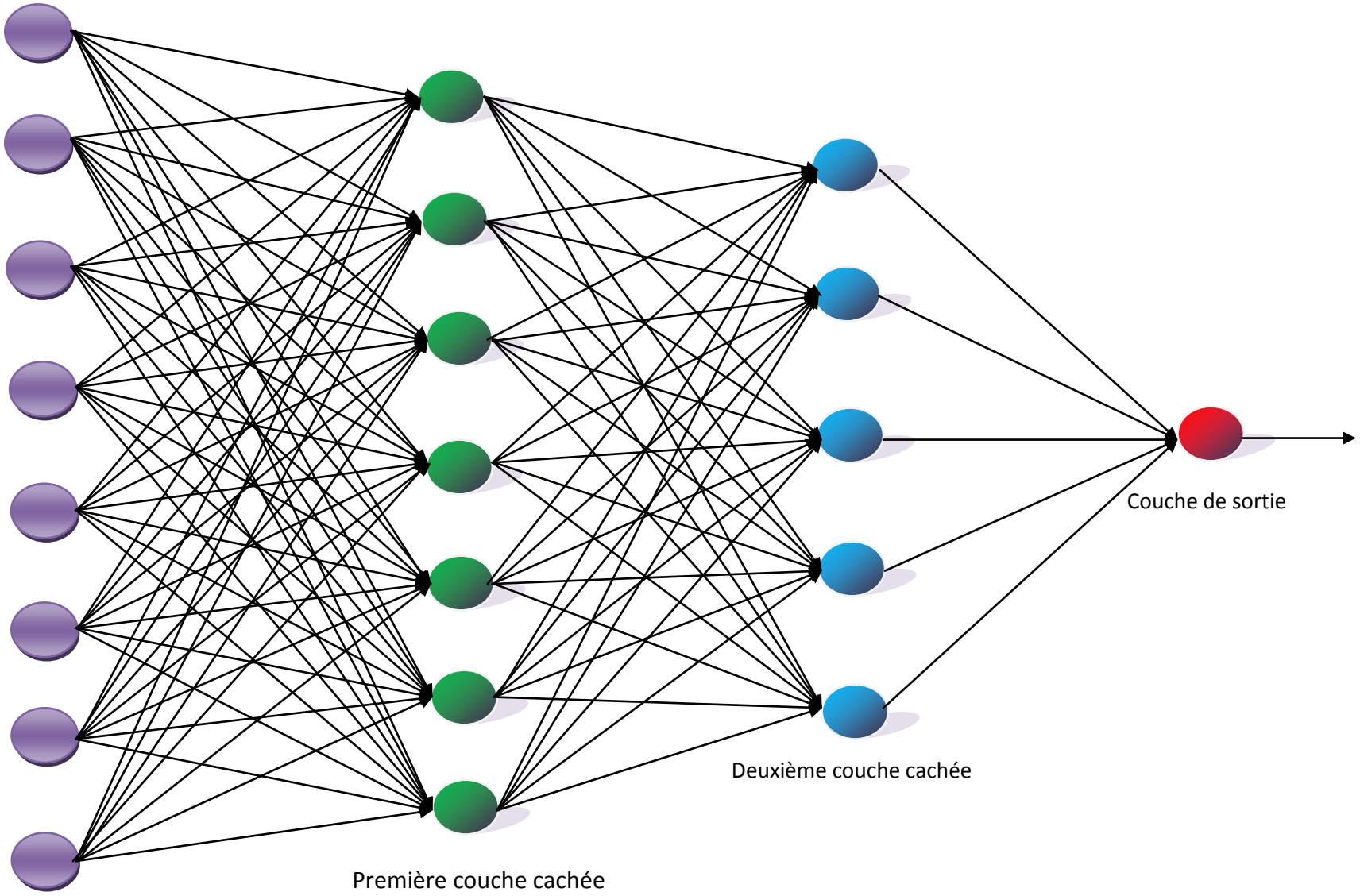


Figure 39: Architecture globale du classifieur neuronal.

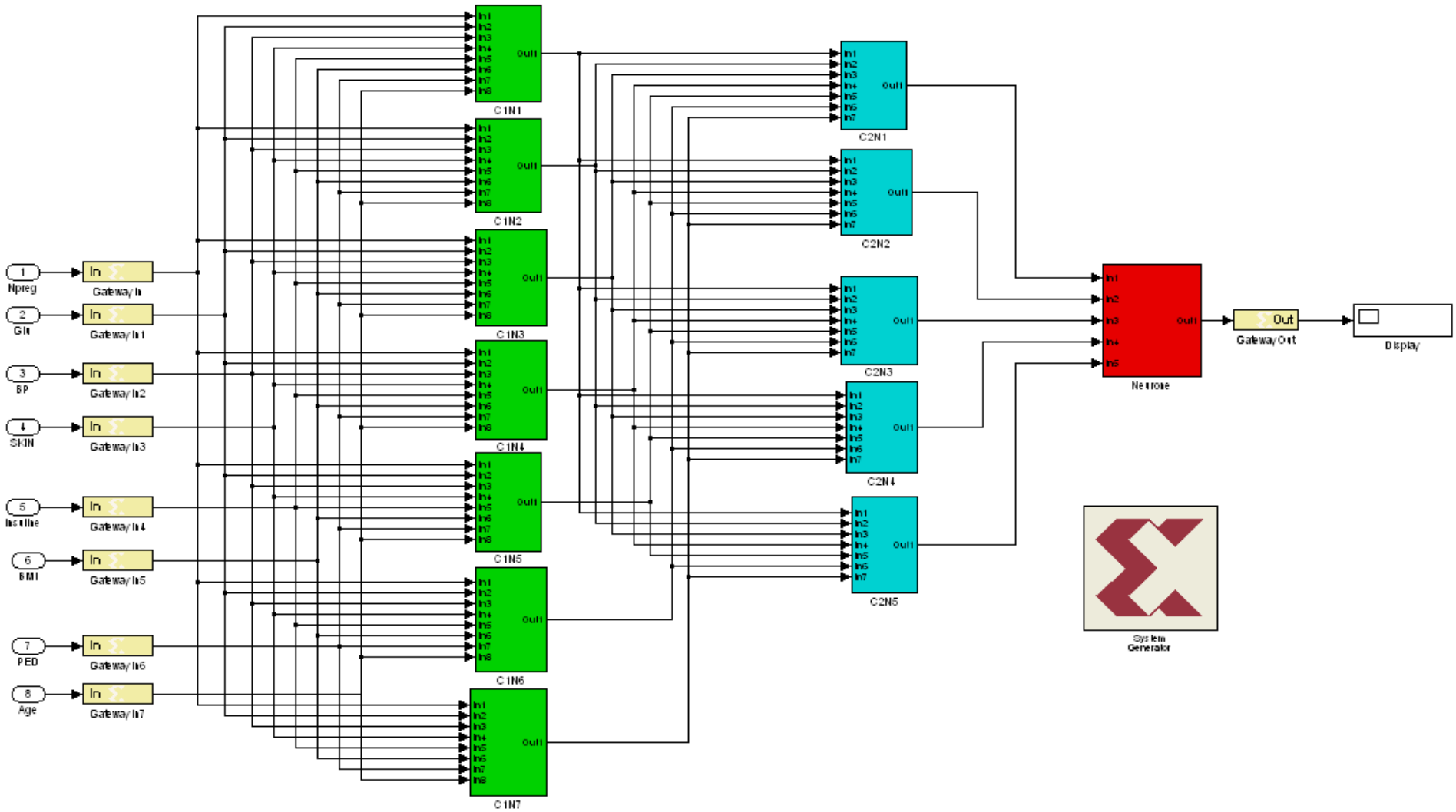


Figure 40 : Architecture globale du classifieur neuronal modélisée par le système générateur.

Une fois le résultat est présenté en sortie du réseau, on doit reconvertir la sortie du binaire au décimale, cette phase est effectué par le bloc ‘Gateway out’ pour pouvoir le comparer avec un seuil à priori. On doit fixer ce seuil pour trancher entre les cas diabétiques et non diabétiques.

Un bloc nommé ‘System Generator’ doit être figuré avec l’architecture modélisée, ce dernier nous permet de spécifier la carte FPGA et le langage utilisé pour l’implémentation de notre système afin de générer le fichier à implémenter sur notre carte à l’aide de l’outil ISE comme illustré dans la figure44.

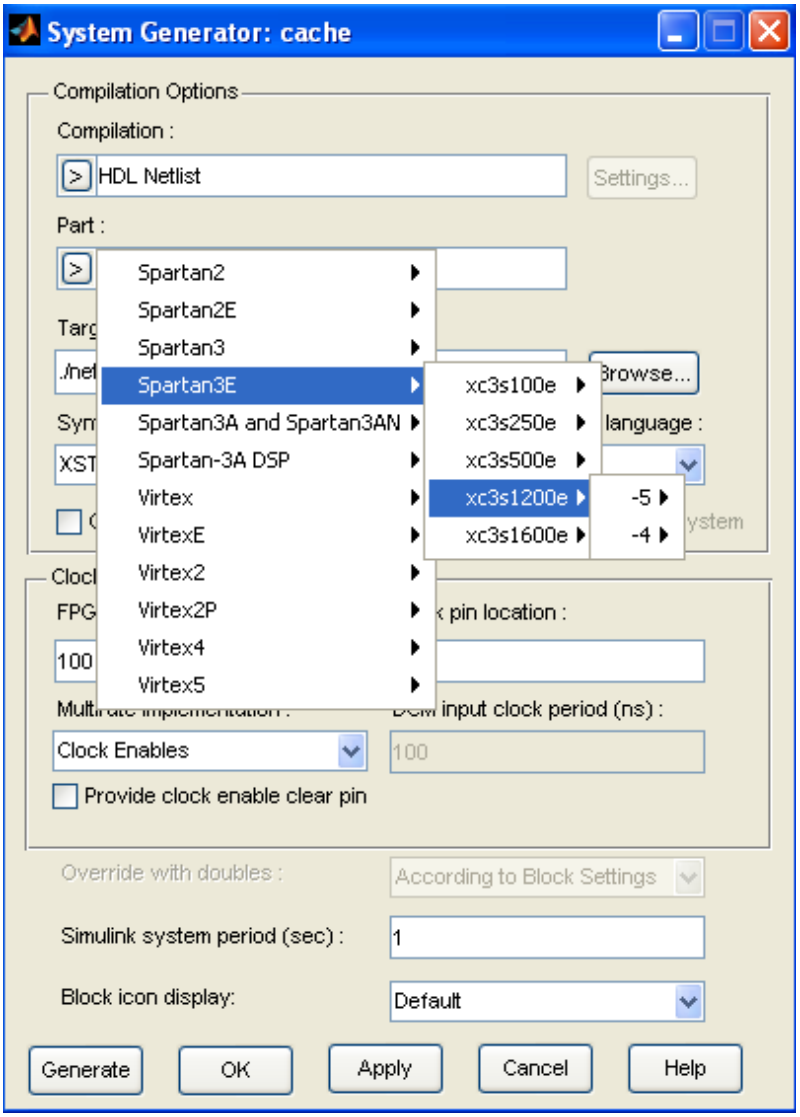


Figure 41: Onglet pour le choix du composant FPGA.

4.2.2. Le passage de la modélisation à l'implémentation :

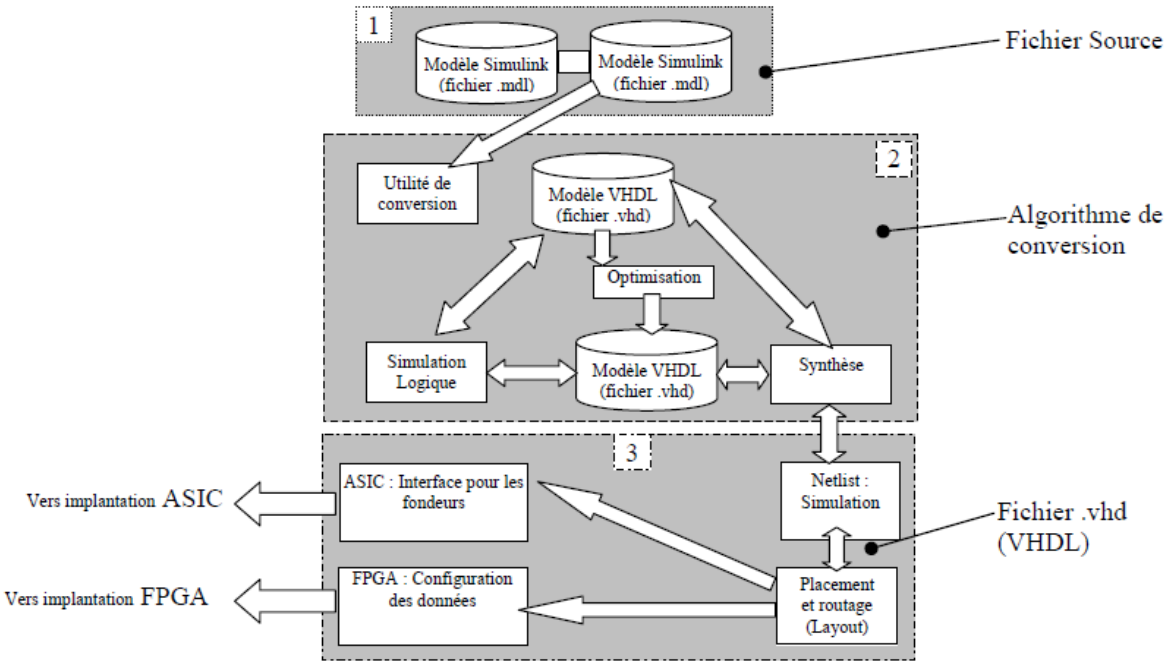


Figure 42: Description du passage des modèles de Simulink en modèle VHDL.

Pour chaque étape du procédé, le fonctionnement du système est vérifié, soit avec des réponses de type comportemental (échelon, sinusoïdale) pour la partie 1, soit avec des réponses de types logiques (0 ou 1) pour la partie 2 et 3. La base de conversion correspond aux éléments du système défini en fichier (*.mdl) sous Simulink. Seuls les éléments numériques seront pris en compte pour la conversion. Ces fichiers sont créés en terme de code VHDL c'est-à-dire avec des architectures et des entités. Le code est alors optimisé pour prédéfinir une architecture, cette dernière est alors vérifiée par une simulation numérique. Lorsque la phase de vérification est achevée, nous pouvons synthétiser le code VHDL, dont un fichier netlist est généré pour le placement/routage. C'est à partir de cette étape qu'on peut implémenter notre architecture sur le circuit FPGA[21].

[21] Céline Guilleminot, Etude et intégration numérique d'un système multicapteurs amrc de télécommunication basé sur un prototype virtuel utilisant le langage de haut niveau vhdl-ams, 2005, p : 27.

Système Générateur propose des mécanismes[22] :

- Pour importer à chaque bloc d'une boite noir son code VHDL dans une conception.
- Pour générer automatiquement un banc d'essai codé en VHDL, y compris les vecteurs de test. Ils permettent aussi à faire la comparaison des résultats de simulation entre Simulink et ModelSim.

- Pour effectuer à chaque circuit réalisé une simulation dans le Simulink suivi d'une implémentation matérielle sur le composant programmable.

Conclusion :

Dans ce chapitre, nous avons présenté une conception matérielle d'un réseau de neurones qui nous a permis de mettre en lumière une méthodologie de conception d'un système de classification dédié à la reconnaissance du diabète. Au cours de ce chapitre, nous avons développé deux réalisations différentes dont la première présente une architecture parallèle/série et la deuxième totalement parallèle par deux approches afin d'aboutir à une implémentation simple et efficace tout en réduisant aux maximumx le temps d'exécution.

Dans la 1ère architecture, nous avons utilisé un outil de CAO (ISE 10.1de Xilinx), et dans la seconde été conçue par le Système générateur de Simulink de Matlab.

[22] Sandro Noto, From Matlab/Simulink to ISE, 2008, P: 7.

Conclusion générale et perspectives :

L'implémentation des circuits sur les composants reconfigurables FPGA est un sujet d'actualité, mais d'une très grande complexité dans le domaine de la recherche, car elle nécessite non seulement une maîtrise des technologies relatives aux *FPGA* mais aussi une très bonne connaissance des applications et de leurs environnements.

Les réseaux de neurones ont prouvé leur efficacité grâce à leur puissance d'apprentissage et ceci en donnant les meilleurs résultats d'évaluation. Mais l'emploi des algorithmes consommateurs de ressources mémoires dans ces réseaux nécessitent un temps très important qui est précieux dans le domaine médical. Pour cela, nous avons proposé dans le cadre de ce mémoire l'utilisation des circuits reprogrammable FPGA qui apportent une grande flexibilité d'emploi et qui permettent une réduction de temps en assurant un fonctionnement parallèle tout en donnant une dynamique de reconfiguration, ce qui est avantageux par rapport aux autres circuits.

Dans cette étude, nous avons proposé une première architecture parallèle/série développée avec les outils de CAO et une deuxième architecture qui assure un fonctionnement totalement parallèle modélisée par le système générateur de Xilinx sous l'environnement Simulink de Matlab. Ces deux architectures sont modélisées pour l'implémentation d'un réseau neuronal afin de reconnaître le diabète.

Les architectures que nous avons développées présentent certainement de nombreuses perspectives d'avenir intéressantes.

Dans le domaine médical, le temps est un paramètre très précieux pour la survie du patient, ce qui favorise l'utilisation des circuits reconfigurables FPGA pour la conception des appareils hautement performants à temps réel (capteur endoscopique, réalisation des détecteurs à temps réel de pathologies pour différentes maladies…).

L'utilisation des réseaux de neurones est un domaine très vaste à cause de la variété de ces réseaux et leurs algorithmes d'apprentissage, ce qui rend l'implémentation de ces derniers sur les circuits reconfigurable un axe de recherches très prometteur.

On peut envisager l'implémentation matérielle des algorithmes d'apprentissage afin d'exploiter la conception des réseaux de neurones uniquement à travers les outils de CAO, pour avoir des classifieurs à temps réel.

Références

1 Iron Grout, “Digital system design with FPGA and CPLDs”, 2008, p: 3.

2 MacMillen, , D., et al., ‘‘An Industrial View of Electronic Design Automation,’’ IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 19, No. 12, December 2000, p: 1428.

3 Baba Hamed Amel, Conception d’un contrôleur vidéo en technologie FPGA, 2010, p : 36.

4 Jean-Max DUTERTRE, Circuits Reconfigurables Robustes, 2002, p : 13-14.

5 Zahir Ait Ouali, Application des FPGA à la commande d'un moteur asynchrone, p :20-21.

6 D.SMITH « HDL Chip Design : A pratical Guide for Designing, Synthesis & Simulating Asics &FPGAs using VHDL or Verilog » Doone Pubns . ISBN : 0965193438.

7 Xun ZHANG, Contribution aux architectures adaptatives : étude de l'efficacité énergétique dans le cas des applications à parallélisme de données, 2009, page : 23.

8 Wayne WOLF : Computers as Components : Principles of Embedded Computer Systems Design. Morgan Kaufmann, 2000.

9 Altera INC. : Stratix ii devic handbook,. Rapport technique, Altera,San Jos,Calif, USA, 2005.

10 Xilinx INC. : two flows for partial reconfiguration : mode based or difference based. Rapport technique, Xilinx inc, sep. 2004.

11 C.ALEXANDRE, Circuits logiques programmables, 2009, page :24-25-26-27.

12 Simon Haykin, Neural networks, Second Edition,1999, page :23.

13 McCullochW. and PittsW. L.R. A logical calculus for the ideas immanent in nervous activity. *Bull.Math. Biophysics*, 1943, pages 115.133.

14 Rachid Ladjaj, 2002/2003, Ingénieur 2000, Filière informatique et réseau. Site internet : http://www.igm.univ-mlv.fr/~dr/XPOSE2002/Neurones/index.php?rubrique=Apprentissage.

15 Jean-Luc Bloechle, Réseau de Neurones Artificiels pour la classification des fontes Arabes et la distinction entre la langue Arabe et les langues Latines**,** Université de Fribourg, page :7, Juin 2003.

16 Yohann BÉNÉDIC, APPROCHE ANALYTIQUE POUR L'OPTIMISATION DE RÉSEAUX DE NEURONES ARTIFICIELS**,** 2007, page : 18-19.
17 Data base PIMA :http ://www.cs.umb.edu/ rickb/files/UCI/diabetes.arff.
18 Si Mahmoud KARABERNOU, « *Méthodologie De Conception Et Langages De Description De Matériel* », Ecole Nationale Supérieure De L'Electronique Et De Ses Applications (ENSEA), Septembre 2009.
19 Jean-Luc Beuchat*, Etude Et Conception D'opérateurs Arithmétiques Optimisés Pour Circuits Programmables*, THÈSE No 2426 (2001).
20 Xilinx, System Generator for DSP User Guide, Release 10.1, March 2008.
21 Céline Guilleminot, Etude et intégration numérique d'un système multicapteurs amrc de télécommunication basé sur un prototype virtuel utilisant le langage de haut niveau vhdl-ams, 2005, p : 27.
22 Sandro Noto, From Matlab/Simulink to ISE, 2008, P: 7.

RESUME

Les réseaux de neurones font partie d'un champ de recherche qui s'attaque à comprendre et réaliser des systèmes capables de copier le comportement du cerveau. Ceci est fait dans le but d'utiliser ce savoir dans le domaine de la recherche pour résoudre des problématiques du domaine médical et proposé des systèmes d'aide au diagnostique dans le développement du matériel et facilité la tache pour les experts dans cet axe. Ce travail de mémoire s'inscrit dans le cadre de développement d'un système à base d'FPGA (Field Programmable Gate Array). Notre objectif principal est de modeliser un classifieur neuronal sur FPGA pour la reconnaissance du diabète. La technologie des circuits logiques programmables FPGA est une alternative innovante par rapport aux processeurs de contrôle conventionnels (ASIC, DSP,microprocesseurs).C'est pour cette raison, les FPGA font l'objet de recherches actives. A cet effet, nous nous sommes posés comme objectif de modéliser un réseau de neurone artificiel par une logique configurable sur FPGA, et nous avons abouti à développer deux structures différents par le biais de différents approche de modélisation.

Mots clé : Réseaux de neurones, PMC, FPGA, CAO, ISE 10.1, Modélisation, Système générateur, VHDL.

Abstract

Neural networks are part of a research field that tackles to understand and to implement systems able to copy the brain behavior. This is made in order to use this knowledge in the research field to solve problems in the medical field and proposed systems diagnostic aid in the development of hardware and to facilate the task for experts in this axis. This research work is part of developing a system based on FPGA (Field Programmable Gate Array).Our main objective is to modelize a neural classifier on FPGA for the recognition of diabetes. The technology of programmable logic FPGA is an innovative alternative to conventional control processors (ASIC, DSP, microprocessors). For this reason, FPGAs are the subject of active research. For this purpose, our objective is to modelize an artificial neuron network with configurable hardware logic on FPGA, and we have reached with two different structures developed through different modeling approach.

Keywords: Neural Networks, PMC, FPGA, CAO, ISE 10.1, Modeling, generator system, VHDL.