République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Projet de Fin d'Etudes
pour l'obtention du diplôme
d'Ingénieur d'Etat en Electronique

Présenté par :
BOUDJEMA Mohammed
BELHAOUAS Nasreddine

Thème :

Commande d'une poursuite du point de puissance maximum (MPPT) par les Réseaux de Neurones et implémentation sur FPGA et DSP

Membres du jury :

M HADDADI Mourad	Professeur	Président
M AIT CHEIKH Mohamed Salah	Chargé de Cours	Rapporteur
M LARBES Chérif	Maître de Conférence	Examinateur
M^{me} HAMMAMI Latifa	Maître de Conférence	Examinatrice

Juin 2007

ملخص :

يهدف هذا العمل إلى دراسة المبدأ الأساسي لطريقة تعقب نقطة الاستطاعة القصوى باستعمال الشبكات العصبية (MPPT-neuronale) الخاصة بالألواح الكهروضوئية ,حيث تمّ تحديد العوامل المؤثرة على هذه النقطة.

قامت برامج " MATLAB " و "SIMULINK" بتفـنيد حسن أداء جهاز السيطرة و الذي سنقوم ببرمجته على الدارات المبرمجة FPGA باستعمال بطاقة التطوير "ميماك ديزاين " (Memec Design)
و على المعالج الرقمي للإشارة دي إس بي C6211 من فئة C6000.
كلمات مفتاحيه : نقطة الاستطاعة القصوى, الدارات المبرمجة FPGA ,لغة البرمجة VHDL , المعالج الرقمي للإشارةDSP , برنامج Composer Studio ’الكهرؤضوئي, الشبكات العصبية**.**

Résumé :

Ce travail s'intéresse à l’étude de la méthode de poursuite du point de puissance maximale par les réseaux de neurones (MPPT-neuronale) des générateurs photovoltaïques, qui a permis de déterminer les différents paramètres influençant le point de puissance maximale.

Des simulations effectuées sous MATLAB et "SIMULINK" confirment la bonne performance du contrôleur MPPT, qui sera implémenté sur le circuit FPGA en utilisant la carte de développement « Memec Design » et DSP TMS 320C6211de la famille C6000.

Mots clés : MPPT**,** Circuit FPGA, langage VHDL, DSP, Code Composer Studio, Photovoltaïque, réseaux de neurones.

Abstract:

This work is interested in the method of Power Point Maximum Tracking of neural network (neural MPPT) for photovoltaic generators, who allowed to determine the various parameters influencing on the maximum power point.

The simulations carried out under MATLAB and "SIMULINK " confirm the best performance of controller MPPT, and which will be implement on circuit FPGA by using the chart of development "Memec Design" and DSP TMS 320C6211 the C6000 family.

Keywords: MPPT, FPGA Circuit,VHDL Languge, DSP,Code Composer Studio, photovoltaic , neural network.

Dédicace

I dedicate this work to
My mother and father
My brothers (Mohammed, Abdelhamid , Abdelkarim)
And to my family
Also, to
All my friends.

NESREDDINE

Je dédie ce travail à :
Ma très chère mère, mon très cher père,
Mes frères,
Ainsi qu'à toute ma famille et mes amis.

HAMIDO

Remerciements

Au terme de ce travail, nous tenons à présenter nos remerciements

- *aux membres du jury, qui m'ont fait l'insigne honneur d'examiner ce travail .*

- *à mon encadreur, qui a sacrifié son temps pour m'accompagner dans l'aboutissement de ce sujet.*

- *Nous tenons à remercier également Monsieur LARBES Chérif, Madame HAMMAMI pour leur précieuse aide et leurs encouragements.*

- *A nos parents pour leur compréhension et pour ce qu'ils ont enduré pour nous offrir le bonheur et la confiance, nous disons notre gratitude.*

- *Nous ne saurons oublier de remercier toute les personnes qui, d'une manière ou d'une autre, nous ont aidés dans l'élaboration de ce travail.*

Table des matières

Table des figures

Introduction

L'industrie moderne a des besoins de plus en plus importants en énergie. Les sources classiques d'énergie, qui sont les sources fossiles telles que le charbon et les hydrocarbures, laissent progressivement la place aux énergies renouvelables. L'augmentation fulgurante du prix du pétrole ces dernières années a en effet contraint les pays développés à investir dans ce type d'énergies (l'énergie solaire, éolienne, marée motrice ou géothermique). Ces énergies, en plus d'être inépuisables, représentent un secteur porteur permettant un développement durable tout en préservant l'environnement.

L'énergie solaire représente certainement la source d'énergie renouvelable la plus élégante. En plus d'être silencieuse, elle s'intègre parfaitement aux constructions (façades, toitures ...), et du fait qu'elle n'intègre pas de pièces mécaniques mobiles, elle ne nécessite pas un entretien particulier et reste fiable longtemps.C'est la raison pour laquelle elle est devenue une référence dans les applications spatiales et dans les sites isolés. Elle est en train de s'imposer comme une valeur sure dans les applications à petite et moyenne consommation d'énergie, surtout depuis que les panneaux solaires sont devenus moins chers pour des rendements meilleurs.

Les panneaux solaires, bien qu'ils soient de plus en plus performants, ont des rendements qui restent assez faibles (autour de 20%).C'est pourquoi il faut exploiter le maximum de puissance qu'ils peuvent générer en réduisant au maximum les pertes d'énergie.

Une caractéristique importante de ces panneaux est que la puissance maximale disponible est fournie seulement en un seul point de fonctionnement appelé (Maximum Power Point) (MPP), défini par une tension et un courant donnés ; ce point se déplace en fonction des conditions météorologiques (ensoleillement, température, etc.) ainsi que des variations de la charge. Extraire le maximum de puissance nécessite donc un mécanisme de poursuite de ce point qu'on appelle MPP Tracker (MPPT).

Il existe plusieurs méthodes MPPT, allant de la plus simple qui utilise une adaptation manuelle aux plus complexes qui font appel à des algorithmes compliqués. On va s'intéresser à la méthode MPPT Neuronale qui est basée sur les réseaux de neurones.

D'autre part, les circuits FPGA (Field Programmable Gate Array), qui sont des circuits à architecture programmable, et qui peuvent être adaptés à des besoins divers, deviennent incontournables dans les applications nécessitant un temps de

développement rapide et une modularité garantie. Ils sont surtout utilisés dans les systèmes embarqués (avionique, automobile, espace,...) et tendent à se généraliser dans le domaine des applications on chip.

De même ,le DSP (Digital Signal Processor) fait partie d'un type particulier de microprocesseurs. Ses fonctions spéciales ainsi que son architecture le rendent performant dans le domaine du traitement du signal et de l'automatisme.

Dans la conception des systèmes de traitement numérique, contrairement aux processeurs classiques, tout système fondé autour d'un DSP bénéficie des avantages dérivant de ses particularités architecturales et de programmation.

L'objet de ce projet de fin d'études consiste à étudier la commande MPPT par les réseaux de neurones et l'implémentation de da méthode MPPT Neuronale sur les deux circuits FPGA et DSP.

Ce mémoire a été divisé en cinque chapitres comme suit :

Le premier chapitre est une introduction à l'énergie photovoltaïque ; on y explique brièvement les différents composants d'un système photovoltaïque avec batterie : le champ de modules photovoltaïques, le convertisseur continu-continu et les batteries .

Le second chapitre, après avoir introduit la théorie des réseaux de neurons, explique le principe de la méthode MPPT neuronale .

Dans le troisième chapitre, on aborde les circuits FPGA et on s'intéresse à leur architecture et leurs caractéristiques. On fait une description de la carte de développement Virtex-II qu'on a utilisé et on introduit le langage de description VHDL. On explique aussi les différentes étapes nécessaires pour développer un projet sur circuit FPGA .

Dans le quatrième chapitre, on aborde les processeurs DSP TMS320C6x et on s'intéresse à leurs architectures et leurs caractéristiques. On fait une description du kit d'évaluation DSK qu'on a utilisé. On explique aussi les différentes étapes nécessaires pour développer un projet sur ne DSP .

Le dernier chapitre est consacré à l'explication des simulations de SIMULINK, et laux étapes de développement des programmes du circuit FPGA et de DSP, ansi qui à la visualisation et l'interprétation des résultats obtenus par simulation et par test.

Chapitre 1

Système photovoltaïque

1.1 Introduction

Originalement conçue pour répondre aux besoins en énergie des capsules spatiales, l'énergie solaire photovoltaïque est de plus en plus utilisée pour opérer diverses applications terrestres comme l'éclairage, les télécommunications, la réfrigération et le pompage.
L'énergie solaire est disponible partout sur la planète en des degrés divers et elle est entièrement renouvelable. Son apport est variable, au gré des jours et des saisons, mais elle est relativement prévisible. Même si elle est relativement diluée, son apport énergétique annuel pourrait répondre à la consommation énergétique de la plupart des pays.
Les systèmes photovoltaïques ne nécessitent aucun apport extérieur de combustible ; de plus, le générateur lui-même ne contient aucune pièce mobile et ne requiert donc pratiquement pas d'entretien.
Par conséquent, les coûts récurrents d'opération et de maintenance sont relativement faibles.
Pour ces raisons, cette source d'énergie convient particulièrement bien pour les utilisations en milieu rural où les populations sont réparties dans de petites communautés et où la demande énergétique est relativement faible. Elle est très utilisée en Afrique sahélienne notamment pour le pompage de l'eau dans les zones arides.
Ce chapitre décrit les concepts de base de système photovoltaïque et de la production d'électricité grâce à l'effet photovoltaïque. On y explique comment l'énergie solaire varie selon l'endroit et la saison.
Les principaux éléments du système photovoltaïque sont étudiés, allant du panneau photovoltaïque passant par le contrôleur et terminons par les batteries.

1.2 L'effet photovoltaïque

L'énergie photovoltaïque PV est la transformation directe de la lumière en électricité.
Elle utilise une photopile pour transformer directement l'énergie solaire en électricité.
L'effet photovoltaïque, c'est-à-dire la production d'électricité directement de la lumière, fut observé la première fois, en 1839, par le physicien français Edmond Bec-

querel.
Toutefois, ce n'est qu'au cours des années 1950 que les chercheurs de la compagnie Bell Telephone, aux États-Unis, parvinrent à fabriquer la première photopile, l'élément primaire d'un système photovoltaïque.

1.3 La photopile

Cette photopile, qu'on appelle aussi cellule solaire ou photovoltaïque, est fabriquée à l'aide de matériaux semi-conducteurs.
On peut la représenter comme une diode plate qui est sensible à la lumière.
Quand un photon, d'énergie suffisante, heurte un atome sur la partie négative de

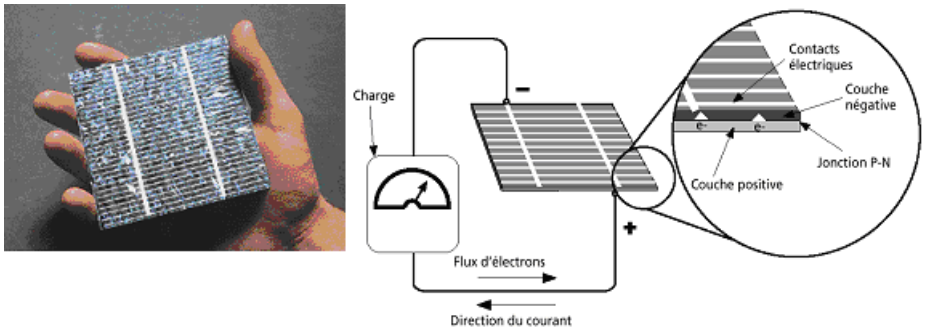

FIG. 1.1 – Description d'une photopile ou cellule photovoltaïque

cette diode, il excite un électron et l'arrache de sa structure moléculaire, créant ainsi un électron libre sur cette partie.
Une photopile est fabriquée de manière à ce que cet électron libre ne puisse pas se recombiner facilement avec un atome à charge positive, avant qu'il n'ait accompli un travail utile en passant dans un circuit extérieur.
La cellule photovoltaïque produira de l'électricité à courant continu (CC), et son énergie produite sera fonction principalement de la lumière reçue par la photopile.

1.4 Caractéristiques de la cellule photovoltaïque

Le schéma du circuit équivalent d'une cellule photovoltaïque qui est largement utilisé dans la littérature [31], est représenté sur la figure.1.2 .
Comme le montre la figure.1.2 une photopile comporte en réalité une résistance

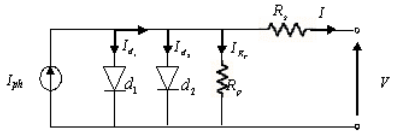

FIG. 1.2 – Schéma du modèle équivalent à deux diodes d'une cellule photovoltaïque

série Rs et une résistance en dérivation ou shunt R_p.
Ces résistances auront une certaine influence sur la caractéristique I-V de la photopile :

- la résistance série est la résistance interne de la cellule ; elle dépend principalement de la résistance du semi-conducteur utilisé, de la résistance de contact des grilles collectrices et de la résistivité de ces grilles .
- la résistance shunt est due à un courant de fuite au niveau de la jonction ; elle dépend de la façon dont celle-ci a été réalisée.

D'après la figure.1.2. Le modèle mathématique pour la caractéristique courant-tension est :

$$I = I_{ph} - I_{s1}(e^{\frac{q(V-R_sI)}{\eta_1 KT}} - 1) - I_{s2}(e^{\frac{q(V-R_sI)}{\eta_2 KT}} - 1) - \frac{V + R_sI}{R_p} \tag{1.1}$$

ou :
I et V sont le courant et la tension de sortie de la cellule photovoltaïque,
I_{ph} : est le photo-courant produit,
I_{s1} et I_{s2} : sont les courants de saturation des diodes,
η_1 et η_2 : les facteurs de pureté de la diode,
R_s et R_p :sont respectivement la résistance série et la résistance parallèle,
T :est la température absolue en Kelvin,
q : est la charge élémentaire constante ,
k :est la constante de Boltzmann.
Le photo-courant est atteint à une insolation maximum. Souvent on a :

$$I_{ph} = SI_{ph.max} \tag{1.2}$$

S : pourcentage d'insolation.
Il est évident de l'équation (1.1), que la caractéristique courant-tension dépend fortement de l'insolation et de la température.
La dépendance de la température est encore amplifiée par les propriétés du photo-courant et les courants de saturation inverse des diodes qui sont donnés par [36] :

$$I_{ph}(T) = I_{(ph.T=29)}.(1 + (T - 29)(5.10^{-4})) \tag{1.3}$$

$$I_{s1} = K_1 T^3 e^{-\frac{E_g}{KT}} \tag{1.4}$$

$$I_{s2} = K_2 T^{\frac{5}{2}} e^{-\frac{E_g}{KT}} \tag{1.5}$$

Où :
E_g est la bande d'énergie du semi-conducteur avec
K_1=1.2 A/cm^2K^3
K_2=2.9 $10^5 A/cm^2K^3$

1.5 Influence de l'ensoleillement et de la température

Le courant produit par la photopile I_{ph} est pratiquement proportionnel à l'éclairement solaire S. Par contre, la tension V aux bornes de la jonction varie peu car elle est fonction de la différence de potentiel à la jonction N-P du matériau lui-même. La tension de circuit ouvert ne diminuera que légèrement avec l'éclairement.
Cela implique donc que :

– la puissance optimale de la cellule (P_m) est pratiquement proportionnelle à l'éclairement.
– les points de puissance maximale se situent à peu près à la même tension (figure.1.3).

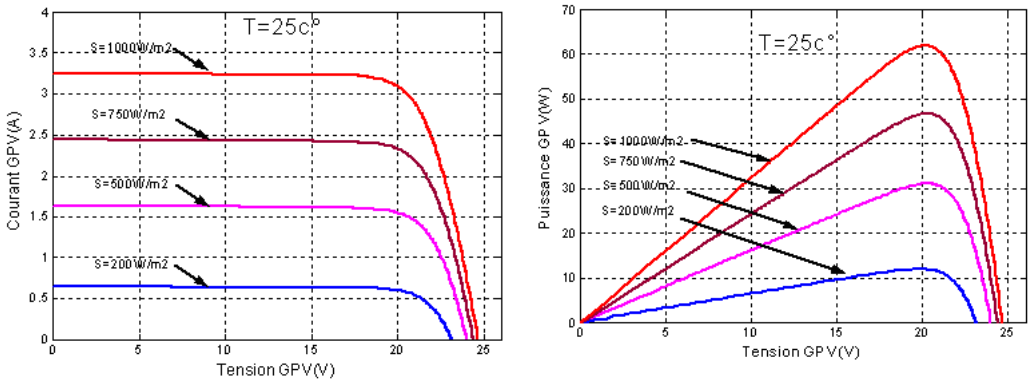

FIG. 1.3 – Influence de l'ensoleillement sur les courbes I-V et PV

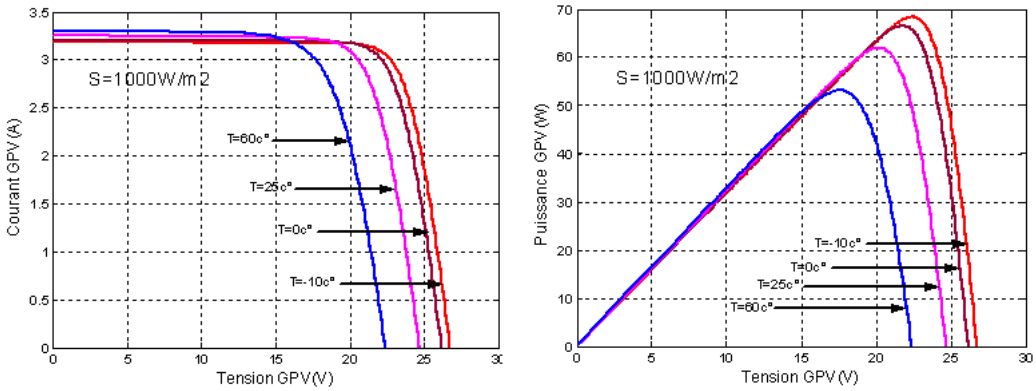

FIG. 1.4 – Influence de la température sur les courbes I-V et PV

L'influence de la température est non négligeable sur la caractéristique courant/tension d'un semi-conducteur (voir figure.1.4).
Pour le silicium, lorsque la température augmente, le courant augmente d'environ 0,025 $mA/cm^2/C$ alors que la tension décroît de 2,2 $mV/C/cellule$.
Cela se traduit par une baisse de puissance d'environ 0,4 /C.

1.6 Le module photovoltaïque

Afin d'augmenter la tension d'utilisation, les cellules PV sont connectées en série. La tension nominale du module est habituellement adaptée à la charge de 12 volts et les modules auront donc généralement 36 cellules. De plus, la fragilité des cellules au bris et à la corrosion exige une protection envers leur environnement et celles-ci sont généralement encapsulées sous verre ou sous composé plastique. Le tout est appelé module photovoltaïque.
Les modules peuvent également être connectés en série et en parallèle afin d'augmenter la tension et l'intensité d'utilisation.

1.7 Modèle mathématique d'un panneau photovoltaïque

La considération du modèle de circuit équivalent la figure.1.2 mène à l'équation.1.6 pour une rangée photovoltaïque de cellules (généralement considéré comme un panneau solaire), avec z cellules photovoltaïques raccordées en série [36].

$$I = I_{ph} - I_{s1}(e^{\frac{q(V-R_sIZ)}{\eta_1 KT}} - 1) - I_{s2}(e^{\frac{q(V-R_sIZ)}{\eta_2 KT}} - 1) - \frac{V + R_sIZ}{R_p} \tag{1.6}$$

Les modules PV sont les éléments de base de tout système photovoltaïque. Ils peuvent être branchés en série pour augmenter leur tension d'utilisation et en parallèle pour augmenter leur courant. Cet ensemble est appelé champ de modules PV.

1.8 Contexte de l'étude sur les systèmes de conversion photovoltaïques (PV)

Plusieurs études ont montré qu'un certain nombre d'améliorations pouvaient être faites pour augmenter le rendement et la fiabilité de ce type de systèmes de conversion, dont un grand nombre étaient déjà commercialisés. En effet, si l'on considère le fonctionnement d'un générateur PV à base de cellules en silicium, comme il est décrit dans la première partie de ce chapitre, les premières améliorations à effectuer pour gagner en rendement sont les suivantes .

- 1. Améliorer le refroidissement des cellules PV,ce qui permettrait de gagner environ 20% de rendement sur la totalité des capteurs PV par un refroidissement judicieux. En effet, le rendement de conversion photons-électrons décroît rapidement au fur et à mesure que la température interne de la cellule augmente.
- 2. Améliorer la connectique entre les différents panneaux PV par une architecture appropriée. En effet, une perte de rendement de 20% supplémentaire est à noter sur les systèmes existants entre la puissance qu'ils pourraient théoriquement délivrer et celle effectivement transférée à la charge. Cette perte est communément imputée à l'interface de puissance entre les panneaux et la charge.

On peut remarquer aussi que la puissance de sortie d'un panneau solaire ne dépend pas uniquement de la température et de l'insolation, mais aussi de la tension V de fonctionnement.
Le point de la puissance maximale indiqué comme MPP (maximum power point) sur la figure.1.5 est le point désiré pour le fonctionnement d'une rangée photovoltaïque pour obtenir un rendement maximal en puissance.
Les valeurs correspondantes pour la tension et le courant s'appellent respectivement V_{OP} et le I_{OP}.
Les courbes de puissances représentées respectivement sur les figures.1.3,4 , montrent explicitement que la puissance de sortie d'un module solaire P_{OP} est subordonnée directement aux changements de l'intensité de lumière incidente sur le module solaire, ainsi qu'à la température de l'environnement. En conséquence la tension V_{OP} et le courant I_{OP} de fonctionnement changent constamment avec le changement de

ces variables environnementales.
Pour remédier à ces contraintes, un équipement spécifique, doit être mis entre le module solaire et la charge dont le rôle est l'adaptation des deux équipements (module solaire/charge) pour un meilleur transfert d'énergie vers la charge.
Ce dernier peut augmenter d'une manière significative le rendement de puissance et cela en ajustant la charge du système de telle manière que la tension V de service soit toujours approximativement égale à la tension V_{OP} de fonctionnement optimal. Sachant qu'un écart de 10 % de la tension de service V par rapport à la tension optimale V_{OP} enregistre une perte de puissance de service P de presque 25 % de la puissance optimale P_{OP} (figure.1.5). Toute installation photovoltaïque dotée d'un

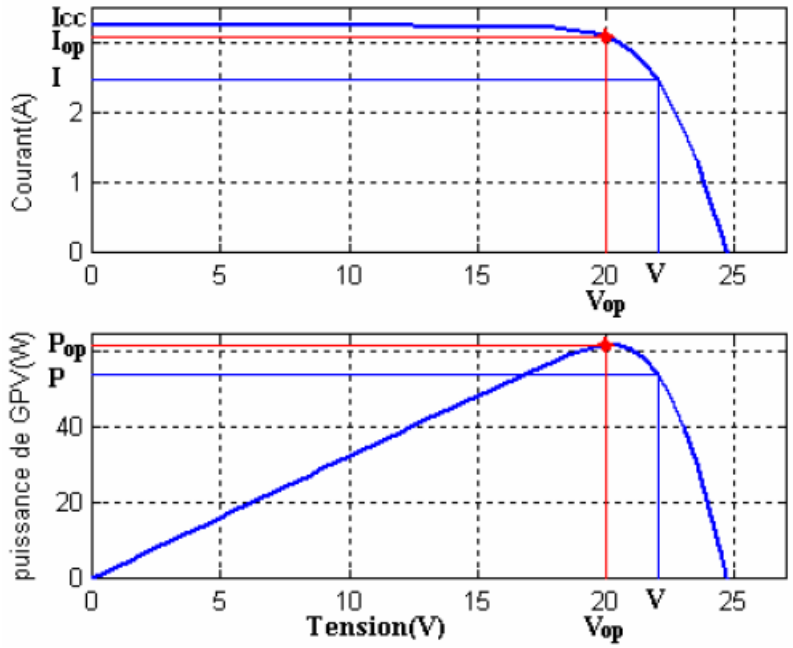

FIG. 1.5 – Point de puissance maximale (PPM), puissance et courant correspondant à V_{OP}

équipement adéquat (contrôleur MPPT) va lui assurer un fonctionnement optimal en un endroit quelconque de la surface énergétique correspondant à un changement donné en température ou en ensoleillement ou bien les deux simultanément.

1.9 Les systèmes photovoltaïques avec batterie

Un système photovoltaïque avec batterie peut être comparé à une charge alimentée par une batterie qui est chargée par un générateur photovoltaïque. Il comprend généralement les composantes de base suivantes :

- un générateur photovoltaïque composé d'un ou de plusieurs modules photovoltaïques pour charger les batteries,
- une batterie pour stocker l'énergie électrique et alimenter les charges à courant continu (DC) ,
- un convertisseur continu-continu (hacheur) pour fournir la tension d'alimentation adéquate pour les batteries ,

– un convertisseur continu-alternatif (onduleur) pour alimenter les charges à courant alternatif.

La figure.1.6 illustre un système photovoltaïque simple avec batterie.

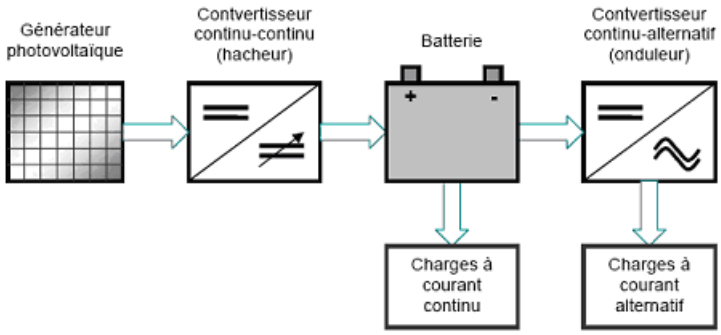

FIG. 1.6 – Composantes de base d'un système PV autonome avec batterie

1.10 Le convertisseur DC-DC (les hacheurs)

Les hacheurs sont des convertisseurs statiques continu-continu permettant de générer une source de tension continue variable à partir d'une source de tension continue fixe.

Ils se composent de condensateurs, d'inductances et de commutateurs.

Tous ces dispositifs ne consomment aucune puissance dans le cas idéal ; c'est pour cette raison que les hacheurs ont de bons rendements.

Généralement le commutateur est un transistor MOSFET qui travaille en mode bloqué-saturé. Si le commutateur est bloqué, son courant est nul, il ne dissipe donc aucune puissance . S'il est saturé, la chute de tension à ses bornes sera presque nulle et par conséquent la puissance perdue sera très petite.

Dans cette étude, l'interrupteur du convertisseur est attaqué par un signal MLI (Modulation Large Impulsion), avec une fréquence F_s fixe et un rapport cyclique d variable.

Il existe différents types de convertisseurs continu-continu (DC-DC). Les trois types

FIG. 1.7 – Tension de commande (PWM) du commutateur durant une période de commutation

les plus fréquemment utilisés dans les systèmes photovoltaïques sont le convertisseur Buck, le Boost et le Buck-Boost.

Nous allons faire dans ce qui suit une présentation du principe de convertisseurs Buck-Boost, utilisés dans notre étude.

1.10.1 Principe de fonctionnement du convertisseur Buck-Boost

La figure.1.8 donne le circuit électrique d'un convertisseur Buck-Boost. Le transistor MOSFET travaille en régime de commutation avec une période Ts. En premier temps, S est fermé et la tension de la source est appliquée aux bornes de l'inductance L, où elle se charge d'énergie jusqu'au début de la deuxième phase de fonctionnement, puis S s'ouvre et la tension de l'inductance se trouve appliquée à la charge, où son courant circule dans le sens inverse des aiguilles de montre à travers la diode D et ainsi la tension de sortie sera négative [1].

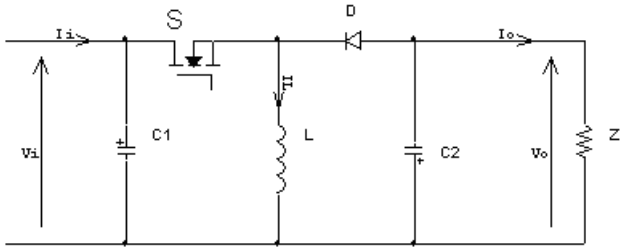

FIG. 1.8 – Circuit électrique d'un convertisseur Buck-Boost

1.10.2 Modèle mathématique équivalent

La figure.1.9 montre les deux schémas équivalents du convertisseur Buck-Boost pour les deux périodes de fonctionnement. En appliquant les lois de Kirchhoff sur

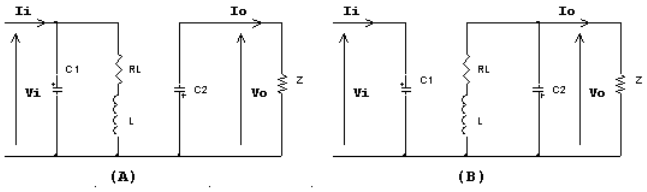

FIG. 1.9 – Circuits équivalents de Buck-Boost, (A) :S fermer, (B) :S ouvert

les circuits équivalents précédents, on obtient :
Pour la première période D T_s :

$$\begin{cases} i_{s1} = C_1 \frac{dV_i}{dt} = i_i - i_L \\ i_{s2} = C_2 \frac{dV_0}{dt} = i_0 \\ i_L = L \frac{dV_L}{dt} = v_i - R_L i_L \end{cases} \tag{1.7}$$

[1] Le convertisseur Buck-Boost combine les propriétés des deux convertisseurs Buck et le Boost. Il est utilisé comme un transformateur idéal de n'importe quelle tension d'entrée pour n'importe quelle tension de sortie désirée

Et pour la deuxième période (1-D) T_s :

$$\begin{cases} i_{s1} = C_1 \frac{dV_i}{dt} = i_i \\ i_{s2} = C_2 \frac{dV_0}{dt} = -i_0 - i_L \\ i_L = L \frac{dV_L}{dt} = v_0 - R_L i_L \end{cases} \tag{1.8}$$

on trouve le modèle approximé du convertisseur Buck-Boost :

$$\begin{cases} C_1 \frac{dV_i}{dt} T_s = DT_s(i_i - i_L) + (1-D)T_s i_i \\ C_2 \frac{dV_0}{dt} T_s = -DT_s i_0 + (1-D)T_s(-i_i - i_L) \\ L \frac{di_L}{dt} T_s = DT_s(v_i - R_L i_l) + (1-D)T_s(v_0 - R_L i_l) \end{cases} \tag{1.9}$$

En arrangeant les termes des équations précédentes, on obtient la modélisation dynamique du Buck Boost :

$$\begin{cases} i_L = \frac{1}{D}(i_i - C_1 \frac{dV_i}{dt}) \\ i_0 = -(1-D)i_L - C_2 \frac{dV_0}{dt} \\ V_i = \frac{1}{D}(-(1-D)V_0 + R_L i_L + L \frac{di_L}{dt}) \end{cases} \tag{1.10}$$

1.10.3 Les ondulations des courants et des tensions

En suivant les mêmes procédures précédentes, on trouve les mêmes résultats que pour le circuit Boost. Les valeurs crête à crête des courants et des tensions sont :

$$\begin{cases} i_{LCC} = 2\Delta i_L = \frac{V_i - R_L i_L}{L} DT_s \\ V_{iCC} = 2\Delta V_{C1} = \frac{i_i - i_L}{C1} DT_s \\ V_{0CC} = 2\Delta V_{C2} = -\frac{i_0}{C2} DT_s \end{cases} \tag{1.11}$$

1.11 Les batteries

Le système tampon utilisé le plus couramment pour les systèmes photovoltaïques est la batterie d'accumulateurs électrochimiques. Les deux types de batteries utilisés le plus couramment dans les systèmes photovoltaïques sont les batteries avec accumulateurs au plomb-acide (Pb acide) et les batteries avec accumulateurs au nickel-cadmium (Ni-Cd).
La batterie au plomb-acide est la plus connue, étant utilisée depuis plus de 150 ans pour fournir le courant de démarrage des voitures.
Une modélisation du circuit électrique de la batterie plomb-acide a été proposée dans la littérature [37], qui est la suivante :
C_{bp} est la capacité électrochimique de la batterie ;elle est donnés par l'expression générale de l'énergie :

$$E_C = \frac{1}{2} C V_C^2 \tag{1.12}$$

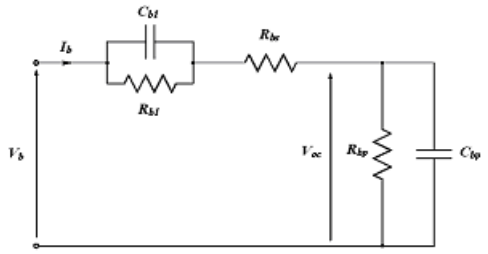

FIG. 1.10 – Modèle équivalent de batterie

avec C étant la capacité et la tension du condensateur.
À la différence d'un condensateur, la tension d'une batterie ne sera pas égale à zéro à son plus bas état de charge.
C'est équivalent à un condensateur ayant un niveau minimum de charge égal à l'énergie .
La capacité entièrement chargée de la batterie est représentée par un niveau maximum de charge .
Ceci est indiqué par l'équation suivante [37] :

$$E_b = E_{Cmax} - E_{Cmin} = \frac{1}{2}CV_{max}^2 - \frac{1}{2}CV_{min}^2 = \frac{1}{2}C_{bp}(V_{max}^2 - V_{min}^2) \tag{1.13}$$

L'énergie E_b est donnée par le constructeur de la batterie directement en kilowatt par heure (kWh).
Les tensions sont la tension maximale et la tension minimale de la batterie en circuit ouvert respectivement.
La conversion mène finalement à une expression pour le condensateur représentant la capacité de la charge de la batterie

$$C_{bp} = \frac{2E_b}{V_{max}^2 - V_{min}^2} \tag{1.14}$$

La résistance interne de la batterie est représentée par les deux résistances R_{bs} et R_{b1} en série.
La résistance en bloc d'électrolyte et de plaque est représentée par la résistance R_{bs} tandis que la résistance R_{b1} représente la diffusion d'électrolyte.
Ceci représente la tension du circuit ouvert de la batterie dés qu'une charge sera reliée.
De même on peut observer un saut soudain de tension avec l'application d'un courant de remplissage En utilisant la notation indiquée su (figure.1.10) on peut exprimer la tension de batterie V_b en fonction de la tension de batterie en circuit ouvert et les autres composants R_{bs} et R_{b1} et C_{b1} avec le constante de temps $\tau = R_{b1}C_{b1}$

$$V_b = V_{oc} + R_{b1}(1 - e^{\frac{-t}{\tau}})i_b + R_{bs}i_b \tag{1.15}$$

Une autre caractéristique très importante d'une batterie est la décharge spontanée, représentée par la résistance R_{bp} parallèle avec condensateur principal C_{bs} .
Elle est provoquée par électrolyse de l'eau aux tensions élevées et par la fuite lente à travers les bornes de batterie aux basses tensions [38] La figure.1.10 peut être

mathématiquement exprimé dans le domaine de fréquence représentant l'impédance équivalente d'entrée d'une batterie plomb-acide .

$$\begin{cases} Z(S) = R_{bs} + (R_{b1} \parallel C_{b1}) + (R_{bp} \parallel C_{bp}) \\ = R_{bs} + \frac{R_{b1}}{R_{b1}C_{b1}S+1} + \frac{R_{bp}}{R_{bp}C_{bp}S+1} \end{cases} \tag{1.16}$$

L'utilisation dans le modèle mathématique du système transforme l'équation (1.16) en une limite simple de la forme suivante :

$$Z(S) = \frac{a_2S^2 + a_1S^1 + a_0}{b_2S^2 + b_1S^1 + b_0} \tag{1.17}$$

Les coefficients a_i et b_j sont employés pour représenter les différents composants :

$$\begin{cases} a_2 = R_{bs}R_{b1}R_{bp}C_{b1}C_{bp} \\ a_1 = R_{bs}R_{b1}C_{b1} + R_{bs}R_{bp}C_{bp} + R_{b1}R_{bp}C_{bp} + R_{bp}R_{b1}C_{b1} \\ a_0 = R_{bs} + R_{b1} + R_{bp} \\ b_2 = R_{b1}R_{bp}C_{b1}C_{bp} \\ b_1 = R_{b1}C_{b1} + R_{bp}R_{bp} \\ b_0 = 0 \end{cases} \tag{1.18}$$

C'est la forme finale du modèle mathématique de la batterie plomb-acide qui sera employée dans les simulations de système en chapitre V et en annexe.

1.12 Méthode de poursuite MPPT

1.12.1 Fonctionnement d'un générateur PV à sa puissance maximale

Aujourd'hui, compte tenu du prix élevé des générateurs PV et du faible rendement des dispositifs de conversion photons-électrons mis en oeuvre (entre 12 et 17 %), le développement de cette énergie à grande échelle nécessite avant tout une amélioration de ces systèmes de telle sorte qu'ils puissent fonctionner, à tout instant, à leur puissance maximale.
En effet, les études en simulation dans les paragraphes précédents ont bien montré que l'énergie des photons convertie en électricité est une fonction fortement variable selon l'éclairement et la température mais aussi selon la charge qui est connectée au générateur PV.
Pour remédier à cette dernière influence, des lois de commandes spécifiques ont été conçues et mises au point à partir de 1968 afin de permettre à ces dispositifs de produire leur maximum de puissance électrique, quelle que soit la charge [31]. Ce type de commande est souvent nommé dans la littérature Recherche du Point de Puissance Maximale ou bien Maximum Power Point Tracking en anglo-saxon (MPPT). Le principe de base, comme l'indique son nom, commun à toutes ces commandes est d'effectuer une recherche permanente du point de puissance maximale (PPM).
Ainsi, la principale fonction effectuée par ces commandes est d'assurer, à tout instant, une parfaite adaptation entre le générateur PV et sa charge fonctionnant au point de puissance max, le rôle d'interface de puissance étant assuré par un convertisseur statique.
Il existe plusieurs variantes, en fonction :

- du rendement électrique global de la chaîne de conversion souhaité par l'utilisateur,
- de la nature de la conversion de puissance permettant la connexion et l'adaptation à une charge donnée (DC-DC, DC-AC), du raccordement à un réseau électrique local ou non et de la quantité de puissance à fournir ainsi que de sa qualité,
- des conditions d'utilisation de l'énergie PV dans différentes applications qui peuvent être soit terrestres, soit spatiales,
- de la précision de la recherche du PPM, de sa rapidité, de sa robustesse vis-à-vis des différentes perturbations du système.

1.12.2 La réalisation

Principe

La conception globale de systèmes photovoltaïques optimisés est par nature difficile. En effet, côté source, pour un générateur photovoltaïque (PV), la production de puissance varie fortement en fonction de l'éclairement, de la température, mais aussi du vieillissement global du système.
Chaque charge, que ce soit en continu (DC) (batteries, certains appareils électroménagers destinés à des réseaux continus isolés) ou bien en alternatif (AC) (réseau électrique Algérien 220 V/50 Hz, certains moteurs), possède un comportement propre. Ainsi, pour qu'une connexion source-charge soit possible, un point de fonctionnement correspondant à l'intersection des caractéristiques électriques doit exister. Pour mieux comprendre ceci, prenons par exemple le cas d'une connexion directe entre un générateur PV et une charge purement résistive Ri d'une part et d'autre part entre un générateur PV et une batterie d'une résistance interne R_i. Nous regardons l'influence de la nature de la charge, comme illustré dans la (figure.1.11) et (figure.1.12).
Comme nous pouvons le constater sur la (figure.1.11.(b)) et la (figure.1.12.(b)), le

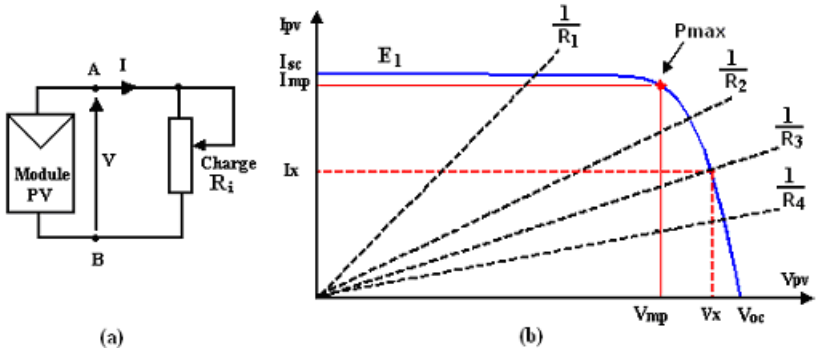

FIG. 1.11 – (a) Connexion électrique directe entre un générateur PV et une charge. (b) Points de fonctionnement résultant de l'association des générateurs PV sous un niveau d'éclairement E1 avec une charge résistive variable (R1, R2, R3, R4)

fonctionnement du générateur PV dépend fortement des caractéristiques de la charge à laquelle il est associé. En effet, pour la charge résistive ou pour l'accumulateur de résistance interne R_i sont données pour différentes valeurs, l'adaptation optimale ne se produit que pour un seul point de fonctionnement particulier, nommé Point de Puissance Maximal (PPM) et noté dans notre cas P_{max}. Ces formes correspondent à la puissance maximale que peut délivrer un générateur PV pour une courbe I(V)

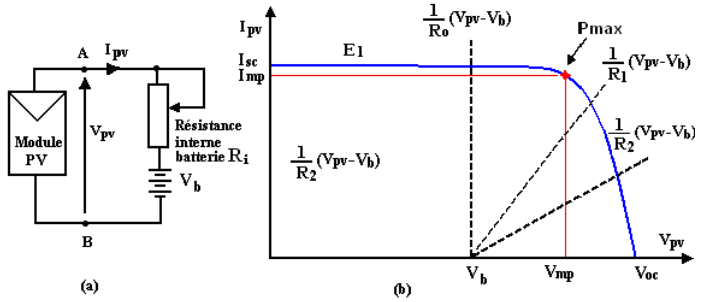

FIG. 1.12 – (a) Connexion électrique directe entre un générateur PV et une batterie. (b) Points de fonctionnement résultant de l'association des générateurs PV sous un niveau d'éclairement E1 avec une batterie comme charge ayant ou pas une résistance interne Ri Variable (R1, R2)

donnée. Pour la charge de type batterie, le point de connexion source-charge n'est pas optimal. Ainsi, lorsque l'on réalise une connexion directe source-charge(figure.1.13), le rendement de l'ensemble est rarement optimal.
Dans le cas d'une connexion directe entre une batterie et un générateur PV, le rendement du système dépend de l'écart entre la tension optimale du générateur PV (V_{op}) et la tension de batterie (V_b) qui varie en fonction de son état de charge.

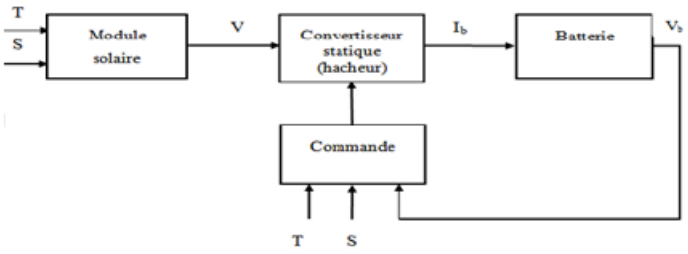

FIG. 1.13 – Structure du système de commande

Chapitre 2

Réseaux de neurones

2.1 Introduction

Avec l'apparition des ordinateurs l'homme a découvert un moyen d'effectuer diverses tâches avec deux capacités non négligeables que lui ne possède pas : la rapidité et la précision. Cependant l'exécution d'une tâche pour l'ordinateur nécessite sa programmation préalable par l'homme. Cette caractéristique fait apparaître les ordinateurs comme des machines exécutant des ordres (aveuglement) et l'homme n'a pas désespéré de voir un jour construire une machine à son image, c'est à dire intelligente, capable d'apprendre, de raisonner, de réfléchir sans son intervention.
Ce sont des recherches basées sur le fonctionnement du cerveau qui ont constitué le point de départ de cette gigantesque recherche.
Des travaux de neurobiologistes ont, en effet, révélé que le cerveau est constitué d'un nombre extrêmement élevé d'unités de traitement élémentaire de l'information (les neurones biologiques) fortement interconnectées.
L'information contenue dans le cerveau est stockée dans les connexions entre les neurones et c'est la coopération entre les neurones, qui effectuent un traitement fortement parallèle et distribué, qui donne sa puissance au cerveau.

2.2 Le modèle neurophysiologique

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone.
L'information traitée par le neurone chemine ensuite le long de l'axone pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms $10^{-9}m$ entre l'axone du neurone afférent et les dendrites du neurone efférent. La jonction entre deux neurones est appelée la synapse (figure.2.1)

2.3 Le modèle mathématique

Les réseaux de neurones biologiques réalisent facilement un certain nombre d'applications telles que la reconnaissance de formes, le traitement du signal, l'appren-

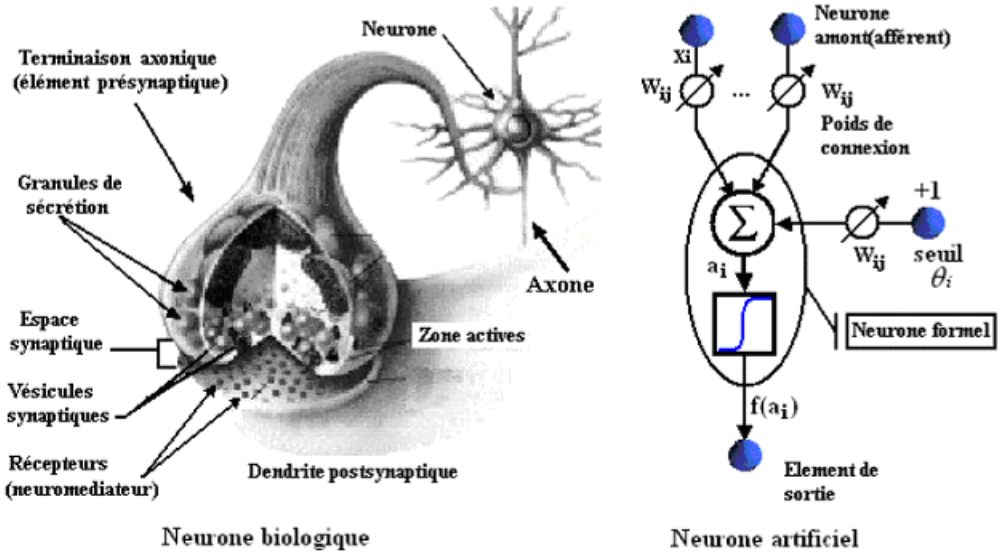

FIG. 2.1 – mise en correspondance neurone biologique/neurone artificiel

tissage par l'exemple, la mémorisation et la généralisation. C'est à partir de l'hypothèse que le comportement intelligent émerge de la structure et du comportement des éléments de base du cerveau que les réseaux de neurones artificiels se sont développés. Les réseaux de neurones artificiels sont des modèles. A ce titre, ils peuvent être décrits par leurs composants, leurs variables descriptives et les interactions des composants.

2.4 Structure

La figure.2.1 montre la similitude entre un neurone artificiel et un neurone biologique qui est l'objet d'inspiration de la structure artificielle. Comme il est illustré sur la structure artificielle, chaque neurone est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance des neurones amont (afférents). A chacune de ces entrées est associée un poids W abréviation de weight (poids en anglais) représentatif de la force (ou bien de la pondération) de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals (efférents).

2.5 Comportement

On distingue deux phases, la première est habituellement le calcul de la somme pondérée des entrées, où le neurone a_i reçoit des signaux de N neurones selon l'expression suivante :

$$a_i = \sum_{j=1}^{n} W_i j.X_j \tag{2.1}$$

A partir de cette valeur, la sortie du neurone sera évaluée par une fonction de transfert.
Lorsque les neurones possèdent une fonction de transition f , leur état de sortie des neurones est évalué par l'équation suivante :

$$X_i = f(a_i) \tag{2.2}$$

Il se trouve que pour résoudre des problèmes complexes, en utilisant les réseaux de neurones, il est très important d'introduire des non-linéarités au niveau du fonctionnement du réseau. Cette caractéristique du réseau ne peut être obtenue que par l'utilisation d'une fonction d'activation qui est à la fois continue, différentiable par rapport aux paramètres du réseau et bornée. On trouve plusieurs fonctions activation, au premier rang des fonctions les plus utilisées dans la phase d'apprentissage . On pourra alors utiliser une technique d'optimisation pour la minimisation d'une certaine fonction de coût, par exemple une descente du gradient.

2.5.1 fonction sigmoïde

La fonction d'activation sigmoïde (figure.2.2)est :

$$f(x) = \frac{1 - exp(-x)}{1 + exp(-x)} \tag{2.3}$$

et sa dérivée :

$$\frac{df(x)}{dx} = \frac{2 + exp(-x) + exp(x)}{(1 + exp(x))^2} \tag{2.4}$$

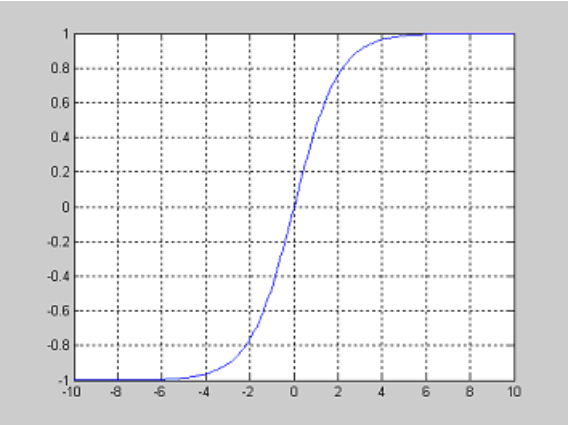

FIG. 2.2 – sigmoïde

2.5.2 fonction binaire sigmoïde

La fonction d'activation binaire sigmoïde (figure.2.3) est :

$$f(x) = \frac{1}{1 + exp(-x)} \tag{2.5}$$

et sa dérivée :

$$\frac{df(x)}{dx} = \frac{f(x)}{1 - f(x)} \tag{2.6}$$

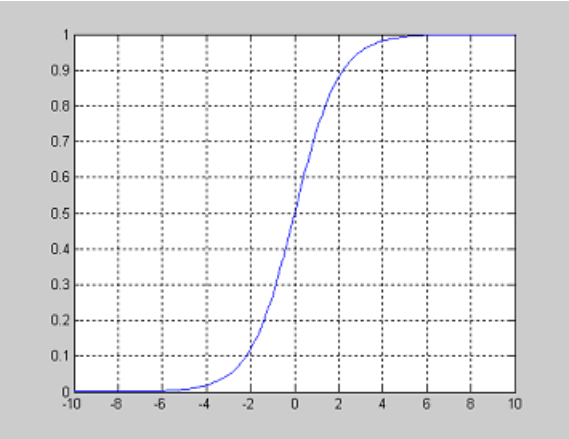

FIG. 2.3 – binaire sigmoïde

2.5.3 fonction bipolaire sigmoïde

La fonction d'activation bipolaire sigmoïde (figure.2.4)est :

$$f(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{2.7}$$

et sa dérivée :

$$\frac{df(x)}{dx} = \frac{4}{(exp(x) + exp(-x))^2} \tag{2.8}$$

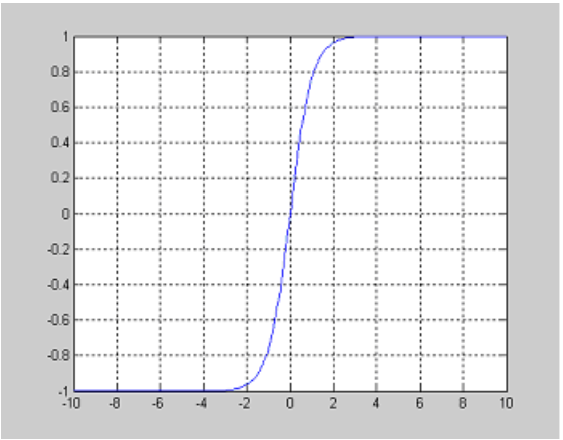

FIG. 2.4 – bipolaire sigmoïde

2.5.4 fonction arctangent

La fonction d'activation arctangent (figure.2.5) est :

$$f(x) = atan(x) \tag{2.9}$$

et sa dérivée :

$$\frac{df(x)}{dx} = \frac{2}{\pi(1 + (x)^2)} \tag{2.10}$$

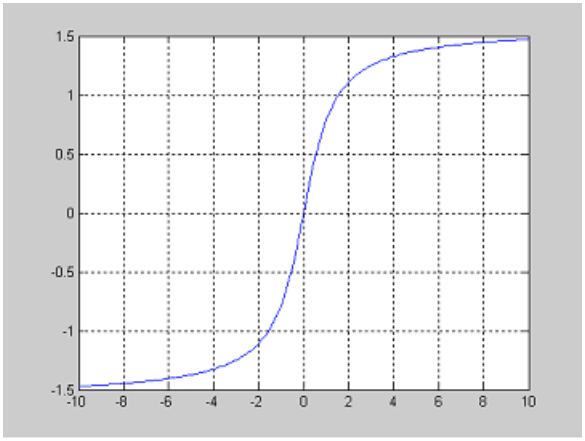

FIG. 2.5 – arctangent

2.5.5 fonction gaussien

La fonction d'activation gaussien (figure.2.6) est :

$$f(x) = exp(-x^2) \tag{2.11}$$

et sa dérivée :

$$\frac{df(x)}{dx} = -2x \exp(-x^2) \tag{2.12}$$

Nous appelons les réseaux multicouches qui utilisent ce type de fonctions Réseaux

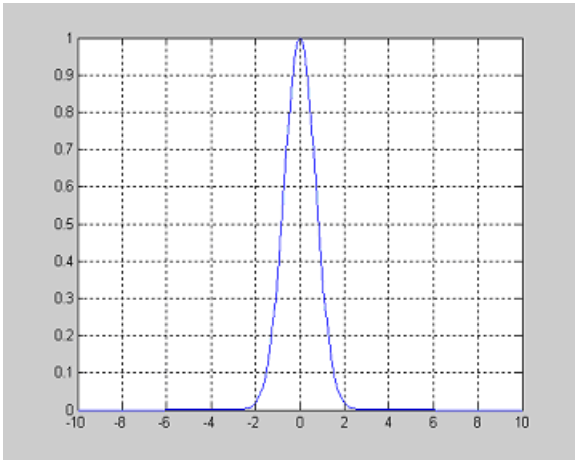

FIG. 2.6 – gaussien

Multicouche Quasi-linéaire (M.Q.L.). Dans notre travail, nous utilisons des réseaux M.Q.L, dont les fonctions de transition sont les fonctions sigmoïdes saturées et non saturées que nous venons de décrire.

2.6 Structure d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité . Dans le présent chapitre, nous abordons en détail la topologie des multicouches.

2.7 topologies neuronales

2.7.1 Réseau multicouche (MLP)

Dans les réseaux MLP (Multi Layer Perceptron), les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales (figure.2.7). Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et à celle-ci seulement. Nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau.
On appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées couches cachées.

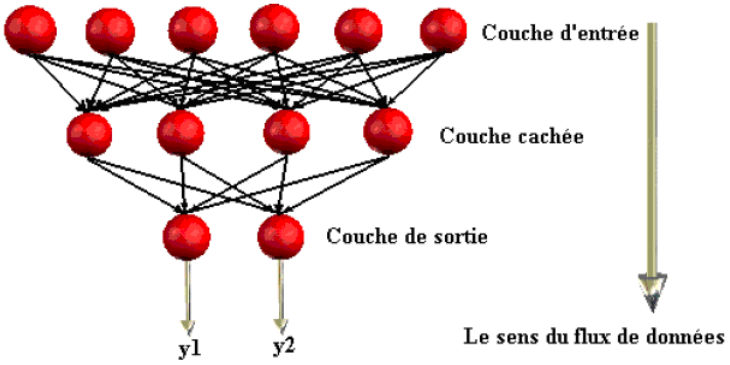

FIG. 2.7 – Topologie d'un réseau multicouche (MLP)

Equation du réseau

Un réseau de neurones MLP à N couches, de P entrées et Q sorties réalise une application de l'entrée vers la sortie, où chaque entrée d'un neurone du réseau se présente comme une fonction linéaire, le tout contrôlé par une fonction non linéaire $f(a_i)$ telle que :

$$\begin{cases} \Re^P \rightarrow \Re^Q \\ f(a_i) \rightarrow Y_i \end{cases} \tag{2.13}$$

où la combinaison linéaire des entrées pondérées s'appelle le potentiel du neurone i.

2.8 Apprentissage d'un réseau multicouche

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. La plupart d'entres eux font appel à des règles d'apprentissage sur des données pour ajuster les poids des connexions synaptiques. Durant cette phase d'apprentissage, l'état du réseau de neurones évolue suivant une loi de minimisation du coût de sa sortie, jusqu'à l'obtention du comportement désiré .
En conséquence, le but de l'apprentissage est l'estimation des poids synaptiques, pour remplir au mieux la tâche à laquelle le réseau est destiné .

2.8.1 Apprentissage supervisé

Un apprentissage supervisé est lié à la disponibilité d'exemples, ou de modèles de réponses. L'ensemble des exemples utilisés pour l'apprentissage se présente sous forme de couples (x, y*) où y* est la réponse désirée du réseau à l'entrée x.
Lorsque le réseau apprend à mimer une fonction multivariable, il utilise des exemples d'entrée/sortie. Ses poids sont ajustés sous l'influence d'un signal d'erreur qui représente la différence entre la sortie estimée par le réseau et la sortie désirée par le modèle.
Les caractéristiques de l'algorithme neuronal lui permettent ensuite de généraliser et de répondre à une entrée quelconque, même si elle n'appartient pas au jeu d'apprentissage.

2.8.2 Apprentissage non supervisé

L'hypothèse d'existence d'un maître qui supervise l'apprentissage n'est pas toujours possible. Dans ce cas, on ne connaît pas les sorties désirées y* des exemples d'apprentissage. Le réseau doit donc apprendre de lui-même et on parle d'apprentissage non supervisé.

2.9 Caractéristique de l'algorithme d'apprentissage supervisé

Un réseau de neurones est conçu pour réaliser une tâche que le concepteur définit par un ensemble d'apprentissage D (base de données). Chaque élément de cet ensemble est appelé exemple d'apprentissage et se présente sous la forme d'un couple (x, y*) où x est une valeur d'entrée du réseau et y* la valeur désirée correspondante pour les sorties des neurones de sortie.
L'architecture du réseau, la structure de ses connexions, ainsi que les fonctions d'activation, peuvent être fixées en fonction de la tâche que doit remplir le réseau.
Les valeurs des poids synaptiques sont, en général, déterminées par un processus algorithmique mettant en oeuvre l'ensemble d'apprentissage.
Le but de l'apprentissage est donc de déterminer les valeurs W* de la matrice W des poids des connexions du réseau de telle sorte que les sorties (y) soient proches des valeurs désirées y*. W* est donc la solution d'un problème d'optimisation, consistant à minimiser une fonction de coût : E(W, D).

2.9.1 La méthode du gradient

La plupart des méthodes d'optimisation non linéaires sont basées sur la même stratégie. On choisit une valeur initiale W(0) de la matrice W, puis on utilise un processus itératif dans lequel on tente d'optimiser la fonction E.chaque itération, cette optimisation implique deux étapes :

- le choix de la direction dans laquelle on va chercher la valeur suivante W(t+1).
- et le déplacement le long de cette direction.

2.9.2 Rétropropagation du gradient

L'algorithme de rétropropagation du gradient est très connu et le plus utilisé dans les applications des réseaux de neurones.
A chaque itération, on retire un exemple d'apprentissage (x, y*) et on calcule une nouvelle estimation de W(t).
Cette itération est réalisée en deux phases

2.9.3 Propagation

A chaque itération, un élément de l'ensemble d'apprentissage D est introduit à travers la couche d'entrée. L'évaluation des sorties du réseau se fait couche par couche, de l'entrée du réseau vers sa sortie.

2.9.4 Rétropropagation

Cette étape est similaire à la précédente. Cependant, les calculs s'effectuent dans le sens inverse.
A la sortie du réseau, on de le critère de performance J en fonction de la sortie réelle du système et sa valeur désirée. Puis, on évalue le gradient de J par rapport aux différents poids en commençant par la couche de sortie et en remontant vers la couche d'entrée.

2.9.5 Calcul du gradient

Pour un exemple i d'un ensemble d'observation E, la fonction de coût des moindres carrés est égale à la somme, sur les N_2 valeurs de l'ensemble d'observation, des carrés des écarts entre la sortie du modèle (sortie du réseau de neurones = y_i) et la sortie désirée (grandeur mesurée notée). On cherche à minimiser, à chaque étape de mise à jour, le critère suivant

$$J = \frac{1}{2}\sum_{i=1}^{N_2}(y_i^{des} - y_i) \tag{2.14}$$

y_i^{des} : la composante i de la sortie désirée du système,
y_i : la composante i de la sortie calculée du système,
N_2 : le nombre d'exemples (des valeurs) dans la base d'apprentissage.
Le problème consiste à déterminer les poids W de toutes les couches qui minimisent le critère de performance J.
La mise à jour de W se fait selon la règle de delta :

$$\Delta W = -\eta\frac{\partial W}{\partial J} \tag{2.15}$$

$$W_i j^k(t+1) = W_i j^k(t) - \Delta W \tag{2.16}$$

ce qui revient à déterminer les variations du critère de performance J par rapport aux variations des poids.

2.10 Modification des paramètres du réseau en fonction du gradient de J

Dans l'étude précédente, nous avons vu le gradient ,à chaque itération du processus d'apprentissage. Une fois que l'on dispose de cette évaluation, on effectue une modification des poids selon l'équation , afin d'approcher d'un minimum de la fonction de coût J dans l'espace des poids. Pour cela il faut que la condition suivante doit être vérifiée :

$$\frac{\partial J}{\partial W_i j} = 0 \tag{2.17}$$

Cette méthode du gradient de premier ordre est simple, mais elle présente de nom-

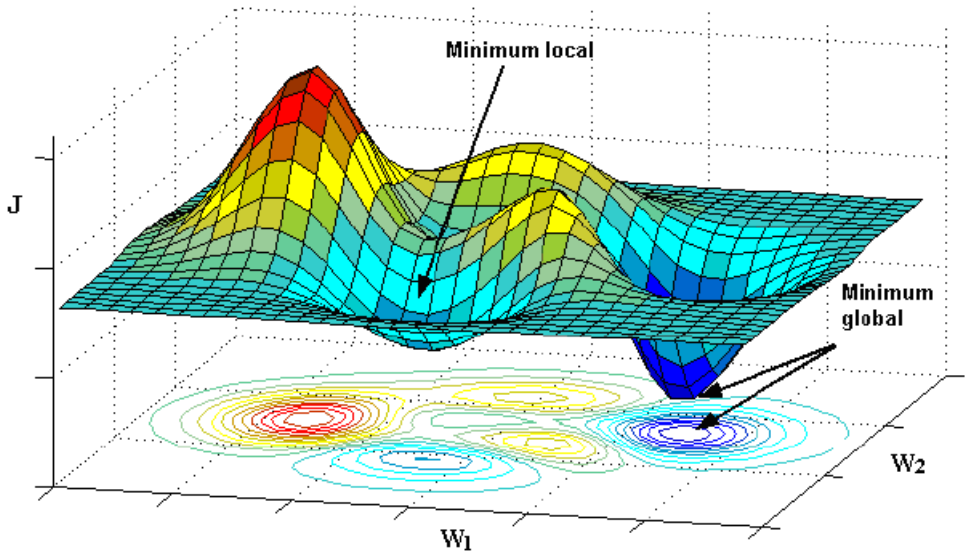

FIG. 2.8 – Représentation de la fonction de coût J d'un neurone à deux entrées pondérées W1 et W2

breux inconvénients.En voici quelques uns :

- Si le pas du gradient est trop petit, la décroissance du coût J est très lente. Si le pas est trop grand, le coût J peut augmenter ou osciller, ce qui représente une forme 3D d'une fonction de coût J ayant des minimum locaux et un minimum global, où sa projection sur le plan de base formé par deux variables (poids W1 et W2) donne un contour de plusieurs niveaux de la fonction de coût(figure.2.8).
- Au voisinage d'un minimum de la fonction de coût, la pente de la descente est faible, ce qui revient à dire que le gradient de cette fonction tend vers zéros. A ce niveau, multiplier cette grandeur par le pas d'apprentissage ?, n'améliorera pas considérablement le résultat, et donc l'évolution du vecteur des paramètres du réseau lors de la mise à jour par l'équation(1.15) devient très lente. Il en va de même si la fonction du coût présente des (plateaux) où sa pente est très faible. Ces plateaux peuvent être très éloignés d'un minimum, et, il est impossible de savoir si une évolution très lente du gradient est due au fait que l'on est au voisinage d'un minimum, ou que l'on se trouve sur un plateau de la fonction de coût.

– Si la courbure de la surface de coût varie beaucoup, la direction du gradient peut être très différente de la direction qui mènerait vers le minimum .

2.11 Choix d'une structure neuronale

Un problème qu'on doit impérativement résoudre avant d'utiliser un réseau de neurones est la définition de sa structure. Pour une topologie multicouche MLP, le nombre de neurones d'entrée/sortie du réseau de neurones est imposé par la structure de fonctionnement globale où il sera inséré, tandis que le nombre de couches cachées ainsi que le nombres de neurones correspondant à chaque couche, ne sont pas limités. L'objectif final étant une réalisation matérielle embarquable, pour diminuer le temps de calcul, il est nécessaire de développer une architecture neuronale aussi petite que possible.

2.12 Initialisation du vecteur du paramètre de poids W

Au lancement de l'apprentissage, les valeurs initiales des poids doivent êtres différentes de zéro pour que l'algorithme de rétropropagation puisse fonctionner. D'autre part, l'utilisation de valeurs élevées peut provoquer un phénomène de saturation prématurée qui contribue à diminuer la vitesse de convergence de l'apprentissage. Ce phénomène est fonction de la valeur des poids et/ou de la pente de la sigmoïde et du nombre de neurones dans chaque couche. Afin de se situer dans la zone linéaire de la sigmoïde (zone définie pour une entrée proche de zéro). Généralement on tire les poids du réseau de neurone entre deux bornes [B1, B2] [1]. En effet l'apprentissage n'est effectif que lorsque l'activité des neurones se trouve dans la partie linéaire de la fonction sigmoïde (car sa dérivée est nulle lorsque le neurone est saturé).

2.13 Application et simulation

2.13.1 Procédure de commande par rétropropagation du gradient

La performance de tout contrôleur MPPT est qualifiée, d'une part par sa rapidité à réagir par une commande adéquate (rapport cyclique), dépourvue de toute oscillation faisant ramener le point de fonctionnement du système PV à la position du PPM; et d'autre part, par sa capacité de synthétiser la variable de commande directement à partir de certaines variables d'états, qui sont respectivement S : ensoleillement, T : température et V_b : tension de batterie et qui ont un effet direct sur la dynamique du PPM.
Les différentes techniques MPPT classiques montrent que leurs algorithmes de recherche sont basés sur la disponibilité des données d'ensoleillement et de température. La mesure de ces données est effectuée à travers d'autres paramètres liés au système PV tels que le courant, la tension et la puissance. Il se trouve que ces

[1] pour l'initialisation nughen et window -B1=B2=1 [4]

paramètres ne dépendent pas que des variables climatiques, mais aussi, des caractéristiques internes du module PV qui imposent automatiquement des contraintes sur le dimensionnement du contrôleur classique.
De ce fait, nous allons essayer dans ce qui suit d'élaborer un contrôleur neuronal, de toute contrainte imposée par le système PV, ce qui nécessitera de ne mettre en jeu que les variables climatiques. Et cela dans le but d'améliorer la qualité de la commande, ce qui améliorera sans doute le rendement en puissance.
En conséquence du raisonnement ci-dessus, nous avons structuré en schéma synoptique (figure.2.9) le système de fonctionnement global en présence du contrôleur neuronal.
Faire entraîner le réseau de neurones afin qu'il puisse fonctionner au mieux dans l'architecture de commande de la (figure.2.9) est une étape incontournable.
Pour une valeur bien déterminée en ensoleillement (S) et en température (T), le

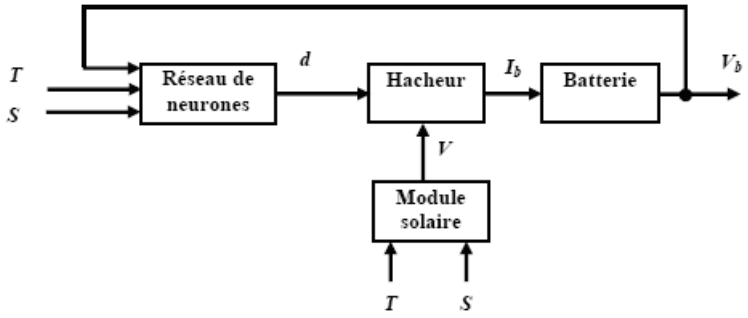

FIG. 2.9 – Structure du système de commande par réseau de neurones

module solaire possède un point de puissance maximale unique PPM ; pour se situer en ce point, il apparaît très évident d'appliquer un rapport cyclique (d^{des}) bien déterminé à l'entrée commande du hacheur.
L'étude effectuée précédemment sur le système photovoltaïque nous fait déduire que le rapport cyclique (d^{des}) ne dépend pas seulement des conditions climatiques (S et T), mais dépend aussi des caractéristiques internes du module solaire, du convertisseur et de la charge à laquelle il est connecté (V_b).
En raison, de la difficulté à modéliser le rapport cyclique en tenant compte de tous les paramètres possibles (propres au module et climatiques), nous avons opté pour un moyen plus simple et judicieux pour établir une base de données, riche en termes d'informations sur le système PV et sur laquelle le contrôleur neuronal sera entraîné.
Son principe, est d'effectuer des mesures réelles à partir d'une installation PV, équipée d'un contrôleur MPPT classique doté d'un appareillage de mesure du rapport cyclique (d^{des}) correspondant à (V_b, T, S). Il est beaucoup plus aisé, si le travail s'effectue sur un simulateur de panneaux équipé d'un matériel adéquat pour faire les tests et les mesures possibles et d'établir ainsi cette base de données.
Dans notre cas, suite à l'équipement très restreint mis à notre disposition, nous nous sommes servi de utilise la base de données déjà utilisée dans [31] (synthétisée à partir de plusieurs simulations effectuées sur un système PV commandé par un contrôleur MPPT classique), et pour le montrer l'applicabilité de la commande neuronale.
Après avoir dressé la base de données composée de (T, V_b,S,d^{des}) l'étape de l'apprentissage sera effectuée sur la structure illustrée à la figure.2.10 , en raison de la simplicité d'entraînement du réseau de neurones hors de la structure de commande de la figure.2.9.

Notons que tout contrôleur entraîné sur cette base de données, ne peut être utile

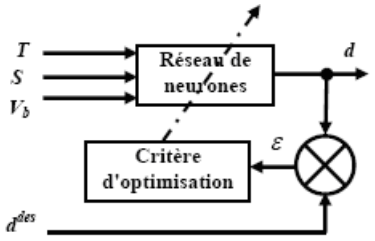

FIG. 2.10 – Structure d'apprentissage du contrôleur MPPT neuronal

dans le contrôle MPPT, que pour les modules PV ayant les mêmes caractéristiques c-à-d issus d'un même fabriquant, et pour la même disposition des modules formant le panneau PV.

2.13.2 Programmes d'apprentissage (sur MATLAB)

Après choix de la structure du système de commande par réseau de neurones (figure.2.9),utilisation de la base de données et établissement la structure d'apprentissage du contrôleur MPPT neuronal (figure.2.10) nous sommes passés à la programmation du système d'apprentissage sur le MATLAB dont le mécanisme du fonctionnement est décrit par l'organigramme présenté sur la (figure.2.11).
Ce dernier montre que, sur chaque cycle de l'algorithme principal de commande, une mesure des variables d'entrées du réseau, respectivement (T, S et V_b), est effectuée, puis une propagation directe de ces données via le réseau de neurones. Cette dernière estime la commande adéquate à sa sortie qui sera appliquée à l'entrée commande du hacheur.
Ce dernier montre que sur chaque cycle de l'algorithme principal de commande, une mesure des variables d'entrées du réseau qui sont respectivement (T, S et V_b) est effectuée, puis une propagation directe de ces données via le réseau de neurones. Cette dernière estime la commande adéquate à sa sortie qui sera appliquée à l'entrée commande du hacheur.
L'apprentissage utilisé dans la structure est basé sur l'algorithme de rétropropagation du gradient (Feed Error Propagation) appelé communément F.E.P exposé précédemment. C'est un apprentissage supervisé basé sur des données connues composée d'une multitude de signaux en entrée/sortie qui sont respectivement température (T), ensoleillement (S), tension dynamique de la batterie (V_b) et le rapport cyclique correspondant (d^{des}) (figure.2.12).

2.13.3 Discussion des résultats de programmation

A partir des résultats de programmation de l'algorithme d'apprentissage (la sortie du système qui est le rapport cyclique d dans la figure.2.13 qui montre l'évolution de la phase d'apprentissage), on remarque que :

- notre système converge vers la solution désirée puisque le rapport cyclique calculé d converge vers le rapport cyclique ' d^{des}' et qu'ils sont presque superposés (sur la figure le rouge est ' $d^{calculer}$' et le bleu c'est d^{des}), qui montre que le

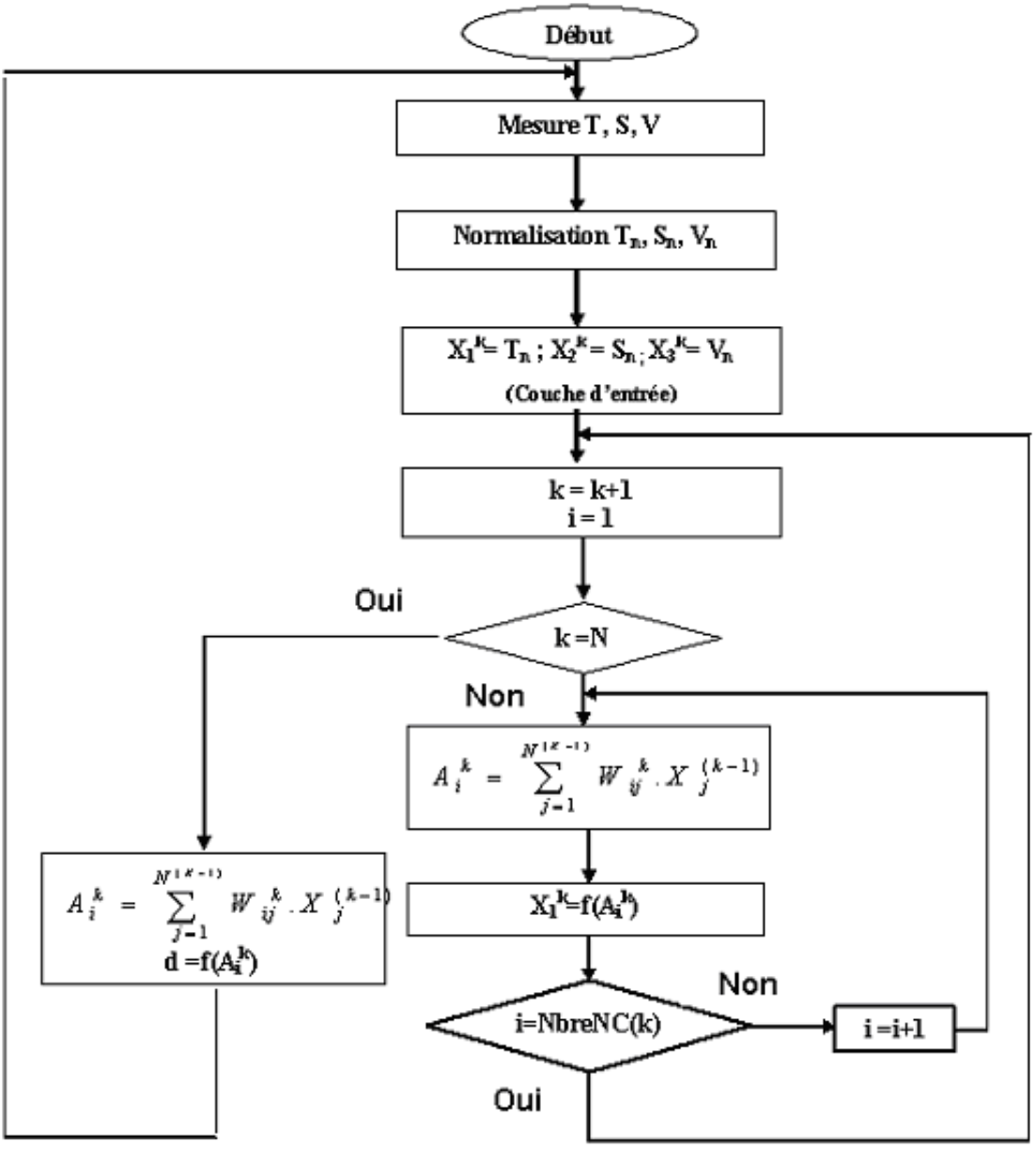

FIG. 2.11 – Organigramme du principe du fonctionnement de la commande neuronale

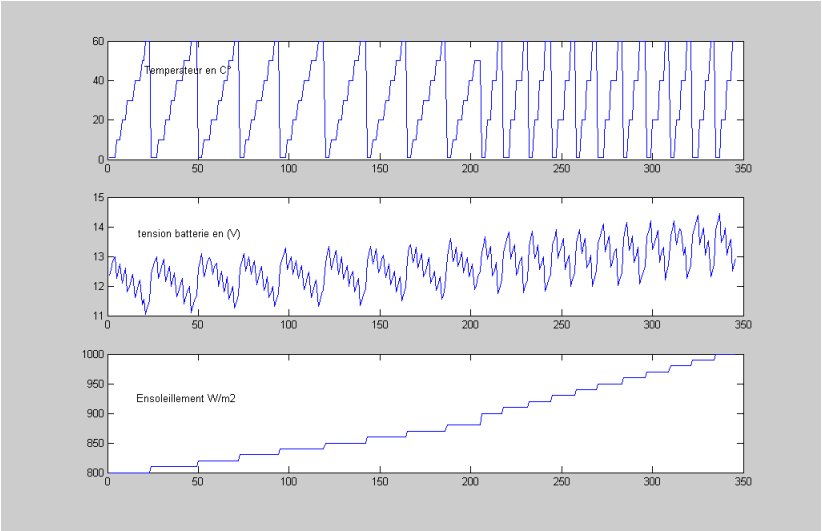

FIG. 2.12 – température, ensoleillement et tension de batterie

système est bien apprendre, et qui veut dire les paramètres du système bien déterminés.

– de l'algorithme dés le début des l'apprentissage (dans les premières itérations) des premières valeurs de la base de donnée,l'algorithme convergence rapidement .

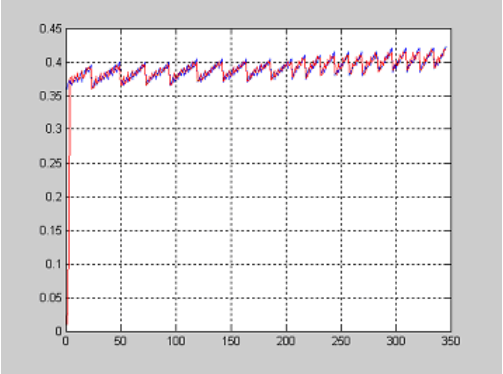

FIG. 2.13 – l'évolution de la phase d'apprentissage

2.13.4 Choix de la structure des réseaux de neurones pour les programmes finaux

Un problème que l'on doit impérativement résoudre avant d'utiliser un réseau de neurones est la définition de la structure du réseau. Le nombre d'entrées et de sorties est généralement imposé par la fonction à approximer (dans le cadre de notre travail, trois neurones pour la couche d'entrée et un en sortie). Le nombre de couches cachées ainsi que leur nombre de neurones correspondant exige une structure optimale.

Pour cela nous avons conçu plusieurs test avec différentes structures (variation du nombre de couches et du nombre de nuerons par couches) et différents paramètres (le biais et les fonctions d'activation) qui nous permet de choisir la structure optimale

de réseaux de neurones.
Nous avons choisi comme structure finale qui sera utilisé pour les programmes finaux dans suite de notre travail (pour la simulation de SIMULINK et, dans l'implémentation sur DSP et FPGA) une structure à trois couches : une couche d'entrée, une couche de sortie et une couche cachée ayant 1 seul neurone, sans biais et avec une fonction d'activation sigmoïde.
Le choix de la structure et le nombre de couches sont justifié dans l'annexe A.

2.14 Conclusion

Dans le présent chapitre nous avons abordé les bases des réseaux de neurones et les commandes connexionnistes. Après avoir donné un aperçu sur la similitude neurone biologique/neurone artificiel, nous nous sommes intéressés plus particulièrement à l'étude des architectures de type perceptron multicouches entraînées avec l'algorithme de la rétropropagation du gradient.
Les performances de la commande neuronale ,qui est l'objet de notre travail sont exposées en détail au chapitre 5.

Chapitre 3

Développement d'un projet sur FPGA

3.1 Introduction

Les progrès technologiques continus dans le domaine des circuits intégrés ont permis la réduction des coûts et de la consommation. Les circuits intégrés spécifiques ont permis une réduction de la taille des systèmes numériques ainsi que la réalisation de circuits de plus en plus complexes, tout en améliorant leurs performances et leur fiabilité. Aujourd'hui les techniques de traitement numérique occupent une place majeure dans tous les systèmes électroniques modernes grand public, professionnel ou militaire . De plus, les techniques de réalisation de circuits spécifiques, tant dans les aspects matériels (composants reprogrammables, circuits précaractérisés et bibliothèques de macrofonctions) que dans les aspects logiciels (placement-routage, synthèse logique) font désormais de la microélectronique une des bases indispensables pour la réalisation de systèmes numériques performants. Elle impose néanmoins une méthodologie de développement très structurée.
Les circuits FPGA (Field Programmable Gate Array) sont certainement les circuits reprogrammables ayant le plus de succès. Ce sont des circuits entièrement configurables par programmation qui permettent d'implanter physiquement, par simple programmation, n'importe quelle fonction logique. De plus, ils ne sont pas limités à un mode de traitement séquentiel de l'information comme avec les microprocesseurs ; et en cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement. L'objet de notre étude a été d'implémenter sur un circuit FPGA la méthode MPPT neuronale déjà citée dans le chapitre précéde . La programmation a été faite en utilisant le langage de description VHDL. Nous allons d'abord faire une description des circuits FPGA, ensuite nous introduire le langage VHDL, et nous terminerons par donner les étapes nécessaires au développement d'un projet sur un circuit FPGA, de la programmation jusqu'au chargement sur la carte.

3.2 Circuits FPGA

Les FPGA (Field Programmable Gate Array) sont des circuits à architecture programmable qui ont été inventés par la société XILINX en 1985. Ils sont entièrement reconfigurables et ne demandent donc pas de fabrication spéciale en usine, ni

de systèmes de développement coûteux ; ceci permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. Un autre avantage de ces circuits est leur grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court (quelques millisecondes).
De nombreuses familles de circuits programmables et reprogrammables sont apparues depuis les années 70 avec des noms très divers suivant les constructeurs. La (figure.3.1) donne une classification possible des circuits numériques en précisant où se situent les circuits FPGA dans cette classification. Les FPGA sont utilisés dans

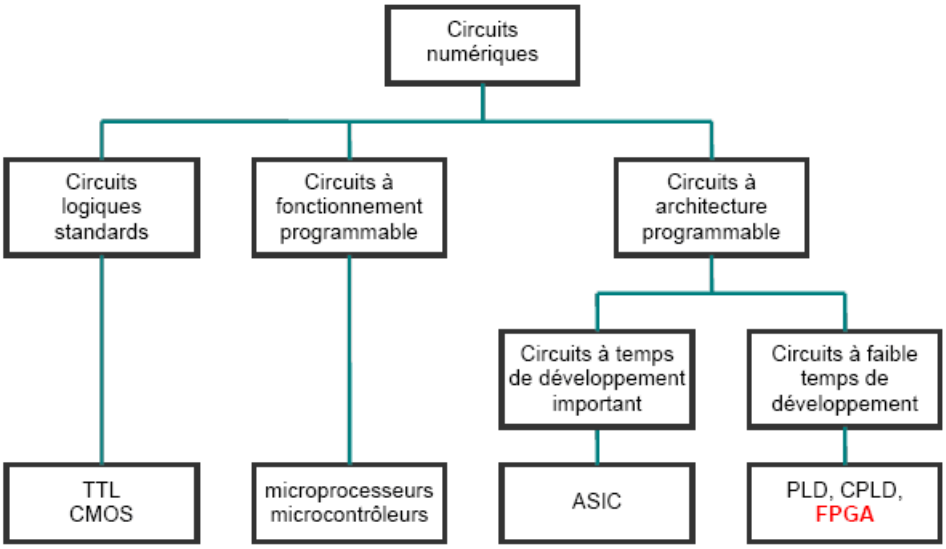

FIG. 3.1 – Classification des circuits numériques

de nombreuses applications, on en cite dans ce qui suit quelques unes :

- prototypage de nouveaux circuits ,
- fabrication de composants spéciaux en petite série ,
- adaptation aux besoins rencontrés lors de l'utilisation ,
- systèmes de commande à temps réel ,
- DSP (Digital Signal Processor) ,
- imagerie médicale.

3.2.1 Architecture des circuits FPGA

Les circuits FPGA possèdent une structure matricielle de deux types de blocs (ou cellules). Des blocs d'entrées/sorties et des blocs logiques programmables. Le passage d'un bloc logique à un autre se fait par un routage programmable. Certains circuits FPGA intègrent également des mémoires RAM, des multiplieurs et même des noyaux de processeurs. Actuellement deux fabricants mondiaux se disputent le marché mondial des FPGA : Xilinx et Altera. De nombreux autres fabricants, de moindre envergure, proposent également leurs propres produits avec des technologies et des principes organisationnels différents. Dans ce qui suit, on va faire une description de l'architecture utilisée par Xilinx, car c'est sur des circuits Xilinx que nous allons va implémenter notre programme. L'architecture retenue par Xilinx se présente sous forme de deux couches :

- une couche appelée circuit configurable.

– une couche réseau mémoire SRAM (Static Read Only Memory).

Circuit configurable

La couche dite (circuit configurable) est constituée d'une matrice de blocs logiques configurables (CLB) permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, nous trouvons des blocs d'entrées/sorties (IOB) dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs (figure.3.2). La programmation du circuit FPGA, appelé aussi LCA (Logic Cells Arrays), consistera en l'application d'un potentiel adéquat sur la grille de certains transistors à effet de champ servant à interconnecter les éléments des CLB et des IOB, afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux. Ces potentiels sont tout simplement mémorisés dans le réseau mémoire SRAM.

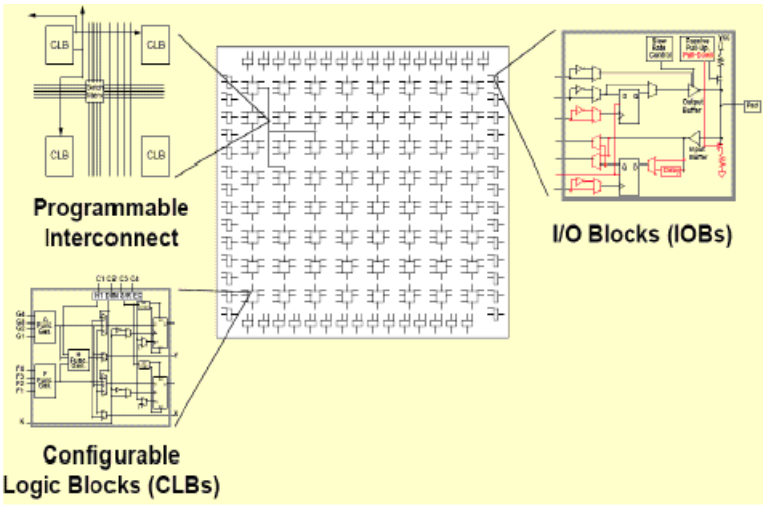

FIG. 3.2 – Architecture interne du FPGA

Réseau mémoire SRAM

La programmation d'un circuit FPGA est volatile, la configuration du circuit est donc mémorisée sur la couche réseau SRAM et stockée dans une ROM externe. Un dispositif interne permet à chaque mise sous tension de charger la SRAM interne (figure.3.3) à partir de la ROM. Ainsi on conçoit aisément qu'un même circuit puisse être exploité successivement avec des ROM différentes puisque sa programmation interne n'est jamais définitive. On voit tout le parti que l'on peut tirer de cette souplesse en particulier lors d'une phase de mise au point. Une erreur n'est pas rédhibitoire, mais peut aisément être réparée. Les blocs logiques configurables sont les éléments déterminants des performances du circuit FPGA. Chaque CLB est un bloc de logique combinatoire composé de générateurs de fonctions à quatre entrées (LUT) et d'un bloc de mémorisation/synchronisation composé de bascules D. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB. La LUT (Look Up Table) est un élément qui dispose de quatre entrées, il existe donc $2^4 = 16$ combinaisons différentes de ces entrées. L'idée consiste à mémoriser la sortie correspondant à chaque combinaison d'entrée dans

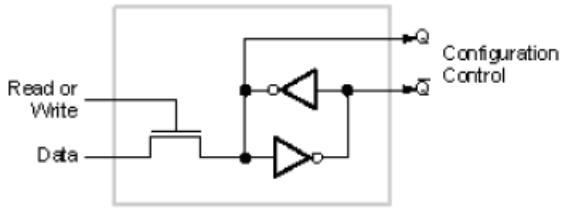

FIG. 3.3 – Structure d'un cellule SRAM

une petite table de 16 bits, la LUT devient ainsi un petit bloc générateur de fonctions. La (figure.3.4) ci-dessous montre le schéma simplifié d'un CLB de la famille XC4000 de Xilinx.

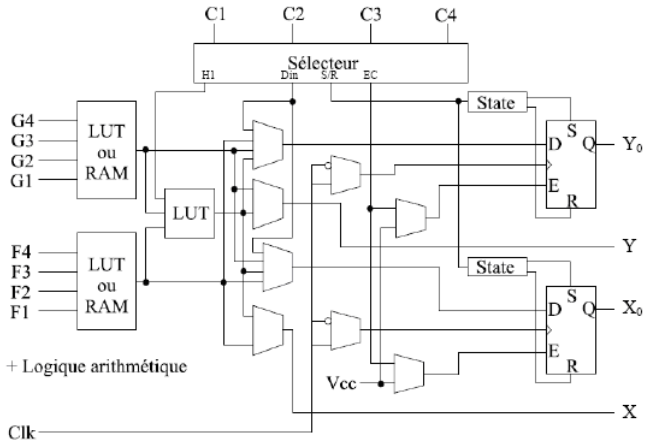

FIG. 3.4 – Schéma d'une cellule logique (XC4000 de Xilinx)

3.2.2 Les IOB (Input Output Bloc)

Ces blocs d'entrée/sortie permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance). La (figure.3.5) présente la structure de ces blocs.

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.

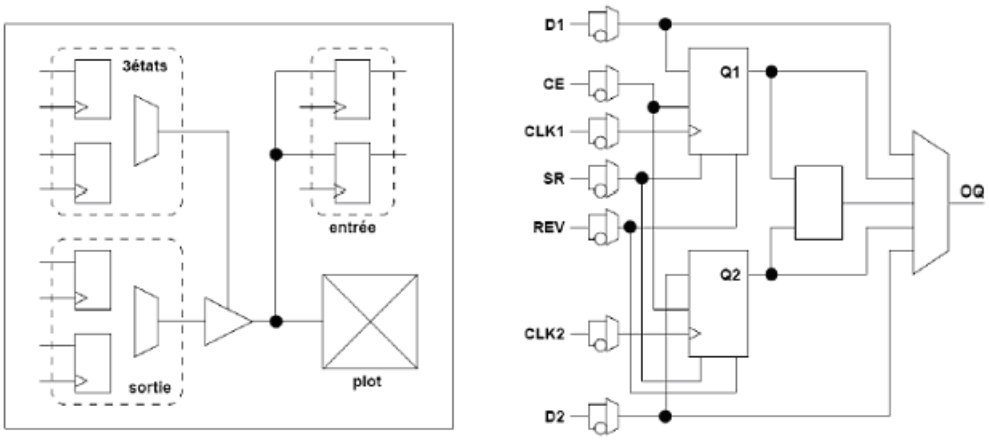

FIG. 3.5 – Schéma d'un bloc d'entrée/sortie (IOB)

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 ,
- signaux en logique trois états ou deux états, le contrôle de mise en haute impédance et la réalisation des lignes bidirectionnelles sont commandés par le signal de commande (Out Enable), lequel peut être inversé ou non.

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.

3.2.3 Les différents types d'interconnexion

Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM (Random Access Memory). Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les blocs d'entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Pour parvenir à cet objectif, Xilinx propose trois sortes d'interconnexion selon la longueur et la destination des liaisons. Nous disposons :
- d'interconnexions à usage général,
- d'interconnexions directes ,
- de longues lignes.

Les interconnexions à usage général

Des aiguilleurs appelés aussi (switch matrix) sont situés à chaque intersection. Leur rôle est de raccorder les segments entre eux selon diverses configurations, ils assurent ainsi la communication des signaux d'une voie vers l'autre . Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre CLB. Pour éviter

que les signaux traversant les grandes lignes ne soient affaiblis, nous trouvons généralement des buffers implantés en haut et à droite de chaque matrice de commutation (figure.3.6).

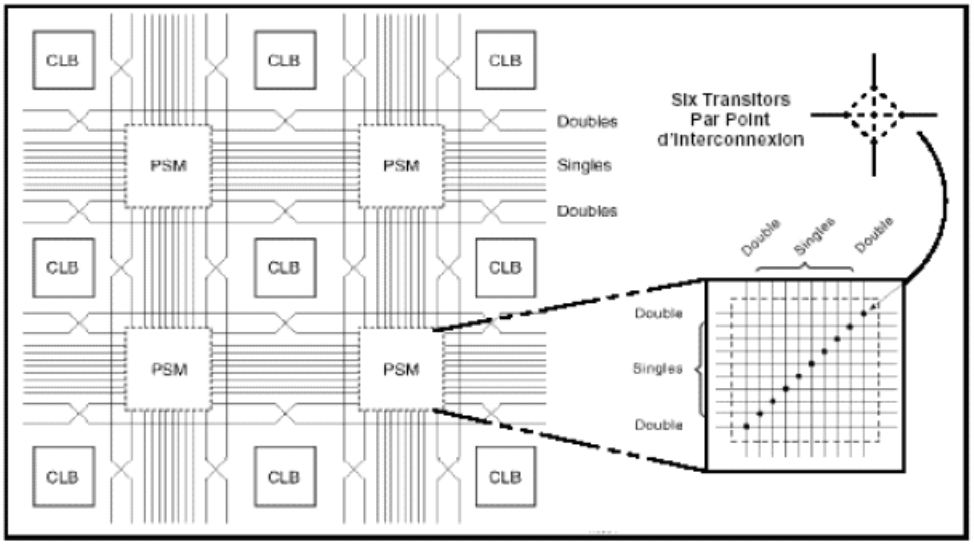

FIG. 3.6 – Connexions à usage général et détail d'une matrice de commutation

Les interconnexions directes

Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en termes de vitesse et d'occupation du circuit. De plus, il est possible de connecter directement certaines entrées d'un CLB aux sorties d'un autre (figure.3.7).

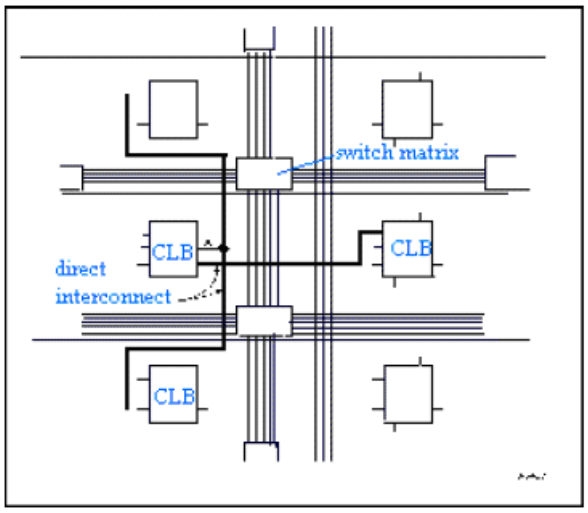

FIG. 3.7 – Les interconnexions directes

Les longues lignes

Les longues lignes sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un

minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion (figure.3.8).

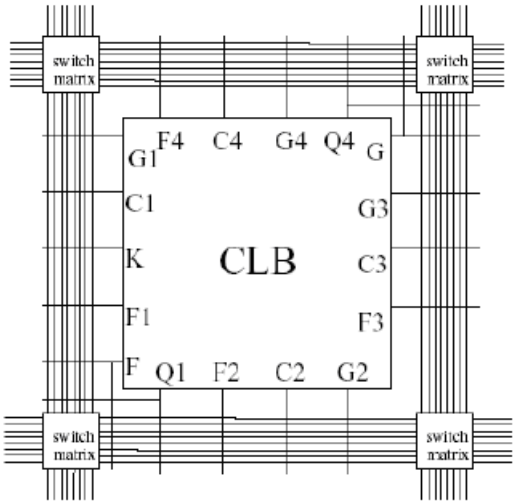

FIG. 3.8 – Les longues lignes

3.3 Kit de développement Virtex-II V2MB1000 de Memec

Le kit de développement Virtex-II V2MB1000 de Memec Design, qu'on a utilisé pour développer notre application, fournit une solution complète de développement d'applications sur la famille Virtex-II de Xilinx. Il utilise le circuit (FPGA XC2V1000-4FG456C)qui appartient à la famille Virtex-II de Xilinx et qui est équivalent à 1 million de portes logiques. La haute densité d'intégration des portes ainsi que le nombre important d'entrées/sorties disponibles à l'utilisateur permettent d'implémenter des systèmes complets de solutions sur la plate forme FPGA. La carte de développement inclut aussi une mémoire 16M x 16 DDR, deux horloges, un port série RS-232 et des circuits de support additionnels. Une interface LVDS est disponible avec un port de transmission 16-bit et un port de réception 16-bit, en plus de signaux d'horloge, d'état et de contrôle pour chacun de ces ports. La carte supporte également le module d'expansion Memec Design P160, qui permet d'ajouter facilement des modules pour des applications spécifiques. La famille FPGA Virtex-II possède les outils avancés pour répondre à la demande applications de haute performance. Le kit de développement Virtex-II fournit une excellente plateforme pour explorer ces outils. L'utilisateur peut alors utiliser toutes les ressources disponibles avec rapidité et efficacité. La figure.3.9 présente une photo de la carte de développement et de ses outils.

3.3.1 Description de la carte de développement

Un diagramme simplifié de la carte de développement Virtex-II est illustré à la figure.3.10. La carte de développement Virtex-II contient le circuit FPGA XC2V1000-4FG456C. Ce circuit fait partie de la famille Virtex-II, qui est une famille de circuits

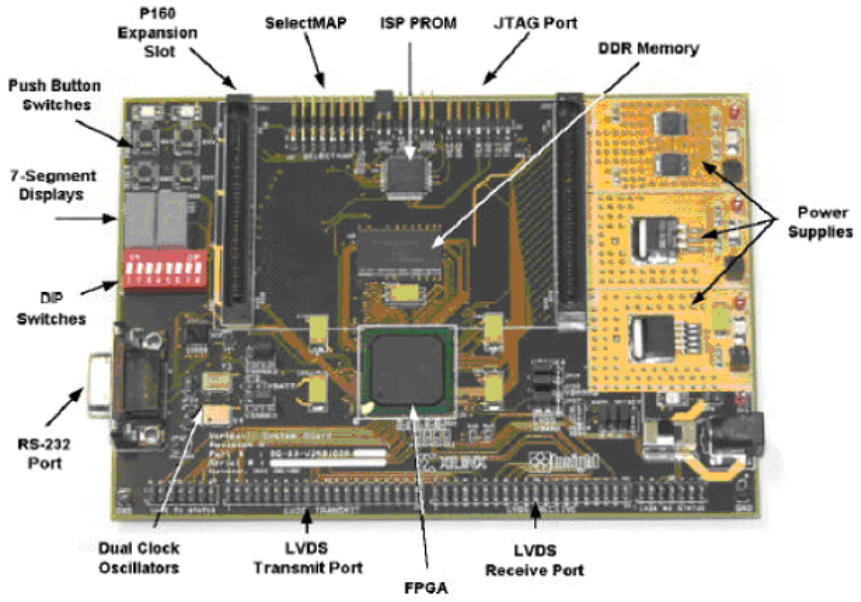

FIG. 3.9 – Carte de développement (Memec Design Virtex-II)

développés pour des applications haute performance telles que celles les télécommunications, l'imagerie et les applications DSP. Il possède 456 broches dont 324 peuvent être utilisées en entrées/sorties. Il se compose d'une matrice de 40x32 CLB et il contient un total de 10.240 LUT et 10.240 bascules (flip-flop). Sa capacité maximale en Select RAM est de 163.840 bits [37]. La carte contient également une mémoire DDR de 32MB. Elle présente 2 générateurs d'horloges internes, générant des signaux d'horloge à 100MHz (CLK.CAN2) et 24MHz (CLK.CAN1). Un troisième signal d'horloge externe est disponible et peut être utilisé si besoin est. Elle contient aussi circuit de remise à zéro (Reset) activé par un bouton poussoir (SW3), un bouton poussoir (PROGn) pour initialiser la configuration et charger le contenu de la PROM dans le circuit FPGA (SW2) ainsi que deux boutons poussoirs (SW5 et SW6) qui peuvent être utilisés pour générer des signaux actifs. Deux afficheurs 7 segments à cathode commune sont présents sur la carte, et peuvent être utilisés durant la phase de test et de debugging. Il existe aussi 8 entrées exploitables par l'utilisateur (DIP switch) qui peuvent être mis statiquement à un état haut ou bas. La carte possède une interface RS232, une interface JTAG pour programmer l'ISP PROM et configurer le circuit FPGA ainsi qu'un connecteur de câble parallèle IV qui peut aussi être utilisé pour configurer le FPGA via la configuration Maître/Esclave. Il existe également des régulateurs internes de tension qui génèrent, à partir de l'alimentation principale de 5,0V, des tensions internes d'alimentation à 1,5V, 2,5V et 3,3V. Les entrées/sorties sont regroupées dans 8 différents groupes, et chaque groupe peut être configuré pour opérer dans le mode 2,5V ou 3,3V. La carte de développement peut être configurée pour travailler en mode Master Serial, Slave Serial, Master SelectMap, Slave SelectMap ou JTAG selon la position des jumpers M0, M1, M2 et M3.

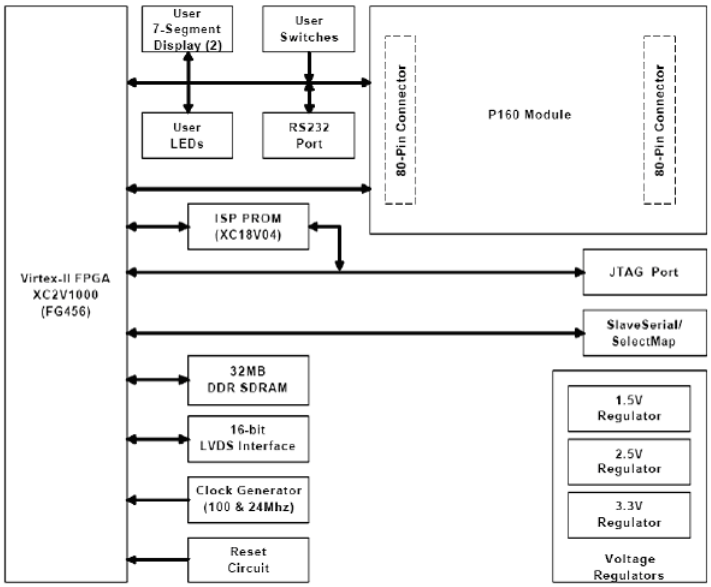

FIG. 3.10 – Diagramme de la carte de développement

3.3.2 Chargement du programme sur la carte de développement

La carte de développement Virtex-II supporte plusieurs méthodes de configuration de son circuit FPGA. Le port JTAG peut être utilisé directement pour configurer le FPGA, ou pour programmer l'ISP PROM. Une fois l'ISP PROM programmée, elle peut être utilisée pour configurer le FPGA. Le port SelectMap/Slave Serial sur cette carte peut aussi être utilisé pour configurer le FPGA. La figure.3.11 montre l'installation pour toutes les configurations de modes supportés par la carte de développement Virtex-II.

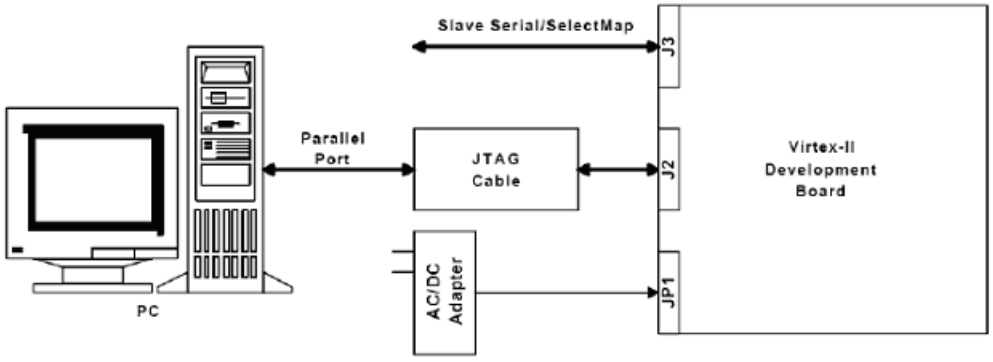

FIG. 3.11 – Chargement du programme sur la carte

Utilisation de l'interface JTAG

Le câble Memec Design JTAG est connecté d'un côté à la carte de développement, et de l'autre au port série du PC. On utilise alors l'outil de programmation du JTAG de Xilinx (iMPACT) pour charger le programme binaire soit directement sur le circuit FPGA en mode JTAG, soit sur l'ISP PROM en mode Master Serial

ou Master SelectMap. Dans ce dernier, cas il faut appuyer sur le bouton poussoir PROGn (SW2) pour initialiser la configuration dans le circuit FPGA.

Utilisation de l'interface Slave Serial

Dans ce mode, une source externe fournit la configuration bitstream et la configuration clock au circuit FPGA Virtex-II. Là aussi, le programme peut être directement chargé sur le circuit FPGA, ou bien sur l'ISP PROM, et dans ce cas il faut utiliser le bouton PROGn pour initialiser le FPGA.

3.4 Langage de description VHDL

Auparavant pour décrire le fonctionnement d'un circuit électronique programmable, les techniciens et les ingénieurs utilisaient des langages de bas niveau (ABEL, PALASM, ORCAD/PLD,..) ou plus simplement un outil de saisie de schémas. Actuellement la densité de fonctions logiques (portes et bascules) intégrées dans les PLD (Programmable Logic Device) est telle (plusieurs milliers voire millions de portes) qu'il n'est plus possible d'utiliser les outils d'hier pour développer les circuits d'aujourd'hui.
Les sociétés de développement et les ingénieurs ont voulu s'affranchir des contraintes technologiques des circuits. Ils ont donc créé des langages dits de haut niveau à savoir VHDL et VERILOG. Ces deux langages font abstraction des contraintes technologies des circuits PLD. Ils permettent au code écrit d'être portable, c'est-à-dire qu'une description écrite pour un circuit peut être facilement utilisée pour un autre circuit. Il faut avoir à l'esprit que ces langages dits de haut niveau permettent de matérialiser les structures électroniques d'un circuit. En effet les instructions écrites dans ces langages se traduisent par une configuration logique de portes et de bascules qui est intégrée à l'intérieur des circuits PLD. C'est pour cela qu'on préfère parler de description VHDL ou VERILOG que de langage. Dans ce qui suit, on s'intéressera uniquement au VHDL et aux fonctionnalités de base de celui-ci lors des phases de conception ou synthèse.
L'abréviation VHDL signifie VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. Ce langage a été développé dans les années 80 par le DoD (Departement of Defense) des Etats-Unis ; l'objectif était de disposer d'un langage commun avec les fournisseurs pour décrire les circuits complexes. En 1987, une première version du langage est standardisée par l'IEEE (Institute of Electrical and Electronics Engineers) sous la dénomination IEEE Std. 1076-1987 (VHDL 87). Une évolution du VHDL est normalisée en 1993 (IEEE Std. 1076-1993 ou VHDL-93) ; cette nouvelle version supprime quelques ambiguïtés de la version 87, et surtout met à disposition de nouvelles commandes. Actuellement, tous les outils dignes de ce nom supportent le VHDL-93. En 1997, de nouvelles libraires ont été normalisées de manière à ajouter des fonctions pour la synthèse (conception) de circuits logiques programmables PLD, et plus particulièrement les FPGA. La dernière révision est celle de 2002 (IEEE Std. 1076-2002 ou VHDL-2002).

3.5 Relation entre une description VHDL et un circuit FPGA

L'implantation d'une ou de plusieurs descriptions VHDL dans un circuit FPGA va dépendre de l'affectation que l'on fera des broches d'entrées/sorties et des structures de base du circuit logique programmable.
La (figure.3.12) représente un exemple de descriptions VHDL ou de blocs fonctionnels implantés dans un FPGA. Lors de la phase de synthèse chaque bloc sera matérialisé par des portes et/ou des bascules. La phase suivante sera d'implanter les portes et les bascules à l'intérieur du circuit logique. Cette tâche est réalisée par le logiciel placement/routage (Fitter), au cours de laquelle les entrées et sorties seront affectées à des numéros de broches. On peut remarquer sur le schéma la fonction particulière du bloc VHDL N °5. En effet dans la description fonctionnelle d'un FPGA on a souvent besoin d'une fonction qui sert à cadencer le fonctionnement de l'ensemble, celle-ci est très souvent réalisée par une machine d'états synchronisée par une horloge,voir (figure.3.12).

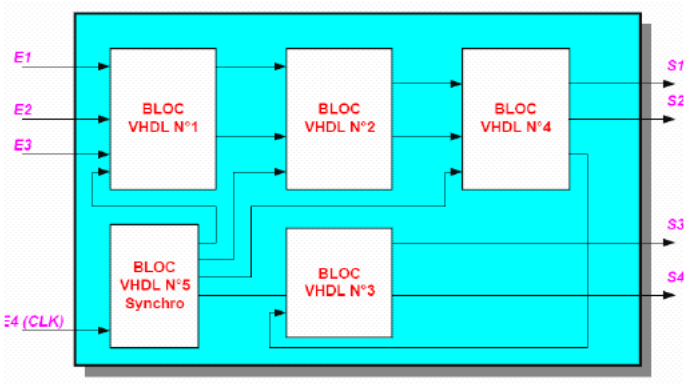

FIG. 3.12 – Schéma fonctionnel d'implantation d'une description VHDL dans un circuit FPGA

3.5.1 Saisie du texte VHDL

La saisie du texte VHDL se fait sur le logiciel « ISE Xilinx Project Navigator »,propose une palette d'outils permettant d'effectuer toutes les étapes nécessaires au développement d'un projet sur circuit FPGA. Il possède également des outils permettant de mettre au point une entrée schématique ou de créer des diagrammes d'état, qui peuvent être utilisés comme entrée au lieu du texte VHDL. figure.3.13 montre comment se présente le logiciel (ISE Xilinx Project Navigator).

3.5.2 Synthèse

La synthèse permet de réaliser l'implémentation physique d'un projet. Le synthétiseur a pour rôle de convertir le projet, en fonction du type du circuit FPGA cible utilisé, en portes logiques et bascules de base. L'outil « View RTL Schematic » permet de visualiser les schémas électroniques équivalents générés par le synthétiseur .
De plus, le synthétiseur permet à l'utilisateur d'imposer de technologie (User contraints) :

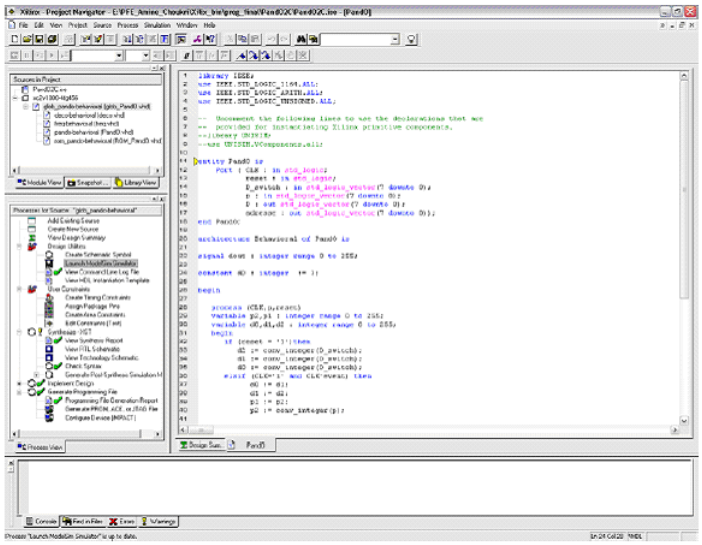

FIG. 3.13 – vue d'ensemble logiciel xilinx

par exemple fixer la vitesse de fonctionnement (Create Timing Constraints), délimiter la zone du circuit FPGA dans laquelle le routage doit se faire (Create Area constraints) ou affecter les broches d'entrées/sorties (Assign Package Pins).

3.5.3 Simulation

La simulation permet de vérifie le comportement d'un design avant ou après implémentation dans le composant cible,le simulateur utilisé est Modelsim .

3.5.4 Optimisantion ,placement et routage

Pendant l'étape d'optimisation, l'outil cherche à minimiser les temps de propagation et a occuper le moins d'espace possible sur FPGA cible .

3.5.5 programation du composant et test

Dans cette dernière étape (generate programming files) ,on génère le ficher à charger sur le circuit FPGA à travers l'interface JTAC .

Chapitre 4

Développement d'un projet sur DSP

4.1 Introduction

De nos jours, le DSP (Digital Signal Processor) présente une puissance importante de calcul et d'implémentation des techniques numériques de traitement du signal. Cette puissance est assurée par sa grande vitesse d'exécution, ses fonctions spéciales et son jeu d'instructions optimisé pour le traitement numérique du signal et de l'automatisme. En effet, tout système fondé autour d'un DSP bénéficie des avantages dérivant de ses particularités architecturales et de programmatives. On peut citer : le temps réel, la flexibilité, la fiabilité et la réduction des coûts.
A fin de développer un projet sur DSP, il fondra prendre du DSP à utiliser choix fait (selon l'application et après étude du cahier de charges), et des outils aussi bien logiciel que matériels fournis par le constructeur. Nous allons fournis par le constructeur .Nous allons aussi présenter dans ce chapitre les différents outils que nous rencontrons dans tout traitement a base de DSP.

4.2 Spécificités architecturales

4.2.1 Architecture de Von Neumann

La structure de Von Neumann consiste à mettre les données et le programme dans la même zone mémoire, et donc utiliser un seul bus de données et d'adresses pour les données et le programme (les instructions). La figure.4.1 représente le schéma en blocs de cette architecture .
Il est bien clair qu'on ne peut lire une instruction ou une donnée qu'en un seul

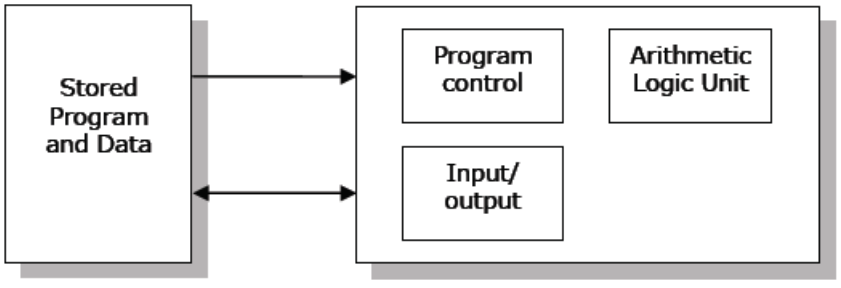

Fig. 4.1 – Architecture de Von Neumann

cycle d'horloge ; donc pour l'exécution d'une instruction il faut souvent plusieurs cycles d'horloge. Est un inconvénient majeur pour les applications dont le temps d'exécution est primordial. Un DSP dédié pour des applications en temps réel ne peut être servi par une telle structure.

4.2.2 Architecture de Harvard

Caractérisée par la séparation entre mémoire données et mémoire programme figure.4.2, cette structure offre la possibilité d'accéder aux données et aux instructions du programme en même temps (en un cycle d'horloge), grâce à deux bus distincts de données et deux bus distincts d'adresses, cela permet une grande souplesse pour l'enregistrement et l'utilisation des données.

Ce type de structure, utilisée par les microcontrôleurs, permet d'avoir de grandes

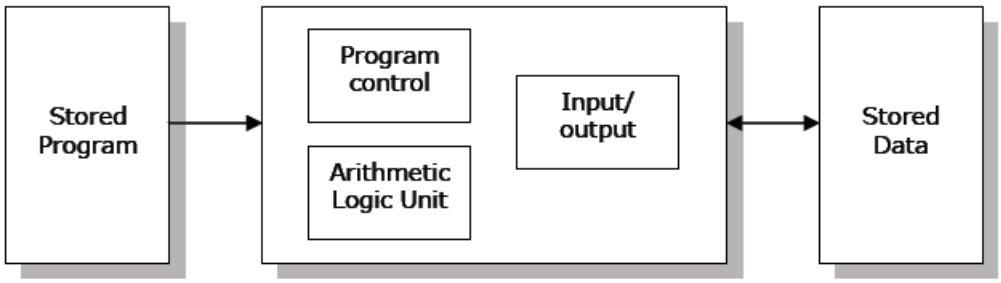

FIG. 4.2 – Architecture de Harvard

vitesses d'exécution. Cependant, l'augmentation du nombre de boîtiers mémoires et donc l'augmentation du nombre de bus rend le processeur très coûteux.

4.2.3 Architecture d'un DSP

A priori, pour augmenter le rapport performance/prix, les concepteurs de DSP ont utilisé une structure de Harvard dite structure de Harvard modifiée. Il s'agit de réaliser deux bus distincts de données et deux bus distincts d'adresse pour les accès en mémoires internes (architecture de Harvard), et pour les accès en mémoires externes deux bus l'un pour les données et l'autre pour les adresses (architecture de Von Neumann). Les transferts de données entre les bus internes et externes s'effectuent par multiplexage temporel.

Les mémoires

Un DSP a la capacité de manipuler deux opérandes, ou même plus, via une seule instruction. Donc il doit réaliser au minimum un double accès mémoire en un cycle d'horloge. Pour les faire, en plus de la RAM (deux boîtiers de mémoire vive au minimum) et de la ROM, et dans certains DSP, une autre mémoire est utilisée pour stocker les instructions répétitives appelée (Cache Memory),elle permet de libérer les deux bus de données internes pour véhiculer les opérandes et de compenser le temps d'accès en ROM qui est critique par rapport à celui d'une RAM.

4.3 Spécificités de programmation

4.3.1 Spécificités de programmation

On peut accéder aux données à partir d'une mémoire, des registres, et des instructions en utilisant les modes d'adressage suivants .

- Adressage registre : l'opérande est dans l'un des registres du CPU.
- Adressage direct : l'opérande est dans l'adresse mémoire spécifiée par l'instruction.
- Adressage indirect : l'adresse de l'opérande est rangée dans un registre auxiliaire.
- Adressage immédiat : opérande encodé dans l'instruction.
- Adressage relatif au PC : utilisé lors des branchements, le déplacement vers la destination est généré par l'assembleur.

En plus de ces modes d'adressage standards, et en fonction du type de DSP, d'autres modes d'adressages spéciaux existent, destinés à optimiser l'exécution de certains algorithmes répandus tels que les filtres, FFT, convolution, corrélation...etc. On peut citer :

- l'adressage circulaire : un registre auxiliaire est incrémenté/décrémenté en respectant les limites du buffer circulaire. Selon ces limites deux versions existent : (@départ, @arrivée) et (@départ, longueur).
- l'adressage par inversion de l'ordre des bits : (figure.4.3) utilisé dans le calcul de la FFT.

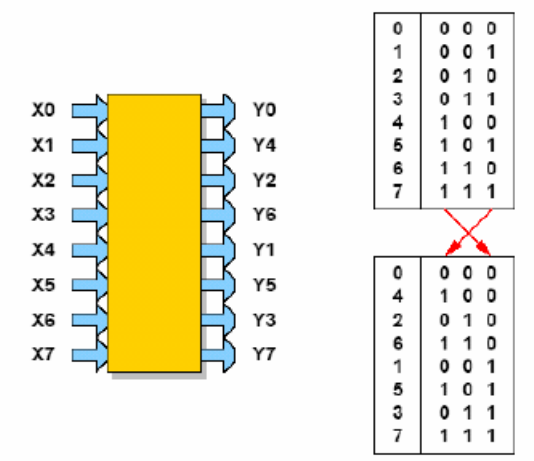

FIG. 4.3 – L'inversion de l'ordre des bits

4.3.2 Pipeline

On a quatre niveaux de pipeline ,représentés dans la figure.4.4 :

- FETCH : recherche de l'instruction par accès à la mémoire.
- DECODE : décodage de l'instruction et des adresses des opérandes suivant le mode d'adressage.

– READ : lecture des opérandes en mémoire.
– EXECUTE : exécution de l'opération et écriture du résultat.

CYCLE	Fetch	Decode	Read	Execute
m-3	w	-	-	-
m-2	x	w	-	-
m-1	y	x	w	-
M	z	y	x	w
m+1	-	z	y	x
m+2	-	-	z	y
m+3	-	-	-	z

FIG. 4.4 – Structure d'un pipeline

Certains conflits de pipeline sont souvent gérés par le DSP, tel que les cassures dues aux instructions de branchement. mais dans certains cas particuliers et suivant le DSP utilisé, l'utilisateur doit prendre des précautions pour assurer un bon fonctionnement.

4.4 Arithmétique des DSP

Il existe deux types de DSP, les DSP à virgule fixe et les DSP à virgule flottante. La figure.4.5 donne un aperçu sur les fournisseurs de DSP, avec la famille et le format .

Rang mondial	Vendeur	Familles de DSP	Format	Données (bits)	Puissance (MIPS)
1	Texas Instruments	TMS320C1x	Fixed	16	8.8
		TMS320C2x	Fixed	16	40
		TMS320C3x	Floating	32	25
		TMS320C4x	Floating	32	30
		TMS320C5x	Fixed	16	50
		TMS320C6x	Fixed	32	2400
2	Analog Devices	ADSP-21xx	Fixed	16	40
		ADSP-21	Floating	32	40
3	Motorola	DSP560xx	Fixed	24	40
		DSP563xx	Fixed	24	80
		DSP568xx	Fixed	16	70
		DSP960xx	Floating	32	40
		StarCore	Fixed	16	1200
4	Lucent	DSP16xx	Fixed	16	40
		DSP32xx	Floating	32	40

FIG. 4.5 – Fournisseurs de DSP et familles associées

4.4.1 Arithmétique en virgule fixe (fixed point)

On utilisera la notation pratique Qi(n) avec n nombre de bits max d'un mot (deux formats existent sur 24 bits, et le plus courant sur 16 bits) et i le nombre de bits fractionnaires.

Entiers signés

format Q0(16) (figure.4.6).

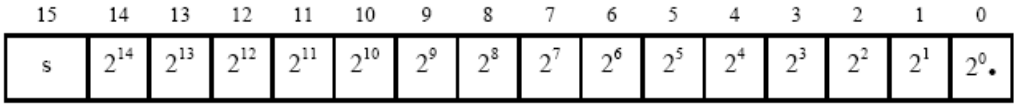

FIG. 4.6 – Format sur 16 bits d'un entier signé

- Dynamique : $-32767 < x < 32767$ (-32768 existe mais est interdit car alors étendue non symétrique pouvant poser des problèmes).
- Erreur max absolue : ± 0.5 si obtenue après arrondi au plus près.
- Erreur max relative : $\pm 0.5/x$ l'erreur relative augmente fortement pour x petit.

Nombres signés fractionnaires

(figure.4.7), de module < 1 : format Q15(16).

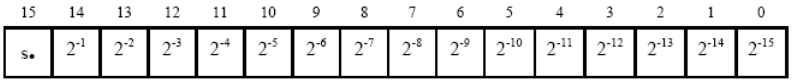

FIG. 4.7 – Format sur 16 bits d'un nombre signé fractionnaire

- Dynamique : -1<x<1 (-1 existe mais interdit)
- Erreur max absolue : $\pm 1/2LSBsoit \pm 2^{-16} = \pm 1.510^{-5}$
- Erreur max relative : $1.510^{-5}/x$ tend vers l'infini si x tend vers zéros.

4.4.2 Arithmétique en virgule flottante (floating point)

format $X = M2^E$ avec :
M, mantisse en mode (Signe-valeur absolue) et $1 = |M| < 2$.
E, exposant.
S, signe de la mantisse.
En virgule flottante, un nombre est codé sur 32 bits, plusieurs représentations sont possibles.

4.5 La famille de DSP de Texas Instruments TMS320

En 1982, Texas Instruments a lancé son premier DSP : le TMS3210. Depuis, Texas instruments s'est imposé comme le principal fabriquant des processeurs de

traitement de signal. La famille de DSP TMS320 est organisée aujourd'hui en cinq générations principales qui correspondent chacune à une classe de performances et d'applications. Elles sont représentées par les sigles suivants :

- TMS320C1X
- TMS320C2X
- TMS320C3X
- TMS320C4X
- TMS320C5X
- TMS320C6X
- TMS320C8X

Les processeurs des générations 1,2 et 5 travaillent en format fixe, ceux de la quatrième génération travaillent en format flottant. Une compatibilité logicielle (assembleur) ascendante est maintenue pour chaque famille. A ces cinq générations, s'ajoutent depuis peu de nouveaux composants comme le TMS320C80, destinés aux applications multimédia et vidéo.

4.6 Architecture du TMS320C6x

Le TMS320C6211 mis en oeuvre sur la carte DSK, qui servira d'un support de développement d'applications par la suite, est un processeur à virgule fixe basé sur l'architecture VLIW(Very-Long-Instruction-Word). Sa mémoire interne inclut deux niveaux de mémoire cache avec 4kB pour le niveau 1 concernant la cache programme (L1P), 4kB pour le niveau concernant la cache données (L1D) et 64kB de RAM en niveau 2 pour les allocations données/programme (L2). L'environnement comprend des mémoires synchrones (SDRAM) et des mémoires asynchrones (SRAM et EPROM). Les périphérique du DSP en question sont : deux ports séries multicanaux avec buffer (McBSPs), deux timers, un port d'interfaçage hôte de 16 bits (HPI) et une interface mémoire externe de 32 bits (EMIF). Il requière 3.3 V pour I/O et 1.8 V pour le coeur.
Le chemin de données (bus internes) est formé d'un bus d'adresse programme de 32bits, d'un bus programme de 256 bits pour faire loger huit instructions de 32bits, deux bus d'adresse de données 32bits, deux bus de données 32 bits et deux bus de données store de 64bits. Avec un bus de 32 bits pour les adresses, l'espace mémoire total adressable est de $2^{32} = 4GB$, incluant quatre espaces mémoires externes : CE0, CE1, CE2, et CE3.la (figure.4.8) montre le diagramme en blocs des parties fonctionnelles du C6211.
L'indépendance des bancs mémoires des processeurs C6x permet un double accès en mémoire en un seul cycle d'instruction, et donc deux instructions loads ou stores peuvent être exécutées en parallèle tant que les données à y accéder résident dans différents blocs. Le C6211 du DSK inclut 72kB de mémoire interne adressée à partir de l'adresse 0x00000000, et 16MB de SDRAM adressée via CE0 à partir de l'adresse 0x80000000. Le DSK inclut aussi une mémoire flash de 128kB à partir de 0x90000000. La (figure.4.9) montre la configuration mémoire du DSP C6211. [25] [38]

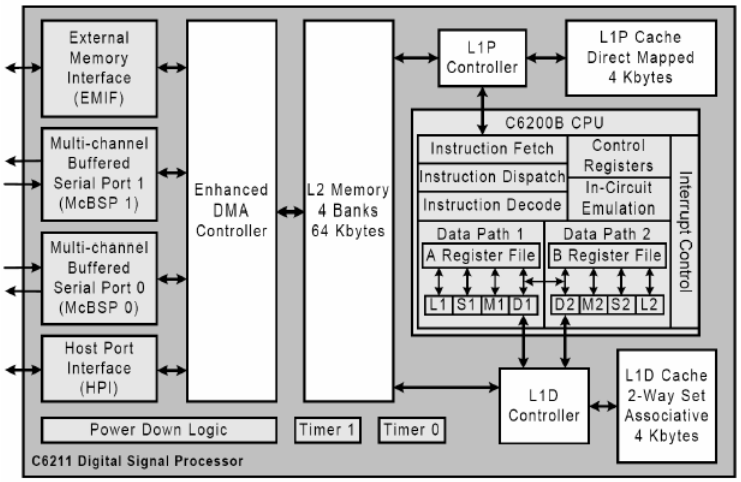

FIG. 4.8 – Block Diagram du TMS320C6211

Address Range (Hex)	Size (Bytes)	Description of Memory Block
0000 0000—0000 FFFF	64K	Internal RAM (L2)
0001 0000—017F FFFF	24M–64K	Reserved
0180 0000—0183 FFFF	256K	Internal configuration bus EMIF registers
0184 0000—0187 FFFF	256K	Internal configuration bus L2 control registers
0188 0000—018B FFFF	256K	Internal configuration bus HPI register
018C 0000—018F FFFF	256K	Internal configuration bus McBSP 0 registers
0190 0000—0193 FFFF	256K	Internal configuration bus McBSP 1 registers
0194 0000—0197 FFFF	256K	Internal configuration bus timer 0 registers
0198 0000—019B FFFF	256K	Internal configuration bus timer 1 registers
019C 0000—019F FFFF	256K	Internal configuration bus interrupt selector registers
01A0 0000—01A3 FFFF	256K	Internal configuration bus EDMA RAM and registers
01A4 0000—01FF FFFF	6M–256K	Reserved
0200 0000—0200 0033	52	QDMA registers
0200 0034—2FFF FFFF	736M–52	Reserved
3000 0000—3FFF FFFF	256M	McBSP 0/1 data
4000 0000—7FFF FFFF	1G	Reserved
8000 0000—8FFF FFFF	256M	External memory interface CE0
9000 0000—9FFF FFFF	256M	External memory interface CE1
A000 0000—AFFF FFFF	256M	External memory interface CE2
B000 000—BFFF FFFF	256M	External memory interface CE3
C000 0000—FFFF FFFF	1G	Reserved

FIG. 4.9 – Résumé,Configuration Mémoire

4.7 Unités fonctionnelles

Le CPU consiste en huit unités fonctionnelles indépendantes divisées en deux chemins de données A et B telle que montrés en (figure.4.10). Chaque chemin comporte une unité pour la multiplication (.M), pour les opérations logiques et arithmétiques (.L), pour les branchements, la manipulation des bits et opérations arithmétiques (.S), et pour le loading/storing et opérations arithmétiques (.D). Les unités .S et .L sont utilisées par les instructions arithmétiques, logiques, et de branchements. Tous les transferts de données se font via les unités .D. [35]
Les opérations arithmétiques telles que celles d'addition ou de soustraction (ADD ou SUB), peuvent être exécutées par toutes les unités exceptées les unités .M. Les huit unités fonctionnelles consistent en quatre unités arithmétiques et logiques ALUs travaillant en virgule fixe (deux .L et deux .S), deux autres unités arithmétiques virgule fixe (.D), et deux multiplieurs virgule fixe (.M). Chaque unité fonctionnelle peut lire directement du ou écrire directement dans un registre du fichier à travers son propre chemin. Chaque chemin inclut un ensemble de seize registres de 32bits, de A0 à A15 et de B0 à B15. Donc il y a 32 registres à usage général ; certains sont réservés pour les modes d'adressage spéciaux ou utilisés par les instructions conditionnelles.
Deux chemins croisés (cross-paths 1x et 2x) permettant aux U.F à partir d'un chemin de données l'accès à un opérande de 32bits dans un fichier du registre du banc opposé. Un maximum de deux accès par cycle utilisant le cross-path est permis (figure.4.10)

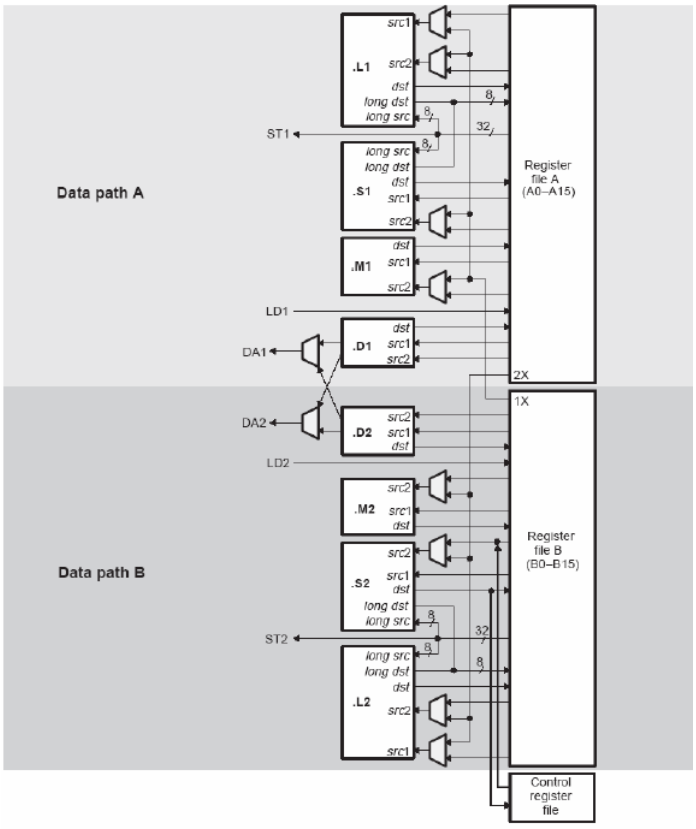

FIG. 4.10 – Chemins de données TMS320C62x

4.8 Fetch et excute packets

L'architecture VELOCITI, introduite par TI, est une dérivée de l'architecture VLIW. L'execute packet (EP) consiste en un groupe d'instructions exécutées en parallèle en un seul cycle d'horloge. Le nombre de EP inclus dans un fetch packet (FP) peut varier de un (avec huit instructions parallèles) à huit (sans instructions parallèles). L'architecture VLIW est modifiée pour permettre plus d'un EP. [25]
Le bit le moins significatif de chaque instruction de 32bits sert à déterminer si l'instruction suivante appartient au même EP (si 1) ou fait partie de l'EP suivant (si 0).
Pour illustrer, considérons un FP avec trois EP tel que : EP1 avec deux instructions parallèles, EP2 et EP3 chacun avec trois instructions parallèles ,La (figure.4.11).
Donc FP est tel qu'il apparaît sur La (figure.4.12), Le bit 0 de chaque instruction de

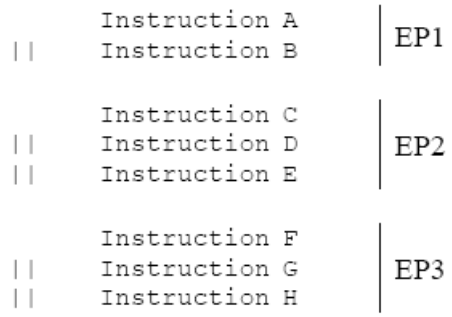

FIG. 4.11 – instructions parallèles

32bits est le bit 'p', qui signale la présence ou non d'un parallélisme avec l'instruction suivante.

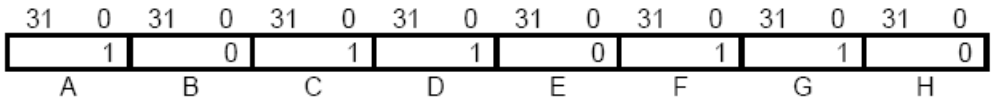

FIG. 4.12 – Un FP avec trois EPs, montrant le bit 'p' de chaque instruction

4.9 Consideration mémoire

4.9.1 Allocation de données

Des blocs de code et de données peuvent être répartis dans la mémoire sous forme de sections définies dans le fichier de commande (.cmd). Ces sections peuvent être initialisées ou non initialisées. Les sections initialisées ou non initialisées, excepté .text, ne peuvent pas être définies dans la mémoire interne de programme . Les sections initialisées sont :

- .cinit : pour les variables globales et locales,
- .const : pour les constantes globales et locales,

- .switch : contient les vecteurs des sauts,
- .text : pour le code exécutable et les constantes ;

et les sections non initialisées :

- .bss : pour les variables globales et locales,
- .far : pour les variable globales et locales déclarées ,
- .stack : réserve de la mémoire pour la pile,
- .sysmem : réserve l'espace pour les allocations de mémoire utilisées par les fonctions telles que malloc, calloc et realloc.

Le linker peut être utilisé pour placer certaines sections, telle que .text, dans la mémoire interne pour plus d'efficacité des opérations.

4.9.2 Alignement de données

Le C6x accède toujours aux données alignées qui lui permettent d'adresser des bytes, des mots de 16 bits, et des mots de 32 bits. Le format de données se compose soit de quatre limites de byte, soit de deux limites de mot 16bits, ou d'une limite de mot 32bits. Par exemple, pour assigner un chargement de 32 bits avec LDW, l'adresse doit être alignée avec une limite de mot 32bits de sorte que les 2 bits inférieurs de l'adresse soient zéro. Autrement, des données incorrectes peuvent être chargées.

4.9.3 Type de données

Les types de données supportées par la famille C6000 en général sont :

- .short : de taille 16 bits représenté en complément à 2, avec une plage de variation de -2^{15} à $(-1+2^{15})$
- .int or signed int : de taille 32 bits représenté en complément à 2, avec une dynamique entre -2^{32} à $(-1+2^{15})$
- .float : de taille de 32 bits représentée selon la norme IEEE 32-bit avec une dynamique de $2^{-126} = 1.175494 \times 10^{-38}$ jusqu'à $2^{+128} = 3.40282346 \times 10^{38}$.
- .double : de taille de 64 bits représentée selon la norme IEEE 64-bit avec une dynamique de $2^{-1022} = 2.22507385 \times 10^{-308}$ jusqu'à $2^{+1024} = 1.79769313 \times 10^{+308}$.

Le processeur C6211, qui est un DSP virgule fixe, ne supporte pas ces deux derniers types.

4.10 Outils de génération du code et de développement

L'utilisation des DSP étant relativement sophistiquée, il est important de disposer d'outils de développement performants et conviviaux qui permettent de minimiser le temps de mise sur le marché d'une nouvelle application. Texas Instruments a

développé une série d'outils logiciels et matériels, ainsi qu'un système d'aide et d'information à l'utilisateur, exploitant un site très riche. Certains de ce outils offrent notamment un environnement de développement intégré matériel et logiciel : IDE (Integrated Development Environnement). [34]

4.10.1 Outils de développement logiciel

Les principaux outils de développement logiciel de cette famille sont listés dans le tableau qui suit. Il fonctionnent généralement sur PC Windows ou NT.
Ces outils permettent de développer du code, de le simuler et de le déboguer. [34]

Outils de développement logiciels	Assembleur éditeur de lien compilateur C
Outils de simulation et de déboggage	simulateur et interface de déboggage DOS ou Windows , Environnement de développement intégré :code composer

TAB. 4.1 – Outils de développement logiciels

Fichier source assembleur/C

Un fichier source assembleur est créé à l'aide d'un éditeur de texte. Il comprend des lignes d'instructions, de directives assembleur, de directives de macro et de commentaires. L'extension de ce fichier est (.asm). Un fichier source écrit dans le langage évolué C porte l'extension (.c)et est traduit en assembleur par le compilateur C. La programmation C est intéressante pour réduire le temps de développement d'une application, sa portabilité dans d'autres systèmes et elle ne nécessite pas une connaissance approfondie de l'architecture interne du DSP.
On l'utilise pour les parties de programme dont le temps d'exécution n'est pas critique mais elle occupe plus d'espace mémoire et donc est moins optimisée. On utilise la programmation en assembleur pour les blocs de programmes effectuant des calculs arithmétiques intensifs.

Assembleur et éditeur de liens

Après avoir écrit un programme à l'aide d'un éditeur de texte (en assembleur ou en C), il faut assembler puis lier le programme pour obtenir un programme exécutable.

L'assembleur

Il traduit les fichiers sources écrits en langage assembleur du DSP en un fichier objet en langage machine au format COFF (Common Objet File Format). Les fichiers source peuvent contenir des instructions, des directives assembleur. Les directives assembleur peuvent être utilisées pour contrôler différents aspects du processus d'assemblage : le format listing du code source, l'alignement des données, le contenu des sections.

L'éditeur de liens (Linker)

L'éditeur de liens combine plusieurs fichiers objet relogeables en format COFF(créés par l'assembleur) en entrée, pour générer un fichier COFF exécutable par un DSP. Il effectue les relocations et résout les problèmes de références externes. Les directives de cet éditeur nous permettent de combiner les sections du fichier objet, lier les sections du fichier objet, lier les sections ou les symboles aux adresses ou dans les rangs de la mémoire et définie et redéfinie les symboles globaux.

Le fichier en format COFF et sections associées

Les outils de langage assembleur créent et utilisent les fichiers objets dans le format COFF (Common Object File Format) pour faciliter la programmation modulaire. On peut programmer efficacement les DSP si l'on comprend les bases du format COFF. Les fichiers objets contiennent des blocs séparés (appelés sections) de code et de données que l'on peut charger dans l'espace mémoire du DSP. Une section est un bloc de code ou de données qui va occuper un espace dans la mémoire.

4.10.2 outils de déboggage software

Le simulateur

C'est un programme software qui permet de simuler l'exécution d'un programme écrit en C ou en assembleur pour un DSP. Le simulateur peut exécuter les fichiers objet COFF linkés. Il permet la vérification d'un programme sans les aspects temps réel. La simulation logicielle est effectuée sur une machine d'usage général, un PC ou une station de travail et les entrées/sorties utilisent en général des fichiers.

L'environnement de développement intégré Code Composer

C'est un environnement de développement intégré(IDE :Integrated Development Environment) , commun aux différents DSPs de Texas Instruments, qui peut s'utiliser seul ou en lien avec une cible matérielle fonctionnant en temps réel telle qu'une carte d'évaluation . Cet environnement permet de construire un projet, d'éditer les fichiers, de les compiler ou les assembler, de faire l'édition de liens, de tester et de débogger une application mono ou multiprocesseur. Les cibles matérielles supportées sont les cartes d'évaluation EVM, les kits de développement DSK , ainsi que les émulateurs. [39] [33][24][25]

4.11 Optimisaton du code

Des techniques d'optimisation, telle que l'utilisation des options du compilateur et les 'intrinsics' en addition avec les directives pragma déjà traitées, présentent la deuxième étape après validation du fonctionnement du code C et avant le passage à une optimisation au niveau assembleur.

4.11.1 Options du compilateur

Quand l'optimisateur est appelé, les étapes suivantes sont exécutées. Un programme en C d'abord passé par un analyseur syntaxique qui exécute des fonctions

de prétraitement et génère un fichier intermédiaire (.if) qui devient le fichier d'entrée de l'optimisateur. L'optimisateur génère par la suite un fichier (.opt) qui passe par le générateur de code pour d'autres optimisations et obtenir un fichier ASM (figure.4.13).
Les options sont, en résumé :

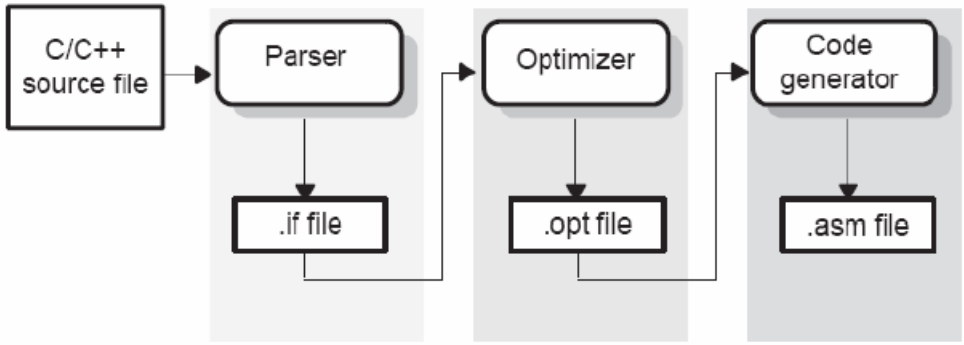

FIG. 4.13 – Compilation d'un programme C/C++ avec optimisations

- -o0 : optimise l'utilisation des registres.
- -o1 : optimisation locale des fonctions en addition à l'optimisation précédente.
- -o2 : optimisation globale en addition des optimisations -o0 et -o1.
- -o3 : optimisation du fichier plus les optimisations précédentes.

4.11.2 Fonction C « intrinsics »

Des fonctions en C, appelées intrinsics functions, sont disponibles pour augmenter l'efficacité du code exécutable :

- $int - mpy()$: l'équivalent de l'instruction MPY en ASM, qui multiplie les 16 LSBs d'un nombre par les 16 LSBs d'un autre nombre.
- $int - mpy()$: l'équivalent de l'instruction MPYH en ASM, qui multiplie les 16 MSBs d'un nombre par les 16 MSBs d'un autre nombre.
- $int - mpylh()$: l'équivalent de l'instruction MPYLH en ASM, qui multiplie les 16 LSBs d'un nombre par les 16 MSBs d'un autre nombre.
- $intm - pyhl()$: l'équivalent de l'instruction MPYHL en ASM, qui multiplie les 16 MSBs d'un nombre par les 16 LSBs d'un autre nombre.
- $int - sadd()$: l'équivalent de l'instruction SADD en ASM, faisant la somme de deux nombres entiers et sature le résultat.
- $int - ssub()$: l'équivalent de l'instruction SSUB en ASM, faisant la soustraction entre deux nombres entiers et sature le résultat.

Il est clair qu'il s'agit de la version C des fonctions fréquemment utilisées qui permettent un accès direct aux équivalentes en ASM. [39]

4.12 Le kit d'évaluation DSK (DSP Starter Kit)

C'est un système à très faible coût, contenant une carte DSP, une alimentation, un débogger et des outils de génération de code assembleur spécifiques au système DSK. La carte se connecte à un PC par un port parallèle. Elle contient un DSP fonctionnant à 150MHz et une interface analogique /numérique et numérique/analogique et bien adapté aux signaux de parole voir (figure.4.14). La carte utilisée pour le dé-

FIG. 4.14 – Kit d'évaluation DSK

veloppement est formée des composants suivants voir (figure.4.15) :

- DSP TMS320C6211 à 150MHz
- 16 M Bytes de mémoire externe SDRAM 100MHz
- 128 K Bytes de mémoire FLASH programmable et effaçable
- port parallèle d'interfaçage (SPP) entre la carte DSK et le PC hôte
- port d'interfaçage hôte permet l'accès à toutes les ressources mémoire du DSP à travers le port parallèle (moyen d'implémentation du code)
- outil embarqué JTAG (Joint Test Action Group), accès en direct par le support XDS510 ou bien par émulation via le port parallèle
- 16-bit audio codec
- quatre LED d'indications utilisées lors de l'amorçage
- deux supports d'extension de mémoire ou périphérique

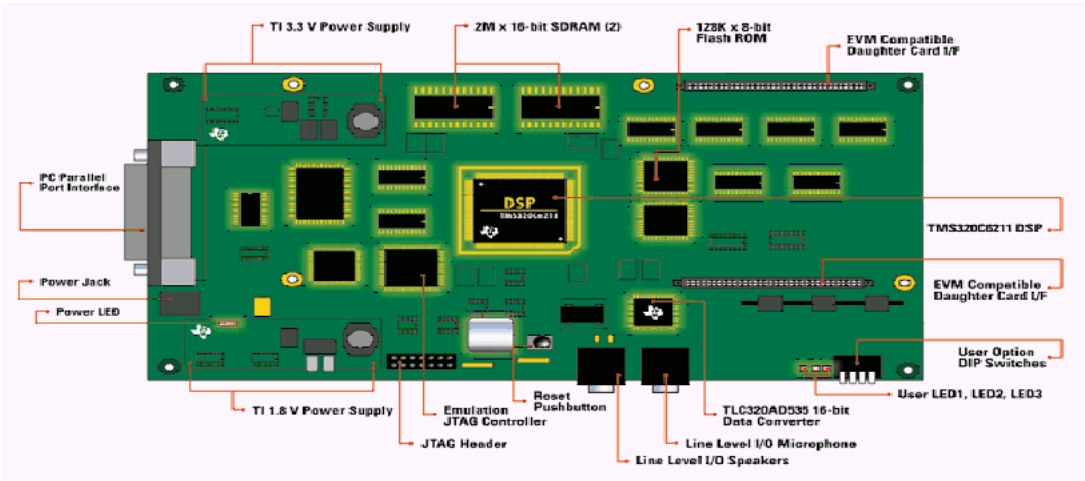

FIG. 4.15 – TMS320C6711-Carte DSK

4.13 La carte d'évaluation

Le module d'évaluation EVM est un système de faible coût comprenant une carte PC ISA 16 bits avec un DSP, un débogger et des outils standards de génération de code assembleur fonctionnant sous Windows 95,98 et NT. L'interface graphique du débogger est identique à celle du simulateur. C'est une carte interne au PC qui inclut un DSP, une interface de conversion analogique/numérique et numérique/analogique, une mémoire RAM, une interface avec le PC et une interface d'émulation. Il offre une rapidité dans l'émulation et le déboggage hardware.

4.14 Outils d'émulation

Les outils d'émulation servent à mettre au point une application pour laquelle l'utilisateur a développé le matériel et le logiciel. Il faut dans ce cas tester à la fois le comportement du logiciel et du matériel. Le système XDS510 (Extended Development Support) est un système d'émulation pleine vitesse qui se compose d'une carte JTAG insérable dans un bus PC et d'un câble JTAG (nous avons utilisé une porte parallèle) . Il permet au débogger de contrôler le DSP sur une carte matérielle, cette carte étant connectée à l'émulateur par le câble d'émulation JTAG relié au port JTAG du DSP. Cet émulateur permet de démarrer l'exécution d'un programme, de l'arrêter, de faire du pas à pas, de mettre des points d'arrêt logiciels et de faire une trace à n'importe quelle adresse programme ou donnée, de charger et inspecter les registres, les mémoires ; mesurer le temps d'exécution d'un programme. Il existe sur PC et sur station SPARC. Il fonctionne avec le débugger Code Composer ou le débugger TMS320.

4.15 Conclusion

Les DSPs sont des processeurs dédiés signal qui ont révolutionné les systèmes embarqués. grâce à leur architecture particulière et à leurs périphériques intégrés qui leur procurent puissance et rapidité. Les DSPs TMS320C6211 sont dédiés au traitement de l'image et sont à faible consommation .Ce sont des DSPs travaillant en virgule fixe pour le codage des données réelles . Le constructeur Texas Instruments a mis au service de l'utilisateur un ensemble d'outils de développement logiciel et matériel riche et simplifié.

Chapitre 5

Simulation et évaluation des résultats

5.1 Introduction

L'objet de notre étude est l'implémentation sur circuit FPGA et sur DSP de la méthode MPPT neuronal déjà étudiée dans les chapitres précédents.
Dans ce dernier chapitre, on présente la simulation du contrôleur neuronale, et on effectue plusieurs simulations pour valider théoriquement l'étude et le choix de la structure du contrôleur, puis on passe aux différentes étapes de développement du projet sur FPGA à partir des programmes écrits en langage VHDL et.En fin nous présents étapes de développement sur DSP à partir des programmes écrits en langage C.

5.2 Simulation sur SIMULINK

Dans la simulation de la technique MPPT-neuronal on a utilisé 'SIMULINK' de 'MATLAB' en raison de la possibilité de simuler des systèmes mixtes (continus et discrets).
Le système continu est utilisé pour la simulation des différentes parties analogiques (panneau solaire, batterie, MPPT analogique, convertisseur DC-DC). Le système discret est utilisé pour simuler la méthode de tracking algorithme neuronal.
La chronologie des étapes à poursuivre pour élaborer un contrôleur neuronal a été décrit au chapitre 2, et sa simulation et la simulation des différents blocs sont détaillées dans l'annexe [B].
Nous allons essayer d'appliquer dans ce qui suit un réseau de neurones comme algorithme de commande, et ainsi, de confirmer son efficacité dans la commande MPPT-neuronal.
Son fonctionnement de base réside dans le fait qu'après l'étape de l'apprentissage, il sera inséré dans une architecture approprié figure. 5.1, où il doit estimer la commande adéquate indiquant l'emplacement réel du PPM, à partir des données d'entrée qui sont respectivement la température (T), l'ensoleillement (S) et la dynamique de la batterie (V_b).
Ce dernier montre que sur chaque cycle de l'algorithme principal de commande, une mesure des variables d'entrées du réseau qui sont respectivement T (°C), S (W/ m^2) et V_b(Volts) est effectuée, puis une propagation directe de ces données via le réseau de neurones. Cette dernière estime la commande adéquate à sa sortie qui sera appliquée à l'entrée commande du hacheur.

Le 'SIMULINK' nous permet aussi de modifier facilement les conditions atmosphé-

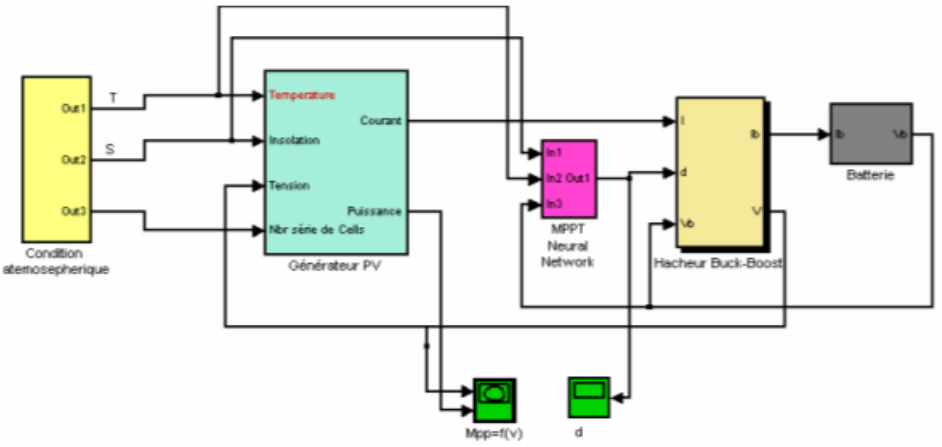

FIG. 5.1 – Structure du système de commande par réseau de neurones

riques (ensoleillement, température) afin d'évaluer la trajectoire de tracking du MPP des différentes techniques MPPT vis-à-vis des changements brusques ou lents de ces conditions. Il nous permet aussi de calculer les différents paramètres caractéristiques du système (efficacité, taux d'ondulation...).
L'insertion du contrôleur neuronal précédent dans un schéma de commande approprié (figure.5.1) permet de donner les signaux de puissance, de commande et la tension du module (figure.5.2) pour les conditions standard de fonctionnement.
L'efficacité de la commande neuronale réside dans le fait que le réseau de neu-

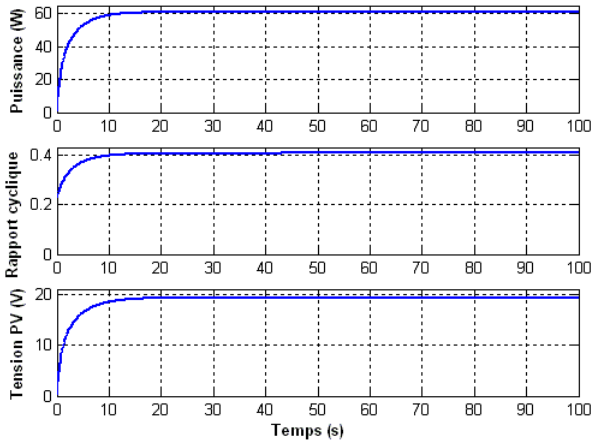

FIG. 5.2 – Forme des signaux de puissance, du rapport cyclique et de la tension du module PV, réalisé par le contrôleur neuronal combiné avec un hacheur Buck-Boost à T= 25 °C et S=1000W/ m^2.

rones n'a pas besoin d'un certain nombre d'itérations sur l'algorithme pour évaluer la sortie de commande. Ainsi, au niveau des commandes classiques, l'évaluation de la commande à la sortie du réseau de neurones ne dépend que du temps de propagation de l'information de ses entrées à sa sortie, et donc on peut augmenter la fréquence d'échantillonnage du calculateur autant qu'on le souhaite sans incidence, pour améliorer sa rapidité.

5.2.1 Performances des commandes sous des conditions différentes

Diminution d'une insolation

Dans ce qui suit, nous allons tester la réponse des différentes commandes, pour un changement d'insolation de 1000 W/ m^2 à 500 W/ m^2, et cela dans le but de confirmer toutes les performances éventuelles que présente cette commande.
Le résultat de simulations obtenu sur la figure.5.3 sont issus des considérations prises au niveau de l'ensoleillement où une diminution de 1000 W/ m^2 à 500 W/ m^2 est effectuée, tandis que la température elle est maintenue constante sur tout l'intervalle de simulation à 25 °C.
L'observation de la courbe de rendement de puissance, ainsi que de la sortie com-

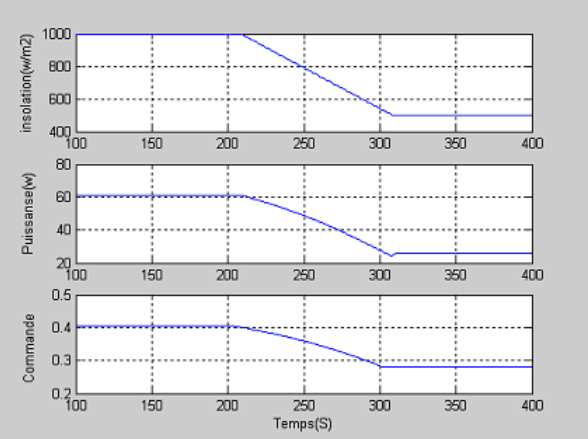

FIG. 5.3 – Signaux de puissance et de commande obtenus par l'application d'un contrôleur neuronal, combiné avec un hacheur Buck Boost,sous une variation lente d'insolation

mande de contrôleur, pendant la variation d'insolation, montre une convergence de la commande, Cette convergence dans la variable de commande bascule le point de fonctionnement plus proche du PPM réel (optimal), ce qui a un effet excellent sur le rendement en puissance. Cela se voit clairement sur la forme du signal de puissance (il n'y a pas de concavité pendant l'intervalle de variation de l'insolation). De plus, la variable de commande ne se manifeste pas des oscillations.

Augmentation de température

Il est très important de tester la performance de la commande, vis à vis les variations éventuelles en température. Elle est considérée aussi comme une variable d'état dont la puissance du système PV dépend fortement. Le paramètre d'insolation est maintenu constant à S=1000 W/ m^2 pour la commande et durant toute la période de simulation.
La température augmente de 10 °C à 60 °C au cours d'une période de 60 secondes (figure.5.4).
L'allure montre que le MPPT neuronale réagit avec finesse en évitant toute oscillation possible.

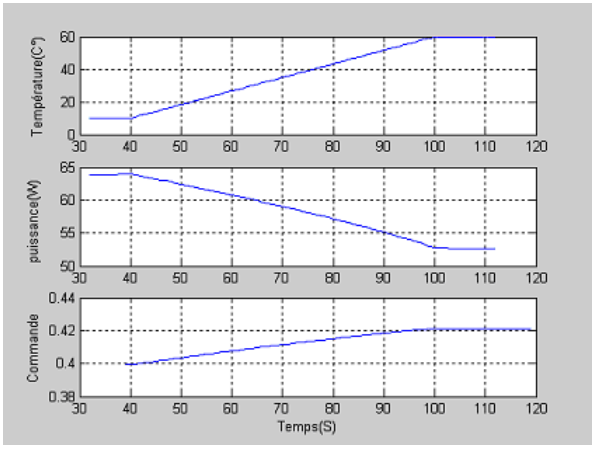

FIG. 5.4 – Signaux de puissance et de commande obtenus par l'application du contrôleur neuronal, combiné avec un hacheur Buck-boost, sous une variation lente température

Variation aléatoire (lente et rapide) de la température et de l'insolation

La performance que présente la commande neuronale réside d'une part dans sa rapidité à estimer la position du PPM sans oscillations, et d'autre part,dans l'obtention d'une puissance maximale dépourvue de toute oscillation. Cela est confirmé (figure.5.5).

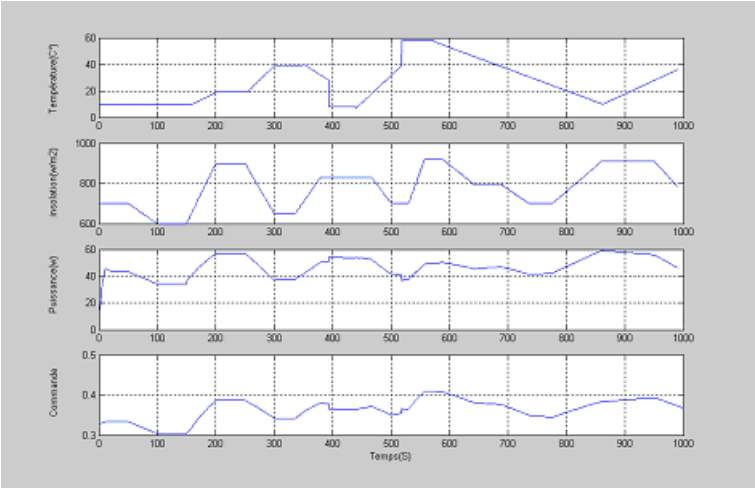

FIG. 5.5 – Signaux de puissances et de commandes générés par la commande neuronale combinée avec un hacheur Buck-Boost, avec une variation aléatoire de la température et de l'ensoleillement

5.3 Développement sur FPGA

5.3.1 Introduction

Dans cette partie nous présentons la structure générale des programmes écrits en VHDL ainsi que les différentes étapes à suivre pour l'implémention des programmes du contrôleur sur le circuit FPGA. Enfin nous donnons les résultats de simulation et de test obtenus pour des conditions de fonctionnement différentes,avec les differentes observations et conclusions que l'on peut tirer à partir de ces résultats.

5.3.2 Description des programmes écrits en VHDL

nous faison ici une description générale des programmes quenous avons écrits en VHDL,nous les illustrons par des schémas simplifiés et nous exlpiquons le rôle de chaque bloc dans le programme et la relation existante entre les différents blocs. Notons juste que l'on a utilisé un codage binaire des entrées sur 10 bits pour S (l'ensoleillement), 7 bits pour V (la tension dynamique de la batterie), et 7 bits pour T (la température) en supposant que l'on dispose d'un convertisseur analogique-numerique pour chaque capteur.

Contrôleur-neuronale

La structure de base du contrôleur-neuronale a été présentée dans le chapitre 2. le Programme est écrit en VHDL conformément á cette structure en divisant en quatre blocs principaux comme suit (figure.5.6) :

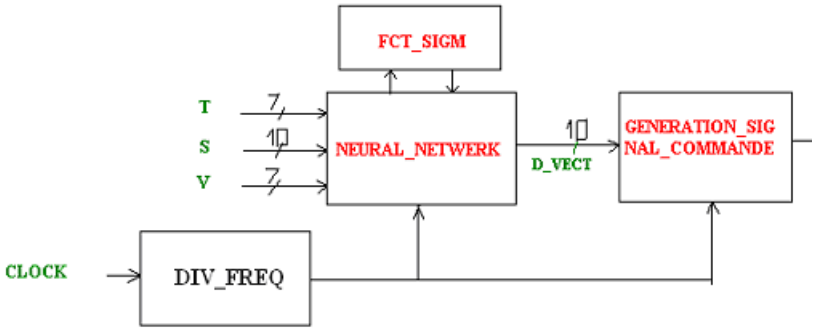

FIG. 5.6 – Schéma synoptique détaillé du bloc (MPPT neuronale)

- DIV-FREQ : diviseur de fréquence pour la synchronistion du système à des fréquences convenables.
- neuronale-NETWORK : il se compose de l'algorithme du réseau de neurone avec :
  - la normalisation de chaque entrée ,
  - les couches de réseau qui propagent les signaux d'entrées vers la sortie pour calculer le rapport cyclique ,
  - Le codage de la fonction sigmoïde dans le bloc (FCT-SIGM).
- GENERATION-SIGNAL-COMMANDE : il génére le signal de commande rapport cyclilque d, qui peut prendre les valeurs 1 pou t_{on} ou 0 pour t_{off} avec la période T désirée ($t_{on} + t_{off} = T$).

Ainsi, le programme peut être représenté par une bôite noire (figure.5.7).

FIG. 5.7 – Schéma synoptique simplifie du bloc (MPPT neuronale)

5.3.3 Synthèse

Pendant l'étape de synthèse, le synthétiseur vérifie les erreurs de syntaxe et de programmation ; ensuite il convertit le programme VHDL en portes logique et bascules, c'est-à-dire en éléments électroniques de base. L'outil (View RTL Schematic) permet de visualiser les schémas équivalents générés par le synthétiseur pour chaque bloc du programme ainsi que la relation entre les différents blocs dans le programme principal.
Les figures.5.8,9,10 montrent les schémas équivalents de chaque du bloc VHDL du contrôleur neuronal généré par le synthétiseur.

De plus ,le synthétiseur permet à l'utilisateur d'imposer des contraintes de tech-

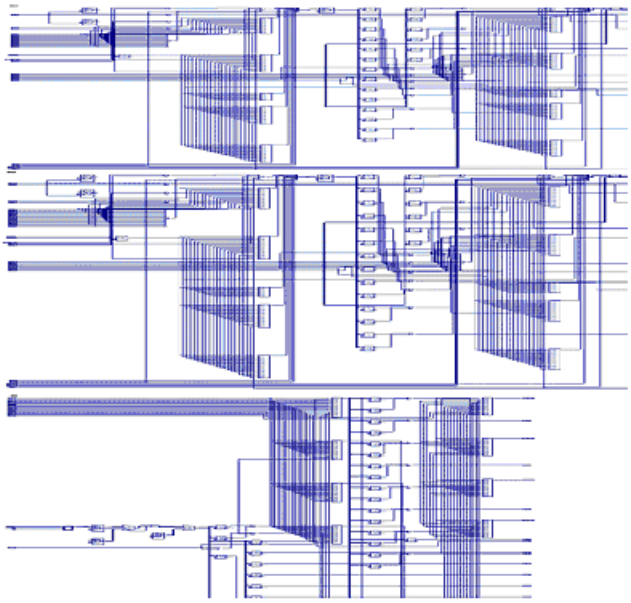

FIG. 5.8 – Schéma équivalente du bloc NEURALNETWERK (avec la fonction sigmoïde)

nologie (User constraints) : par exemple fixer la vitesse de fonctionnement (Create timing Constraints), délimiter la zone du circuit FPGA dans laquelle le routage doit se faire (Create Area constraints) ou affecter les broches d'entrées/sorties (Assign Package Pins). La figure 5.11 montre un aperçu de l'outil d'assignation des broches d'entrées/sorties.

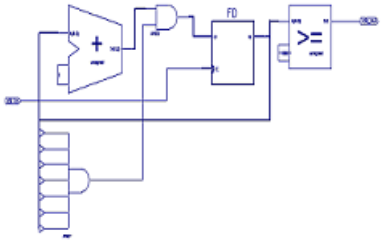

FIG. 5.9 – bloc diviseur de fréquence

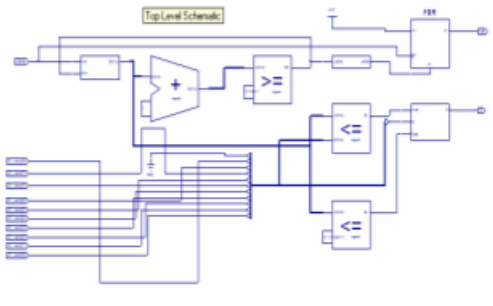

FIG. 5.10 – bloc génération signal de sortie

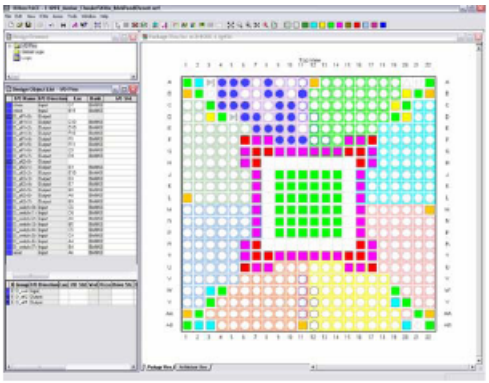

FIG. 5.11 – Aperçu de l'outil d'affectation des branches d'entées /sorties

5.3.4 Simulation

La simulation nous permet de valider théoriquement les programmes écrits en VHDL avant la programation du circuit FPGA.
Pour une meilleur validation et afin de tester l'efficacité des programmes on établir deux model de simulation :

Détermination point de puissance maximale

Nous avons fait entrer des valeurs de T, S, V de la base de donée et etudier le comportement du rapport cyclique qui set un signal électrique qui peut prendre les valeurs 1 pour t_{on} ou 0 pour t_{off},et vérifié sa valeur avec la valeur de la base de donnée(figure.5.12) . Cette simulation nous a permis de :

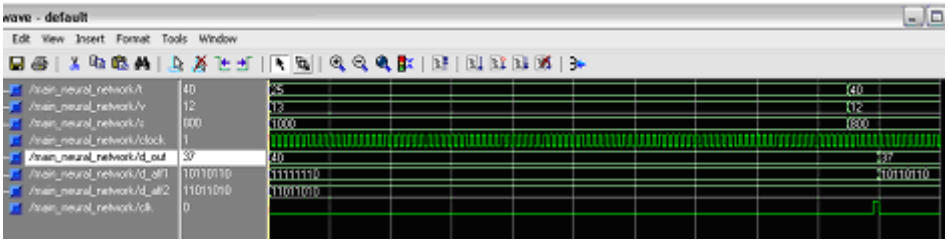

FIG. 5.12 – Simulation de la détermination du point MPP par le contrôleur neuronale

- Valider le fonctionnement du programme et du contrôleur.
- Evaluer la précision de la commande.
- confirmer que notre système est synchronise à la fréquence de travaille (la fréquence fixée par le bloc (div-fréq) diviseur de fréquence) qui peut être changé selon les besoins de l'application.

On remarque que la réponse du contrôleur à des variations des entrées est instantanée (avec le front montant de signal clock du bloc (div-fréq)) et ne nécessite pas de temps de calcul (en réalité le temps de réponse sera le temps de propagation des signaux, c'est-à-dire le retard des éléments électroniques).

Poursuite du point de puissance maximale

Dans cette étape, on observe la réponse du contrôleur lors de changements brusques des conditions météorologique ,et cela en varient toutes les valeurs d'entrées la base de donnée [1](figure.5.13). Cette simulation nous a permis de :

- tester la réaction et la poursuite du point de puissance maximale (PPM) pour le contrôleur pour des variations brusques des conditions de fonctionnement.
- tester l'apprentissage du réseaux de neurones suite à la variationde des entrées T, S, V utilisées lors l'apprentissage de réseau de neurones(cette simulation est similaire à la simulation de MATLAB déjà vue au chapitre 2, et donne les mêmes résultats (figure.2.12,13)).

[1]Nous avons apporté quelques modifications aux programmes au niveau des entrées pour qu'on puisse balayer toute la base de données, et au niveau de la sortie pour données des valeurs de la sortie d comme des valeurs réelles et non comme des signaux.

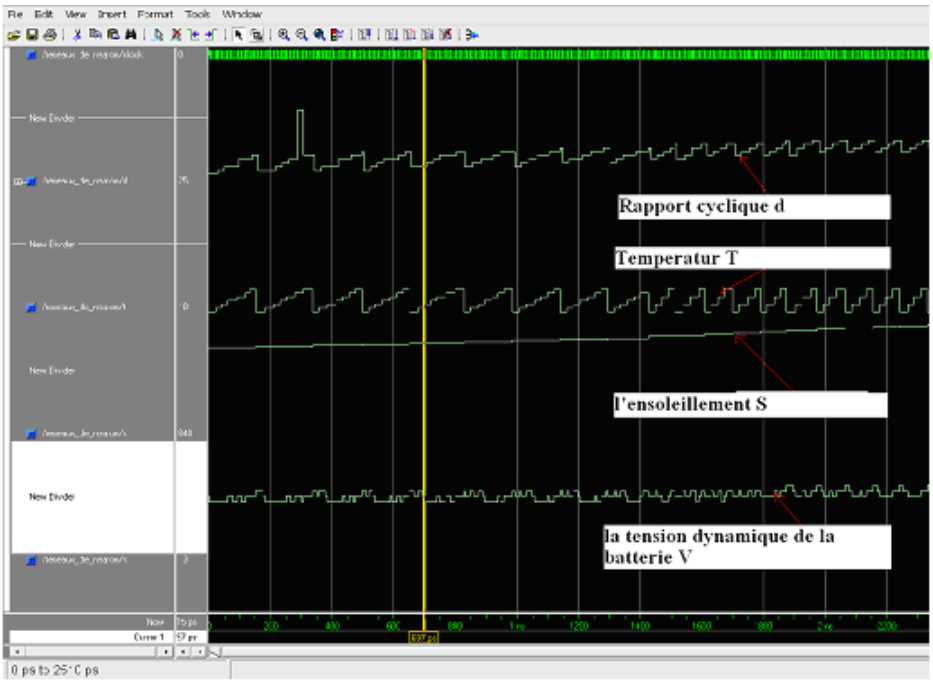

FIG. 5.13 – simulation de la poursuite MPP par le contrôleur neuronale, pour des changements brusques des conditions météorologiques

5.3.5 Environnement de test

Une fois les programmes chargés sur la carte, il faut effectuer des tests réels pour les valider pratiquement. Afin de teste le bon fonctionnement de programme écrit en VHDL sur la carte de développement Vertex II, on a conçu un environnement de test (figure. 5.14) qui exploite les ressources offertes par la carte de développement et permet de visualiser les résultats en temps réel. Cet environnement de test est constitué de plusieurs blocs . le bloc User DIP Switch permet de sélectionner dif-

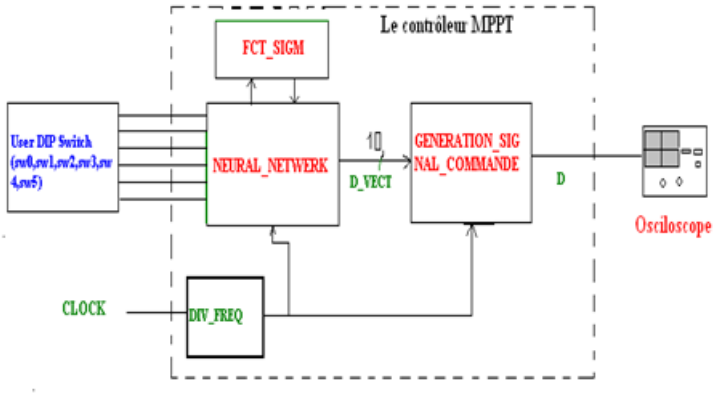

FIG. 5.14 – environnement de teste de la carte

férentes valeurs entrées à partir des 8 Switch (User DIP Switch) disponibles sur la carte Memec Dsign qui peuvent être statiquement mise à 0 ou à 1.
Les résultats sera visualise sur un oscilloscope numérique qui donne les différents paramètre du signal rapport cyclique (la fréquence, le rapport cyclique, le temps de montée, la période, etc.)
Les figure 5.15, 16, 17 donne quelque résultats des tests effectués dans le laboratoire

pour des valeurs prés de la base de données et, et nous comparent la valeur de d calculée avec la valeur de d désirée.
Ces teste donnes des bonnes résultats puisque les deux valeurs presque égaux.

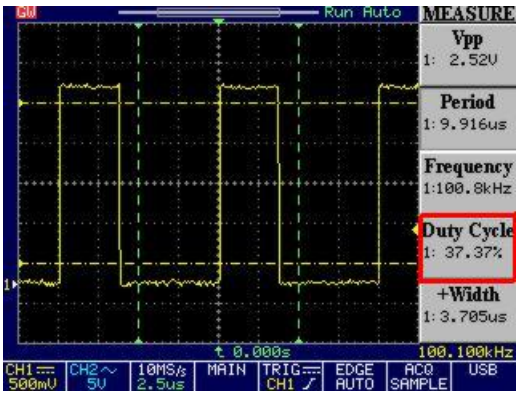

FIG. 5.15 – Pour T=30 °,S=800W/ m^2,V=11.8v, $d^{des} = 37\%$,$d^{cal} = 37.37\%$

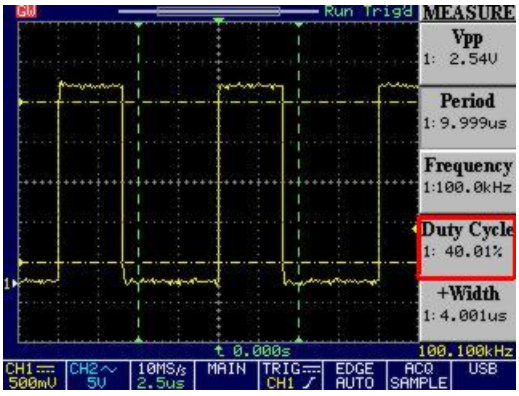

FIG. 5.16 – Pour T=60 °,S=930W/ m^2,V=12v, $d^{des} = 40.5\%$,$d^{cal} = 40.01\%$

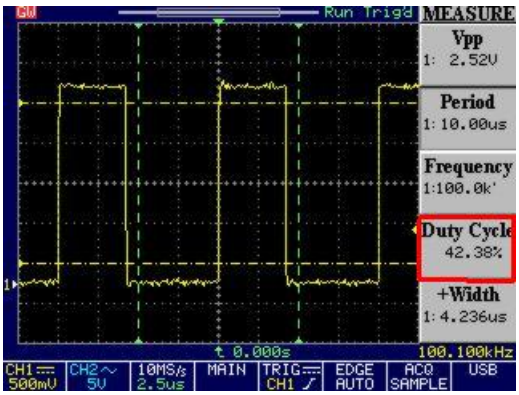

FIG. 5.17 – Pour T=60 °,S=1000W/ m^2,V=12.92v, $d^{des} = 42.5\%$,$d^{cal} = 42.38\%$

5.3.6 Placement et routage du programme sur le circuit FPGA

Dans cette étape, l'outil de placement et de routage trace les routes sur le circuit FPGA qui accomplit le fonctionnement attendu. L'outil FPGA Editor nous permet

de visualiser le routage réaliser sur le circuit FPGA. La figure suivante montre le routage du contrôleur neuronalE(figure.5.18). On remarque que le contrôleur prend

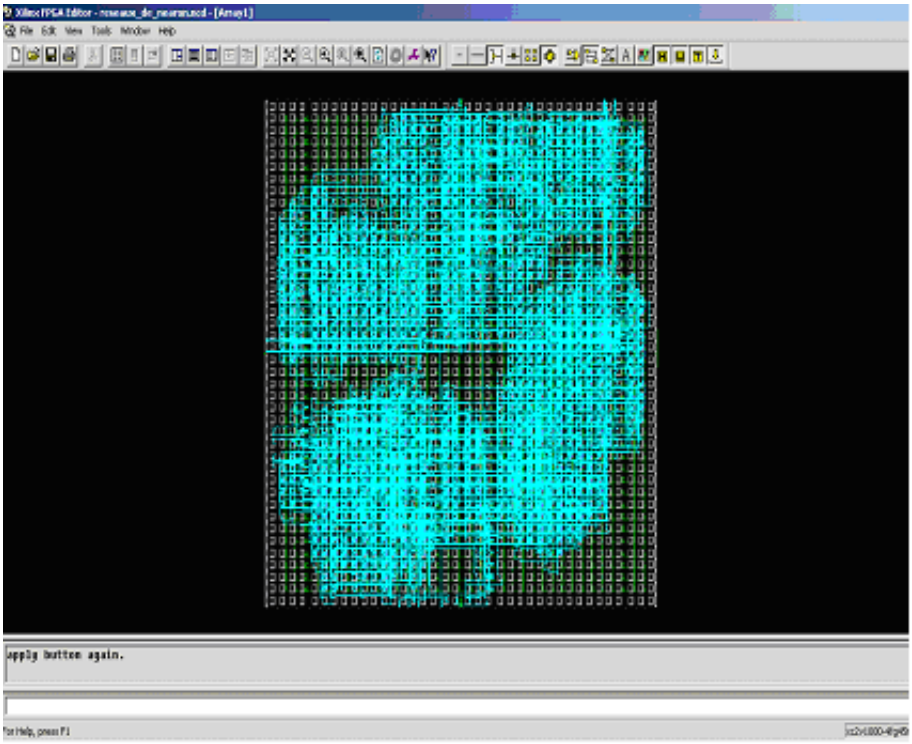

FIG. 5.18 – Routage du circuit FPGA pour le programme MPPT neuronal

un peut espace dans le circuit FPGA à cause de la fonction sigmoïde représenté par une fonction escalier avec un pas très petit pour augmenter la précision,et qui sera utilise deux fois dans le programme .
Nous obtenons informations suivants sur les ressources utilisées par la carte vertex II : 2v1000fg456-4 : Number of Slices : 3145 de 5120 61%
Number of Slice Flip Flops : 33 de 10240 0%
Number of 4 input LUTs : 5862 de 10240 57%
Number of IOs : 26
Number of bonded IOBs : 26 de 324 8%
IOB Flip Flops : 11 Number of MULT18X18s : 6 de 40 15%
Number of GCLKs : 1 de 16 6%

5.3.7 Conclusion

A partir des résultats obtenus, on peut dire que la méthode MPPT implémentée sur circuit FPGA a ses avantages et ses inconvénients.
L'avantage de la méthode neuronale est la rapidité de détermination du rapport cyclique.
Son inconvénient est le codage de la fonction sigmoïde qui en diminue la précision (augmentantml le risque d'erreurs).

5.4 Développement sur DSP

5.4.1 Introduction

Nous utiliseons ici toutes les notions déjà vues dans les chapitres (1,2 et 4) pour effectuer des liaisons entre les différentes composantes régissant les exigences de notre application.
Nous présenterons en détails les différentes étapes du développement.

5.4.2 Outils de développement

TI offre une ligne étendue d'outils de développement pour le TMS320C6000, y comprenant des outils pour évaluer l'exécution des processeurs, la génération de code et le développement d'algorithmes d'implémentation, ainsi que la gestion du software et du hardware.
Les produits suivants soutiennent le développement d'application sur le TMS320C6000
.

- Pour le développement Software : on a le Code Composer Studio, le compilateur C/C++/ et l'Assembleur, ainsi que le Real Time Data eXchange (RTDX) et DSP/BIOS pour le transfert de données entre un ordinateur hôte et les périphériques du DSP.
- Pour le développement Hardware : On trouvera l'émulateur XDS (Extended Development System) et EVM (Evaluation Module)

Code composer studio

Code Composer Studio (CCS) est un environnement intégré de développement de code pour les DSP de Texas Instrument. Il est fourni en standard avec la carte de développement pour le DSP.
CCS fournit plusieurs outils pour faciliter la construction et la mise au point des programmes de DSP. Il comprend un éditeur de code source, un compilateur de langage C/C++, un assembleur de code source, un éditeur de liens et un environnement d'exécution qui permet de télécharger un programme exécutable sur une carte cible, de l'exécuter et de le déboguer au besoin. CCS comprend aussi des outils qui permettent l'analyse en temps réel d'un programme en cours d'exécution et des résultats produits.
Finalement, il fournit un environnement de gestion de fichiers qui facilite la construction et la mise au point des programmes.

Outils de génération du code

Le Code Composer Studio offre une interface graphique adéquate pour l'utilisation des outils de génération du code. Il garde en permanence toutes les informations ou les fichiers utilisés pour la création ou la compilation d'un programme ou d'une librairie. La figure suivante montre le processus du développement software(figure.5.19).

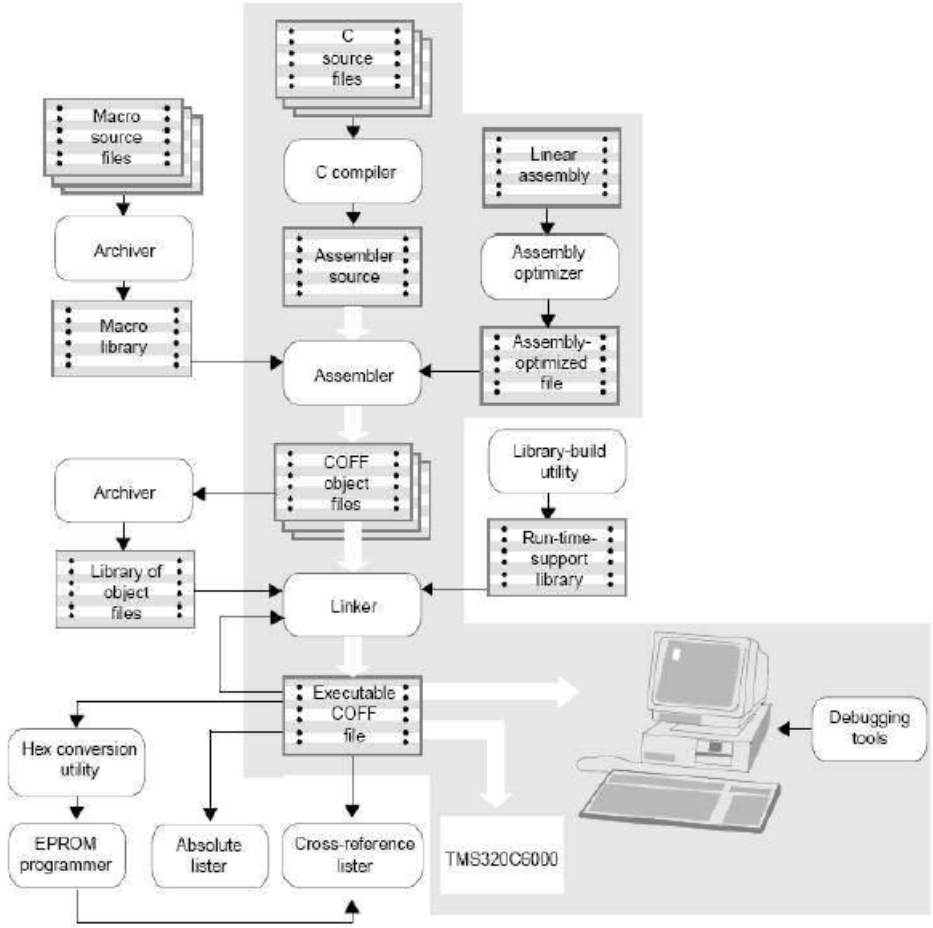

FIG. 5.19 – Processus du développement du code exécutable

5.4.3 Données

Base de données

Notre vecteur a été pris dans la base de données utilisant Matlab sous une format (.Dat),constuit de trois colonnes (T,V et S).
Ensuite on sauvegarde les vecteurs, colonne qui dépendent notre programme, dans un fichier notepad avec l'entête (1651 4 0x8000FE40 1 0x0000040E). Ce fichier sera par la suite enregistré sous le nom (TVS.dat).
Un nouveau fichier DAT est ouvert et il représentera notre base de données.

5.4.4 La simulation

Pour cette partie, on a installé le code composer studio (C6000), configuré le CCS au niveau du CCS setup . On le met en simulation, comme l'indique la figure.5.20.

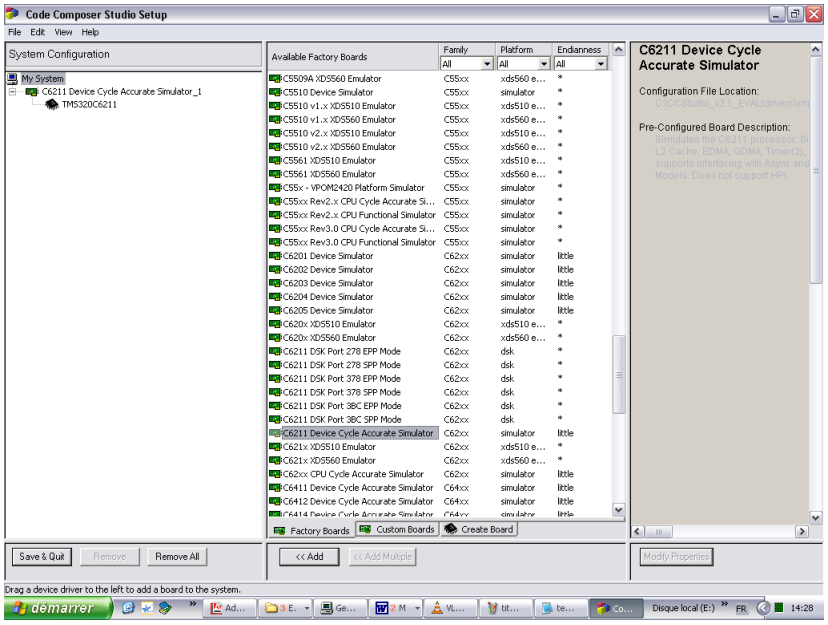

FIG. 5.20 – configuration du CCS setup

Programs →Texas Instruments →Code Composer Studio →Code Composer Studio

A partir du ce lien en lance CCS (figure.5.21)

5.4.5 Création de projet

Au niveau du menu projet, on choisit new voir (figure.5.22)

au niveau du project name

On tape le nom de notre projet (exemple : MPPT NEURAL)

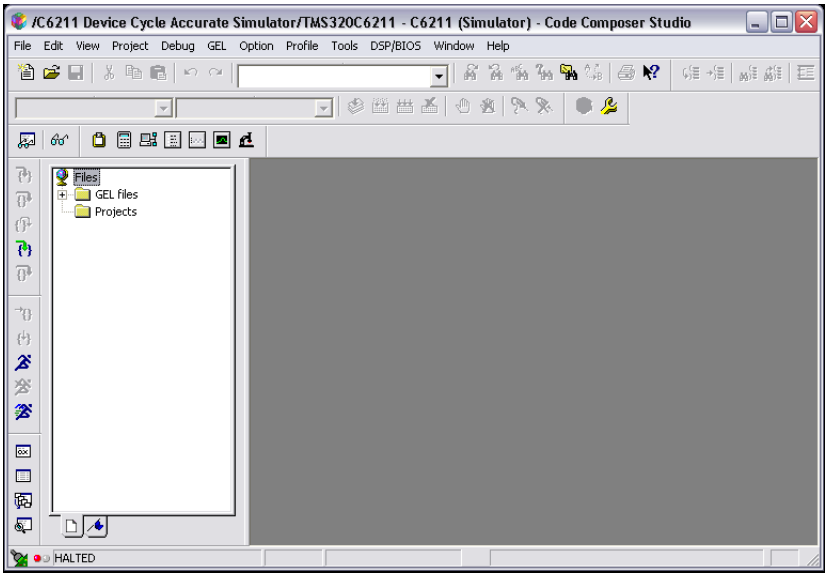

FIG. 5.21 – Fenêtre de base du CCS

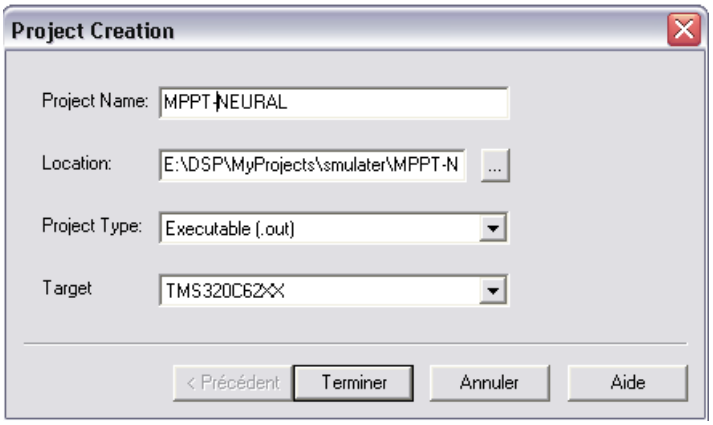

FIG. 5.22 – new project

Project → Add Files(*.c) to Project

On sélectionne (MPPT NEURAL.c) puis on clique sur open.
MPPT NEURAL.c : code source de l'application, on peut le voir comme suit voir (figure.5.23).

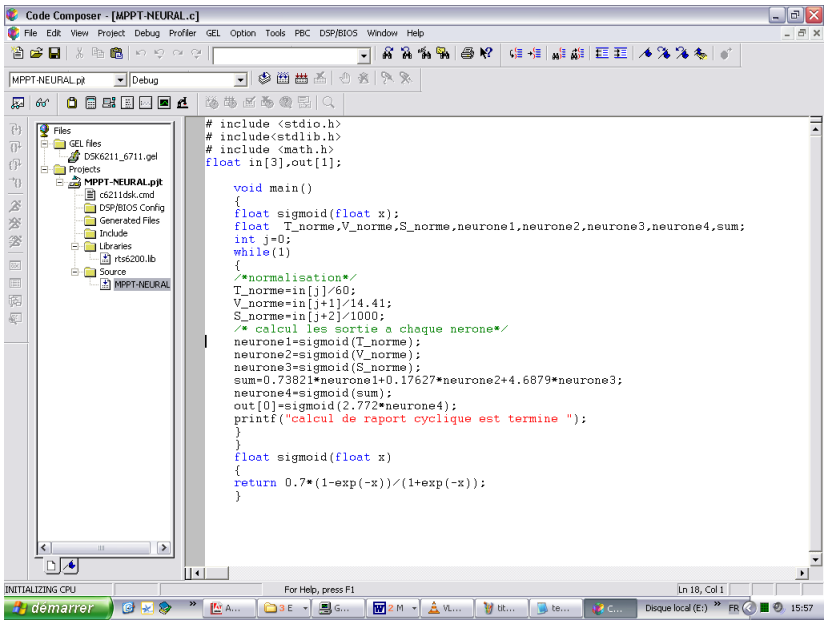

FIG. 5.23 – Add files

Project →Add Files(*.cmd,*.lcf) to Project

On sélectionne Linker Command File (*.cmd,*.lcf) dans le fichier de type.
Ce fichier peut être vu comme (the memory map) . Des blocs de code et de données peuvent être répartis dans la mémoire sous forme de sections définies dans ce fichier de commande (.cmd). Ces sections peuvent être initialisées ou non initialisées. Les sections initialisées ou non initialisées, excepté .text, ne peuvent pas être définies dans la mémoire interne de programme(voir chapitre 4)

Project →Add Files(*.o, *.l) to Project

On va sur (C ti c6000 cgtools lib) et on sélectionne Object and Library Files (*.o, *.l) dans le fichier de type. Un certain nombre de bibliothèques s'affichera et on choisit la bibliothèque rts6200 voir (figure.5.24).
rts6200.lib : librairie standard pour prendre en charge les fonctionnalités du processeur.

Project →Build Options →Compiler

On configure target version on section C621x voir (figure.5.25).

Project →Build Options →Linker →Heap size (.heap)

et on met 0x400 dans la case voir (figure.5.26). heap size : c'est la taille de la mémoire allouée pour les variables locales.

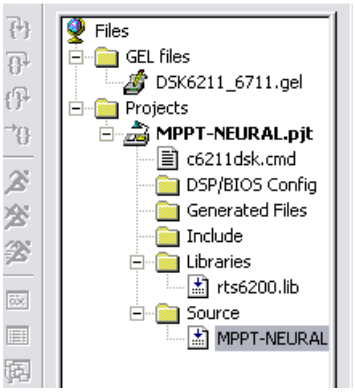

FIG. 5.24 – Add files

Project →Build Options →Linker →Stack size (.stack)

et on met 0x400 dans la case voir (figure.5.26). stack size :c'est la taille de la mémoire allouée pour la pile qui sert à stocker les adresses mémoires de retour et certaines variables [2].

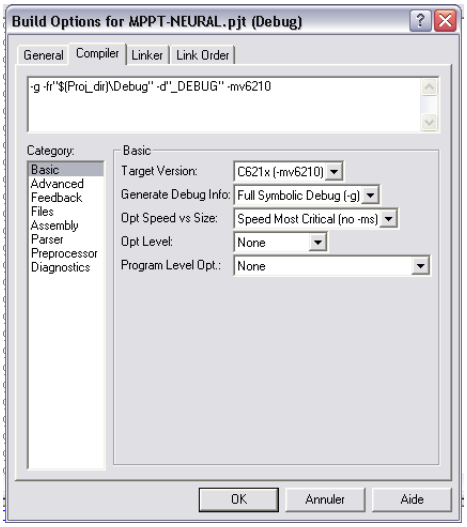

FIG. 5.25 – Target version en C621x

rebuild all

Une fois toutes les étapes précédentes exécutées, on exécute rebuild all. En fin d'exécution de cette commande, un message s'affichera en bas de la fenêtre contenant les erreurs, les warning et les remarques.
Si aucune erreur n'est déclarée, on passe à l'étape suivante, sinon, puison corrige les erreurs et on refait cette étape.

File →Load program

Une boite de dialogue s'affichera ; on sélectionne le fichier exécutable (MPPT-NEURAL.out) en suite ouvrir. Cette étape permet de charger le programme en mémoire voir (figure.5.27).

[2] les vecteurs d'entrée et de sortie sont des variables globales

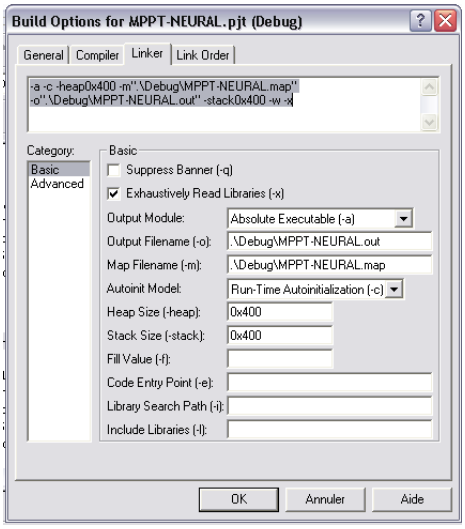

FIG. 5.26 – Stack,Heap size

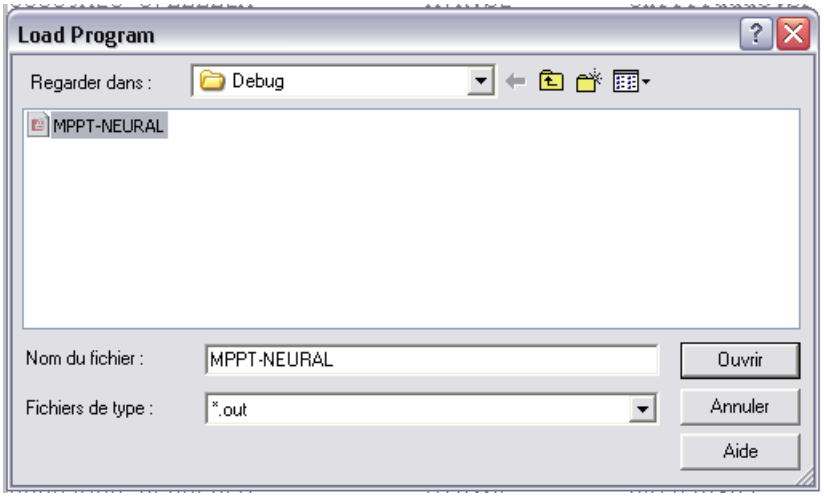

FIG. 5.27 – Load program

File →data →Load

Une boite de dialogue s'affichera ; on sélectionne le type real(*.dat) puis la source de données ,Le vecteur d'entrée (in) sera de taille égale à 3 (T,V et S)(figure.5.28). Ensuite, une boite de dialogue suivante s'affichera. Dans l'espace Adresse, on fait

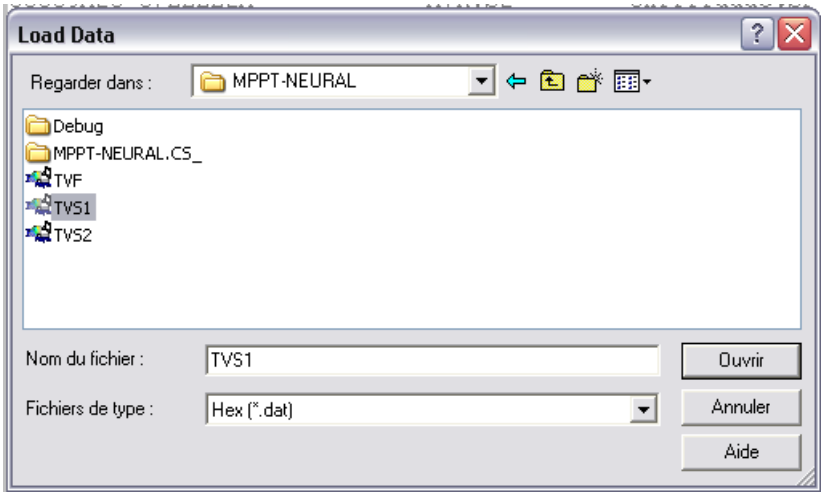

FIG. 5.28 – Load data

FIG. 5.29 – Load data into memory

rentrer (in) et dans l'espace Length en fait rentrer la taille de notre vecteur d'entrée(3),voir (figure.5.29).

Run

View →graph

La boite de dialogue s'affiche : On peut visualiser le vecteur d'entrée, on mettant (in) dans Start Address et (4) dans Lenght.

View →graph

La même boite de dialogue s'affichera : On peut visualiser le vecteur de sortie, en mettant (d) dans Start Address et (240) dans Lenght.

5.4.6 Résultats obtenus

Ces résultat sont obtenus pour (T=40, V=12 et S=800) le qui cyclique d=0.385, et le nombre n=240(vecteur de temps) de telle sorte à avoir la période T= 30ms [3].

[3] T=n*0.125ms pour la démonstration voir[25]

voir (figure.5.30,31)

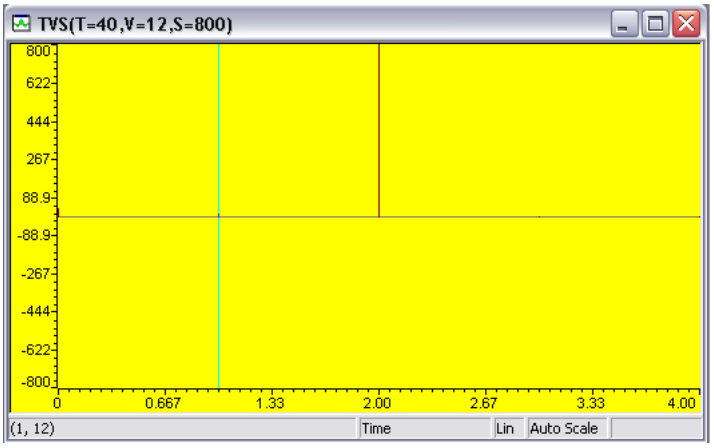

FIG. 5.30 – vecteur d'entrée (T,V et S)

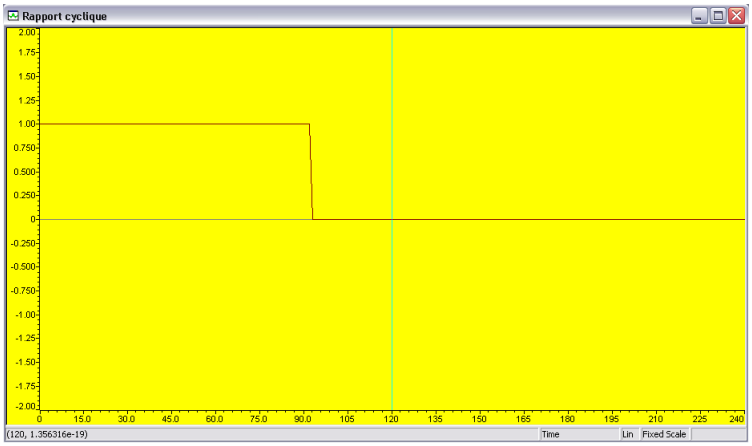

FIG. 5.31 – vecteur de sortie rapport cyclique

5.4.7 Poursuite du point de puissance maximale

Dans cette étape, on observe la réponse du contrôleur lors des changements brusques des conditions météorologiques en propageant à l'entrée toutes les valeurs de la base de donnée T,V et S voir les (figures.5.32,33,34),On aurait alors le rapport cyclique voir (figure.5.35).

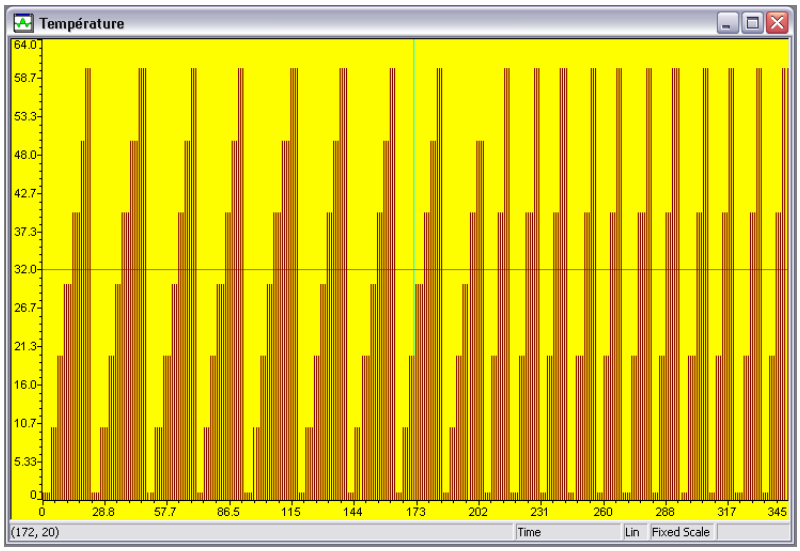

FIG. 5.32 – Température T

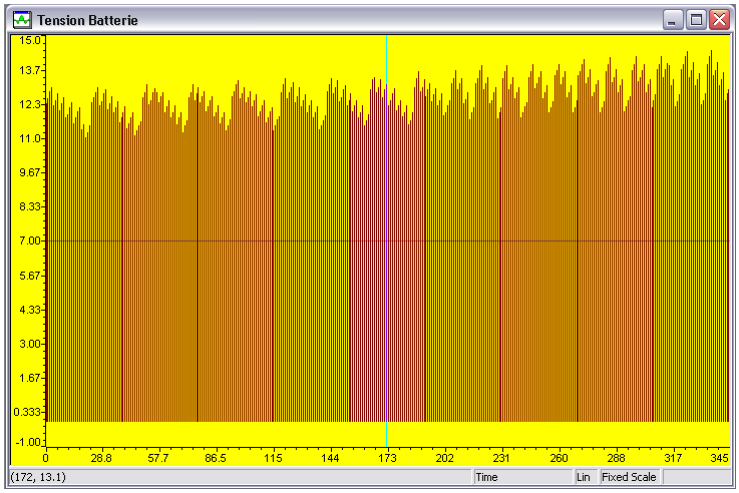

FIG. 5.33 – Tension Batterie V

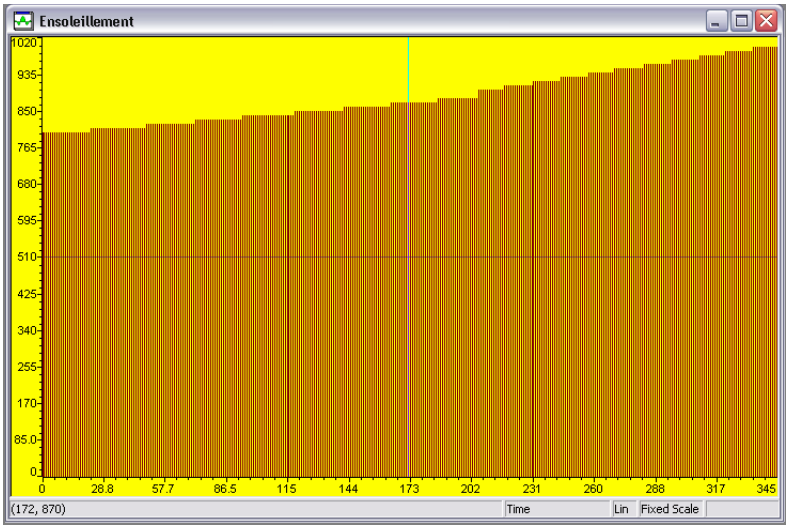

FIG. 5.34 – Ensoleillement S

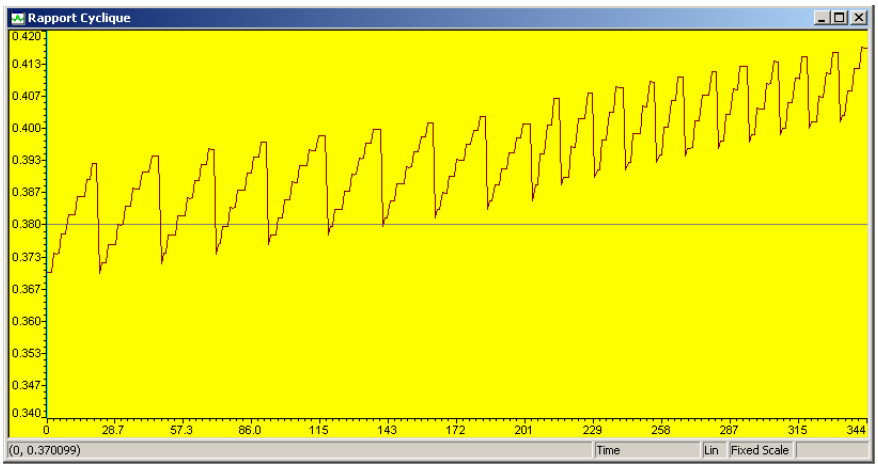

FIG. 5.35 – Rapport Cyclique d

5.4.8 Conculsion

Nous avons présenté les outils de développement de notre application. L'environnement de développement intégré (Code Composer Studio) de TI qui sera un moyen de simulation et d'implémentation,afin d'établir la communication avec la cible (DSK) et visualiser le rapport cyclique traité.
On peut dire que la méthode MPPT implémentée sur circuit DSP a ses avantages et ses inconvénients.
L'avantage de la méthode neuronale est sa facilité de programmation en raison d'existence d'un compilateur C et d'une bibliothèque math qui augmente la précision de la fonction sigmoïde.
Le multiplexage d'entrée des vecteurs T, V et S qui peut engendrer un conflit est l'inconvénient de cette méthode.

Conculsion générale

Ce travail avait comme objectif primordial l'amélioration du rendement d'un système d'alimentation photovoltaïque. Grâce à la présence d'un processus de poursuite du point de puissance maximale (Contrôleur MPPT neuronale), pour un but adaptation source/charge permettant un transfert maximum de puissance, on a implémenter l'algorithme sur DSP et FPGA.

Dans un premier temps, nous avons élaboré en détail la synthèse du modèle mathématique de chaque organe constituant le système photovoltaïque, à savoir les cellules PV formant un module, l'élément de stockage, les hacheurs, ainsi que, le principe de MPPT . Pour cela, nous nous sommes attachés surtout à améliorée et étudie complètement le contrôleur MPPT neuronale.

Les simulations effectuées par l'utilisation de cette technique de commande sous différentes conditions atmosphériques nous ont permis de comprendre les problèmes liés au fonctionnement de la méthode MPPT neuronale afin de localiser les inconvénients et les avantages de cette méthode et essayer de trouver des améliorations du point de vue efficacité et de la complexité de réalisation.

On s'est ensuite intéressé aux circuit FPGA, et on a détaillé les différentes étapes de la procédure de développement d'un projet sur circuit FPGA, de l'écriture du texte VHDL, en passant par la synthèse et la simulation jusqu'au placement et au routage, et enfin a la programmation du composant et teste.

La simulation nous a permis de valider l'étude théorique avant l'implémentation sur le circuit FPGA. Elle nous a également aidés à tester les performances de la méthode pour différentes conditions de travail. Grâce à elle, on a pu déterminer les avantages et les inconvénients de cette méthode.

Enfin nous nous sommes intéressé à implémenter ce contrôleur sur un DSP de la famille C6000 de TI, à savoir la cible C6211 mise en oeuvre sur une carte DSK, à l'aide du logiciel Code Composer Studio.

Grâce à cette recherche , nous avons pu acquérir des connaissances précieuses dans une branche très intéressante de l'électronique que nous n'avions pas eu l'occasion d'étudier(réseaux de neurones,Matlab ,Simulink,FPGA,VHDL, DSk,CCS), ainsi qu'un bon contact avec la pratique et la théorie.

Bibliographie

Bibliographie

Photovoltaïque

[1] TOM MARKVAT et LUIS CASTAFIE, photovoltaic fundamentals ,Elsevier,2003.
[2] MUKUND R PATEL,Wind and Solar Power Systems, CRC Press,1999.
[3] ANTONIO LUQUE et STEVEN HEGEDUS, Handbook of Photovoltaic Science and Engineering,Wiley,2003.

Réseaux Neuronaux

[4] LAURENE FAUSETT ,Fundamentals of neural networks,x.
[5] TEUVO KOHONEN,Self Organizing,Spring,2001.
[6] MICHAEL A.ARBIB,Handbook Of Brain Theory And Neural Networks , MIT Press 2003 ,Part 1,2.
[7] BEN KROSE et PATRICK VAN DER SMAG, Mathematics - An Introduction to Neural Networks,Eighth edition,1996.
[8]JAMES A. FREEMAN et DAVID M. SkAPURA ,Neural Networks Algorithms, Applications, and Programming Techniques, SERIES,1991.
[9] PETTER DAYAN et L.F. ABBOTT, Theoretical Neuroscience,Drafet,2000.
[10] MADAM M. GUPTA, LIANG JIN, et NORIYASU HOMMA , Static and Dynamic Neural Networks,wiley,2003.
[11] CLAUDE TOUZET, Les Réseaux De Neurones Artificiels,1992.

FPGA

[12] AMOS R. OMONDI et JAGATH C. RAJAPAKSE ,FPGA Implementation Of Neural Networks ,Springer,2006.
[13] Clive "Max" Maxfield ,The Design Warrior's Guide To FPGA,Elsevier and Newnes,2004.
[14] JEAN-PIERRE DESCHAMPS et GE´RY JEAN ANTOINE BIOUL et GUSTAVO D. SUTTER , Synthesis Of Arithmetic Circuits FPGA, ASIC, and Embedded Systems, Willey,2006.
[15] RICHARD MUNDEN, ASIC And FPGA Verification A Guide To Component Modeling, Morgan Kaufmann,2005.

VHDL

[16] DELMAR THOMSON, Digital Design with CPLD Applications & VHDL,Online,2000.
[17] STEPHEN BROWN et ZVONkO VRANESIC , Fundamentals Of Digital Logic With Vhdl, McGraw Hill,2005.
[18] ENOCH O. HWANG, Brooks-Microprocessor Design with VHDL, Brooks / Cole, 2004.
[19] UWE MEYER BAESE, Digital Signal Processing With Fpga,Willey,2001.
[20] JACQUES WEBER et MAURICE MEAUDRES, Le Langage Vhdl Cours Et Exercices, Dunod ,1997.
[21] DOUGLAS L PERRY, VHDL Programming by Example, McGraw Hill,2002.
[22] VOLNEI A PEDRONIolnei A. Pedroni, Circuit Design With Vhdl,Mit Press ,2005.
[23] DAVID PELLERIN et DOUGLAS TAYLOR , VHDL Made Easy ,Prentice HALL,1997.

DSP
[24] SEN M KUO et BOB H LEE,Real Time Digital Signal Processing , Wiley , 2001 .
[25] RULPH CHASSING, DSP Applications Using C and the TMS320C6x DSK, Wiley,2002.
[26] PAUL M EMBREE, C Algorithms for Realtime DSP,Prentice Hall,1995.
[27] STEVEN W. SMITH , Digital Signal Processing, California Technical Publishing, 1999.

MATLAB et SIMULINK
[28] ANDREW KNIGHT, Basics Of Matlab,CRC, 1999.
[29] BRAIN R. HUNT , RONALD L. LIPSMAN et JONATHAN M. ROSENBERG, A Guide to Matlab,Cambridge,2001.
[30] STEVEN T. KARRIS, Introduction to Simulink with Engineering Applications,Orchard Publications,2006.

Mémoires
[31] M. ISSAADI SALIM, Commande d'une Poursuite du Point de Puissance Maximum (MPPT) par les Réseaux de Neurones, Diplôme Magister d'état en électronique ENP, promotion 2006.
[32] BENMOSBAH AMINE et MECHERAOUI CHOUKRI ADEL, Implémentation sur FPGA des Méthodes MPPT : "P&O" et "Floue Optimisée par les Algorithmes Génétiques", Diplôme d'ingénieur d'état en électronique ENP, promotion 2006.
[33] ROULA NADJAH et TIRES SABRINA, Implémentation d'Algorithmes de Traitement d'Images sur DSP C6000. Application à la Séparation des Chromosomes, Diplôme d'ingénieur d'état en électronique ENP, promotion 2006.
[34] BOULFANI YASMINE et DOUMANDJI SAMAH, Implémentation sur DSP TMS320C5000 de Filtres Optimaux Appliqués aux Images et Introduction de Réseaux Neuronaux , Diplôme d'ingénieur d'état en électronique ENP, promotion 2004
[35] AHMED ZAID SALIM, Implémentation d'Algorithmes de Traitement d'Images sur DSP C6000, Diplôme d'ingénieur d'état en électronique ENP, promotion 2005.

Data Sheet
[37] XILINX, Virtex-II Platform FPGAs : Complete Data Sheet, http://www.xilinx.com, Mars 2005.
[38] TMS320C6000 Peripherals, SPRU190d.pdf, Texas Instruments.
[39] Code Composer Studio Getting Started Guide, SPRU509c.pdf, Texas Intruments.

Annexe

ANNEXE A

.1 Introduction

Les logiciels de simulations sont des outils puissants qui testent et évaluent l'exécution théorique des systèmes. Les conditions d'exécution du dispositif à tester peuvent être facilement contrôlable.
La simulation nous permet de passer de la conception du système théorique à la réalisation pratique avec plus d'assurance car les changements lors de la conception peuvent être faits facilement dans un système simulé, ceci nous permet d'expérimenter un ensemble large de variations et de choisir enfin la solution optimale.
Dans ce chapitre on commence par évalué le système solaire sans régulateur par simulation avec le logiciel de Mathworks Matlab qui inclut l'outil de simulation Simulink, ensuite la méthodes de poursuite MPPT sont étudiée, notre méthode de poursuite utilisant le contrôleur NEURAL.

.2 description du système solaire global

La figure 1 montre le schéma fonctionnel sous Simulink du système solaire générale, ce dernier se compose des éléments suivants :

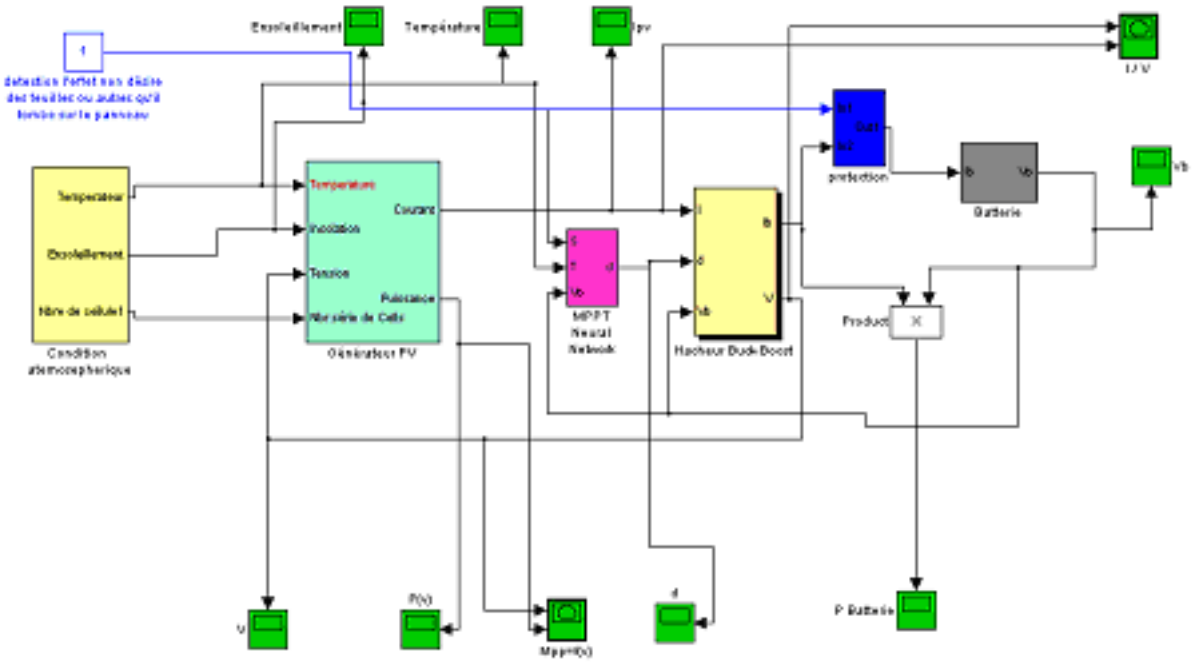

FIG. 1 – Schéma synoptique pour la simulation du système photovoltaïque

.2.1 le panneau solaire

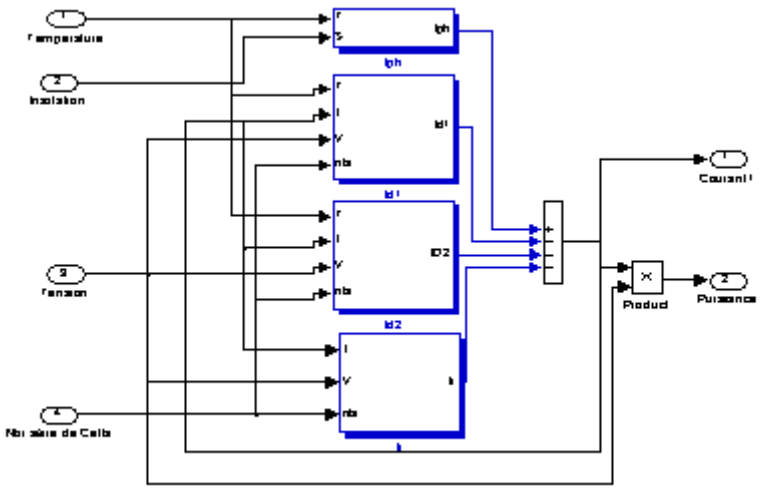

FIG. 2 – Modèle (SIMULINK) du panneau solaire

La figure 2 montre le schéma fonctionnel du panneau solaire sous Simulink , ce schéma modélise les équations mathématique du panneau solaire vu dans le chapitre 1 équation (1.(1,2,3,4,5)) . On donne [31] :
I_{ph}=3.25 A (a T=298K)
R_p=30Ω
R_S=30 10^{-3} Ω
E_g=1.1 eV
n_1=1
n_2=2
K=1.38 10^{-23}J/K
q=1.16 10^{-19}C

.2.2 le modèle de batterie

Si on revient à la chapitre 1 on trouve qu'on a modéliser mathématiquement l'impédance de batterie acide plomb et la fonction de transfert,voir figure 3.
Et c'est le fabriquant qui spécifie les valeurs de $R_{bs}, R_{b1}, R_{bp}, C_{b1}, C_{pb}$ dans notre cas on prend[31]
$R_{bs} = 0.0013\Omega$
$R_{b1} = 2.84\Omega$
$R_{bp} = 1010^3\Omega$
$C_{b1} = 2.5KF$
$C_{bp} = 4.6501KF$

.2.3 Modélisation de l'hacheur

Le modèle mathématique de convertisseur dévolteur-survolteur a été donné au chapitre1.

FIG. 3 – Modèle 'SIMULINK' de la batterie

Les équations de base pour le hacheur dévolteur-survolteur ont été données dans le chapitre 1 équation (1. (7, 8, 9, 10,11)) .
Ces équations sont implémentées en simulink. Le bloc correspondant peut être illustré dans la figure 4 : Dans tous les convertisseurs DC-DC précédents, les valeurs des

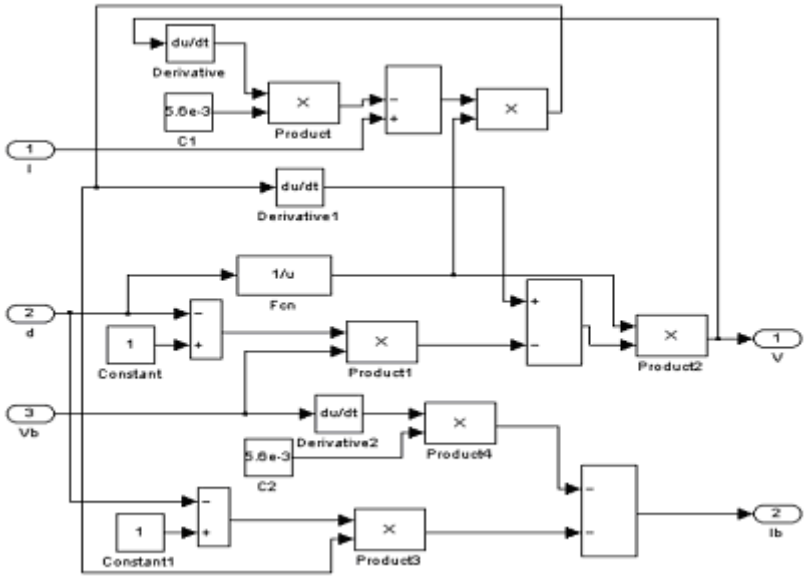

FIG. 4 – Modèle (SIMULINK) du convertisseur Buck-Boost

composants utilisés sont [31] :
$C_1 = 5,6mF$
$C_1 = C_2$
$L = 3.5mH$

.2.4 Contrôleurs MPPT

On va voir dans ce partie Contrôleurs MPPT NEURAL .

Le contrôleur NEURAL

Ce contrôleur intelligent déjà étudié dans le chapitre 2 est modélisé sous Simulink par le bloc qui apparaît dans la figure 5.

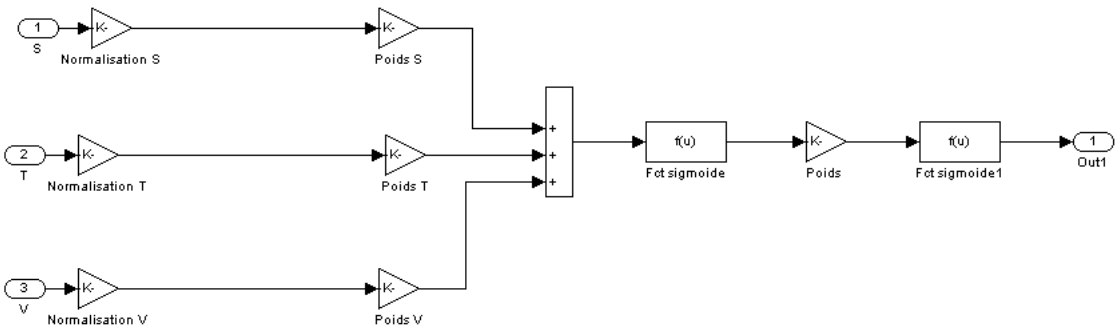

FIG. 5 – Modèle (SIMULINK) de la méthode par NEURAL

.2.5 protection

Pour un but de déconnecte la charge,voir figure 6.

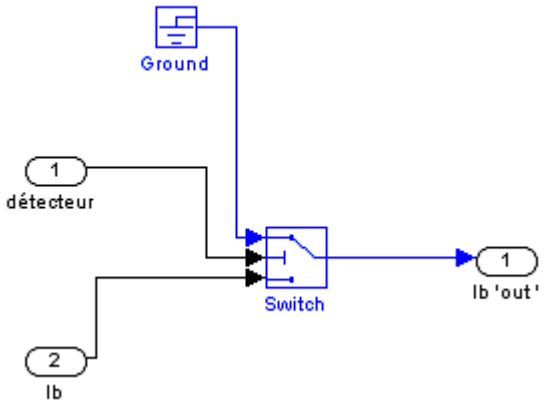

FIG. 6 – Modèle (SIMULINK) de protecteur

ANNEXE B

.1 Etude l'apprentissage en fonction les fonctions d'activations avec et sans condition d'initialisation nughen et window note (Ng-Win)

.1.1 Etude l'apprentissage en fonction les fonctions d'activations sans condition (Ng-Win)

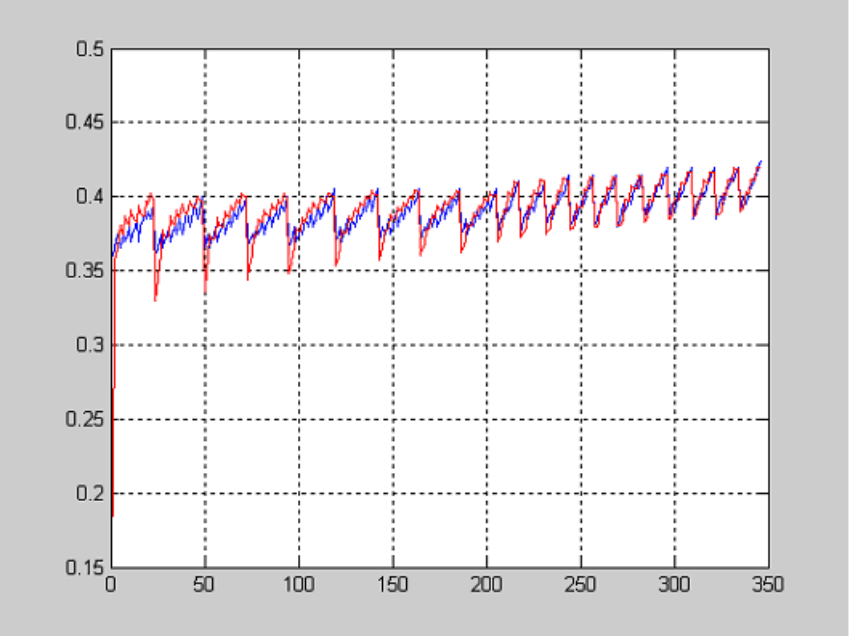

FIG. 1 – avec fonction sigmoïde

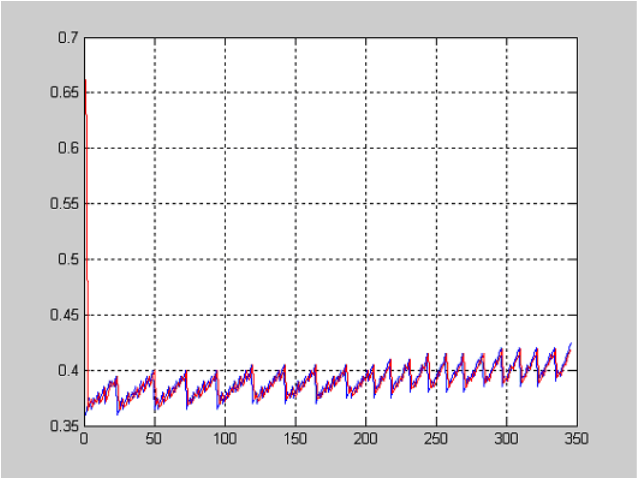

FIG. 2 – avec fonction binaire sigmoïde

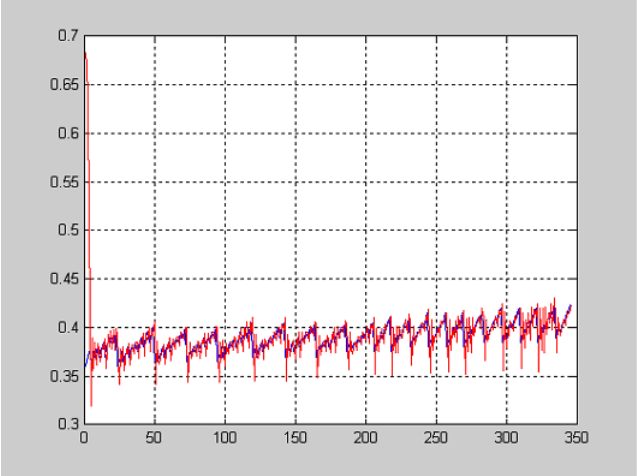

FIG. 3 – avec fonction bipolaire sigmoïde

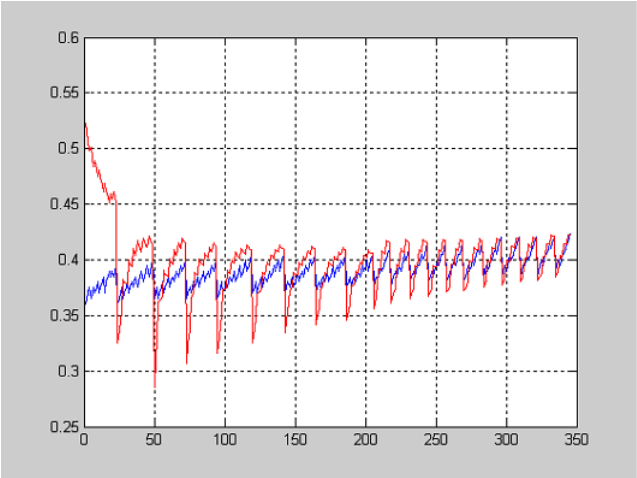

FIG. 4 – avec fonction arctangent

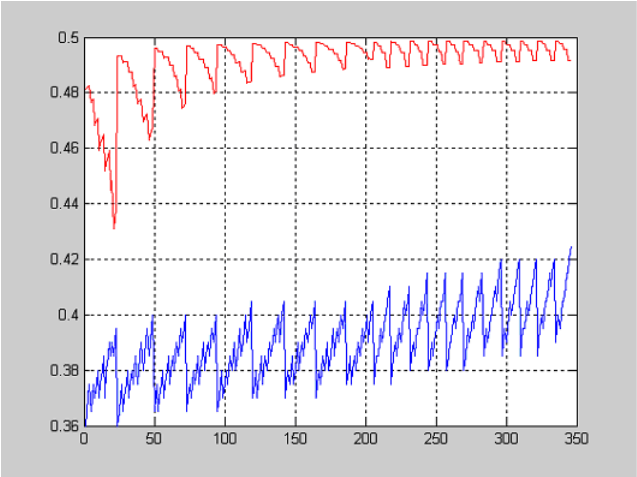

FIG. 5 – avec fonction non saturé gaussien

.1.2 Etude l'apprentissage en fonction les fonctions d'activations avec condition (Ng-Win)

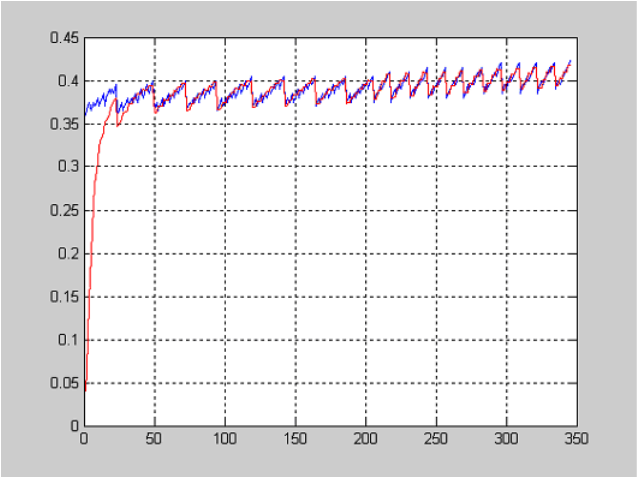

FIG. 6 – avec fonction sigmoïde

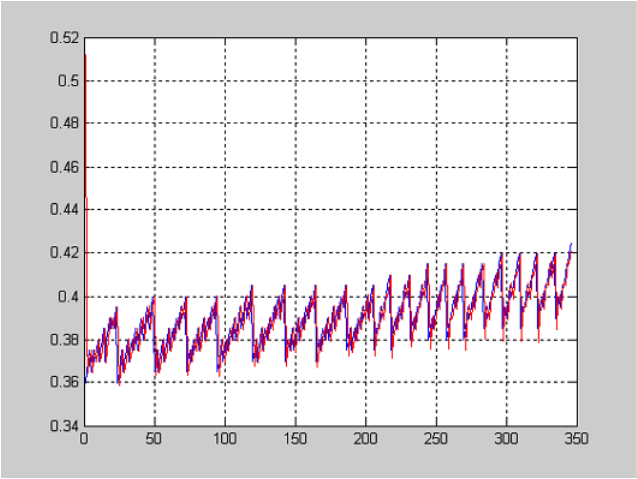

FIG. 7 – avec fonction binaire sigmoïde

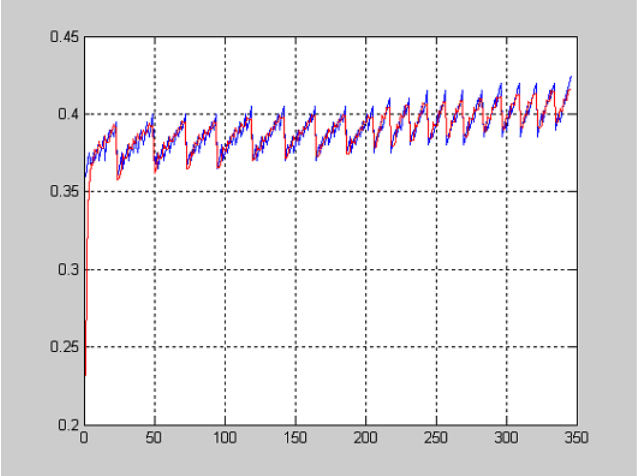

FIG. 8 – avec fonction bipolaire sigmoïde

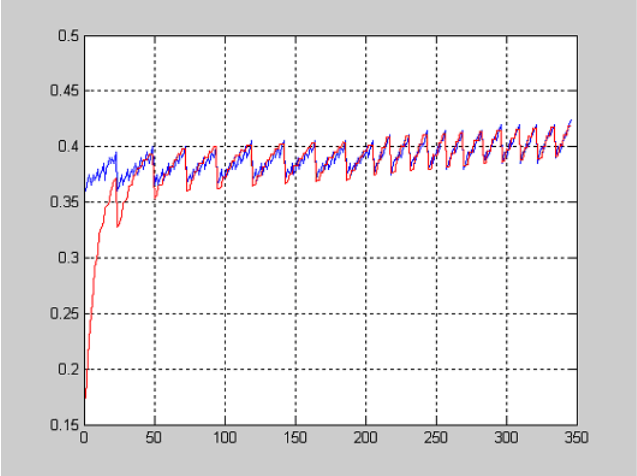

FIG. 9 – avec fonction arctangent

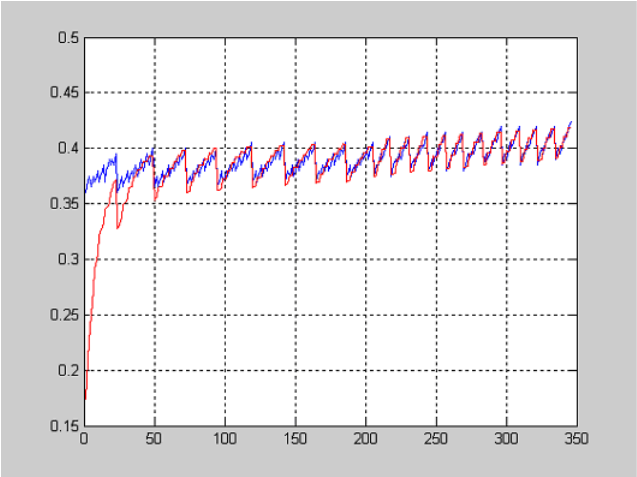

FIG. 10 – avec fonction non saturé gaussien

.2 Etude l'apprentissage en fonction les fonctions d'activations avec biais et avec et sans condition d'initialisation nughen et window

.2.1 Etude la apprentissage en fonction les fonctions d'activations avec biais et sans condition (Ng-Win)

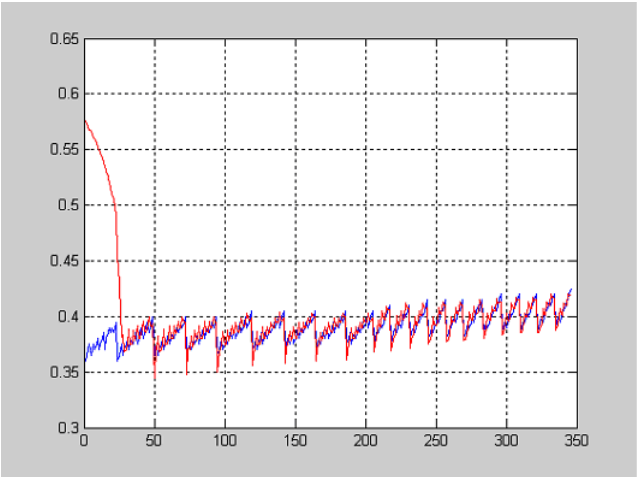

FIG. 11 – avec fonction sigmoïde

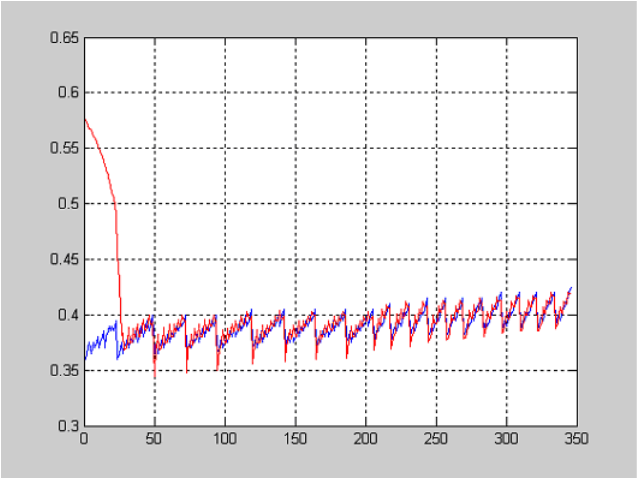

FIG. 12 – avec fonction binaire sigmoïde

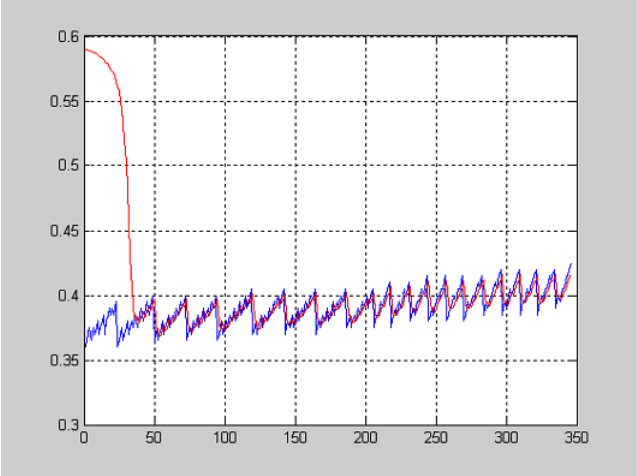

FIG. 13 – avec fonction bipolaire sigmoïde

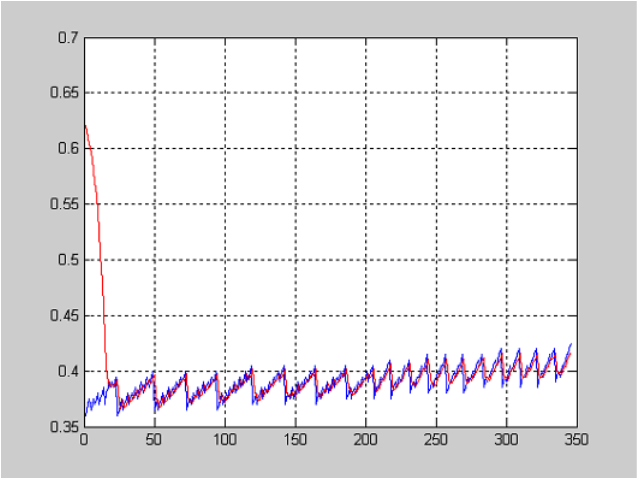

FIG. 14 – avec fonction arctangent

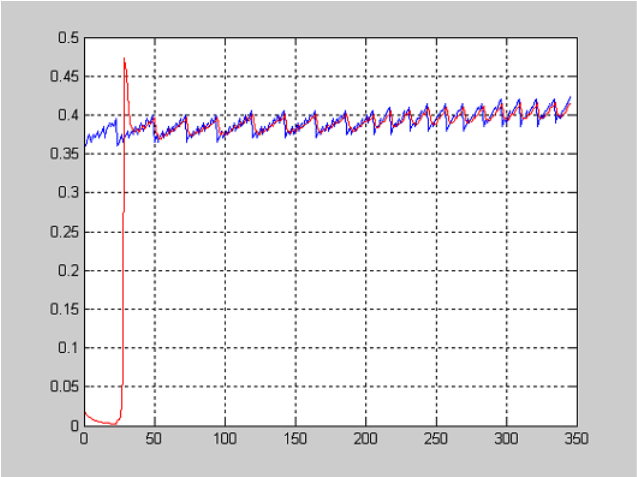

FIG. 15 – avec fonction non saturé gaussien

.2.2 Etude l'apprentissage en fonction les fonctions d'activations avec biais et avec condition (Ng-Win)

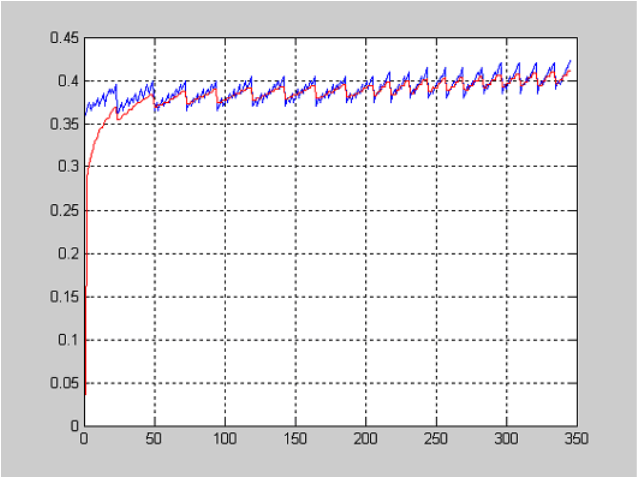

FIG. 16 – avec fonction sigmoïde

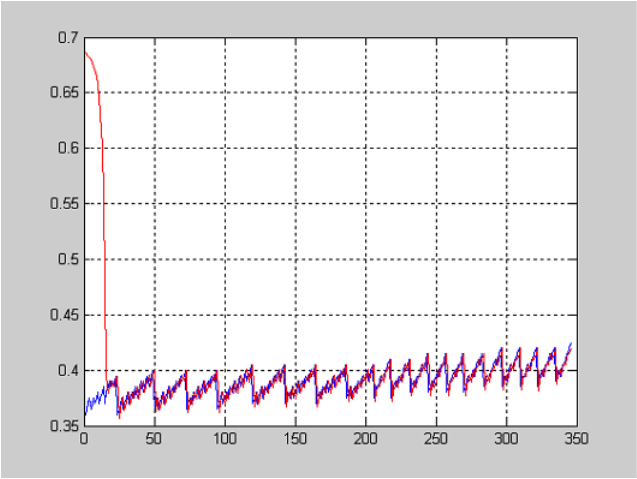

FIG. 17 – avec fonction binaire sigmoïde

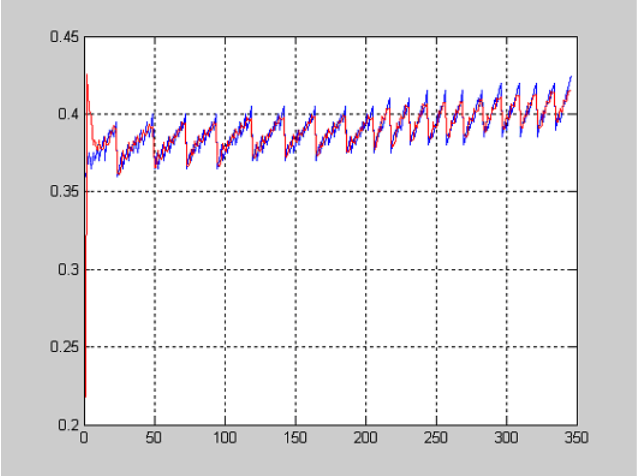

FIG. 18 – avec fonction bipolaire sigmoïde

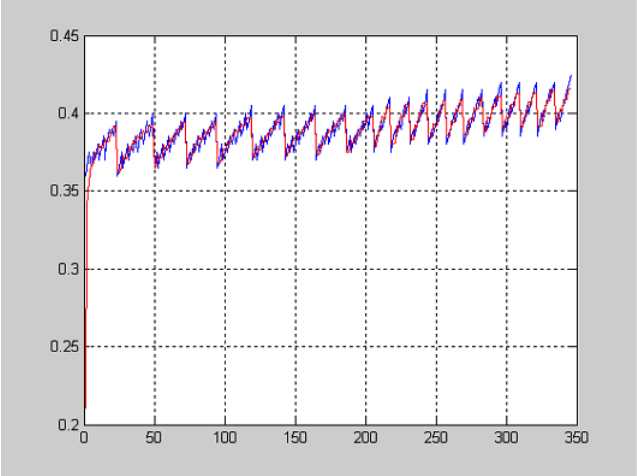

FIG. 19 – avec fonction arctangent

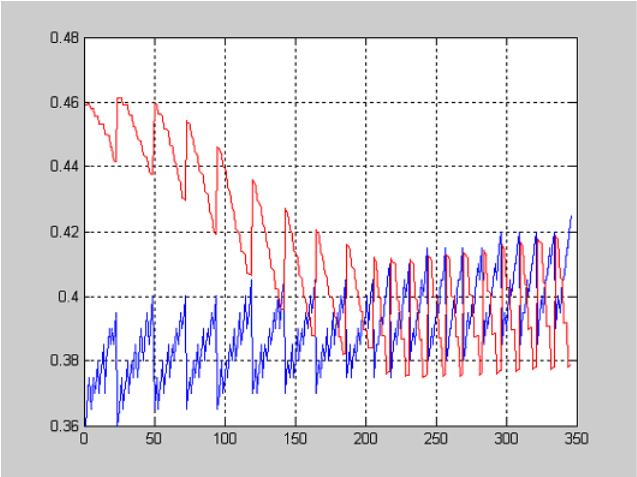

FIG. 20 – avec fonction non saturé gaussien

.3 Etude l'apprentissage en fonction les couches caches

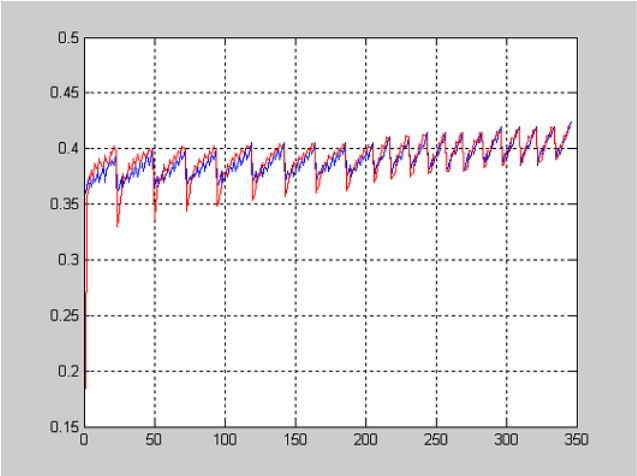

FIG. 21 – avec fonction sigmoïde

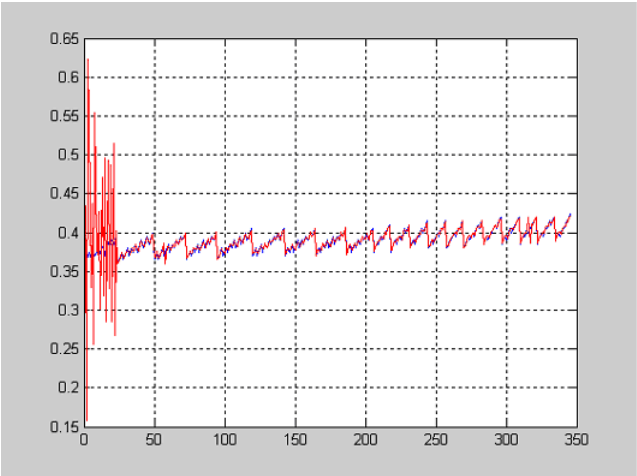

FIG. 22 – avec fonction sigmoïde

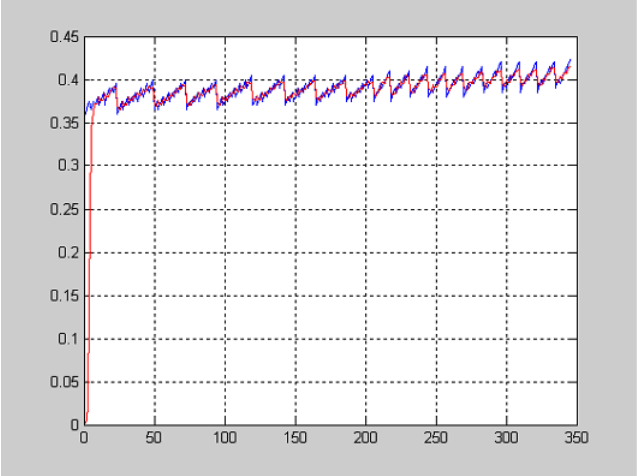

FIG. 23 – avec fonction sigmoïde