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 :

BENMOSBAH Amine
MECHERAOUI Choukri Adel

Thème :

Implémentation sur FPGA des méthodes MPPT : "P&O" et "floue optimisée par les Algorithmes Génétiques"

Membres du jury :

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

Juin 2006

(بسم الله الرحمن الرحيم)

ملخّص:

تهدف دراستنا هذه إلى تعريف و تفصيل لمختلف مراحل برمجة الدّارات المبرمجة "FPGA" باستخدام بطاقة التطوير "ميماك ديزاين" (Memec Design). كتطبيق لهذه الدّراسة اخترنا برمجة و مقارنة طريقتين لتتبّع نقطة الاستطاعة القصوى (طريقة "التشويش والملاحظة" و طريقة "المنطق الرّياضي اللاّواضح") اللّتان يتمّ استعمالهما لاستغلالٍ أحسن للطاقة المستمدّة من الألواح الشمسية. و بهذا نُبيِّن المزايا التي تقدّمها دارات "FPGA" بفضل اختصار وقت برمجتها، قِلة تكاليفها و مرونة استعمالها.

كلمات مفتاحية : دارة "FPGA"، لغة البرمجة "VHDL"، تتبّع نقطة الاستطاعة القصوى، التشويش و الملاحظة، المنطق الرّياضي اللاّواضح، كهروضوئي.

Résumé :

Ce mémoire présente les différentes étapes nécessaires à la programmation d'un circuit FPGA en utilisant la carte de développement « Memec Design ». Nous avons choisi comme application l'implémentation sur FPGA et la comparaison de deux commandes de poursuite du point de puissance maximale (méthode "Perturbation et Observation" et méthode de "la logique floue optimisée"), qui servent à exploiter au mieux la puissance délivrée par les panneaux solaires. On démontre ainsi l'avantage apporté par les circuits FPGA grâce à leur faible temps de développement, leur faible coût et leur flexibilité de fonctionnement.

Mots clés : Circuit FPGA, langage VHDL, MPPT, P&O, Logique floue, Photovoltaïque.

Abstract :

This memory presents the different steps necessary to the programming of FPGA circuits by using the development kit "Memec Design". We chose like application the implementation on FPGA and the comparison between two methods of Most Power Point Tracking (method of "Perturbation and Observation" and method of "optimized fuzzy logic"), which are used to exploit effectively the power delivered by the solar panels. Thus, we can show the advantages apported by FPGA circuits thanks to their weak development time, their low cost and their flexibility of operation.

Keywords: FPGA circuit, VHDL language, MPPT, P&O, Fuzzy logic, Photovoltaic.

Dédicaces

Je dédie ce travail à :

Mes très chers parents, qui m'ont toujours encouragé et guidé dans mes choix, et qui ont su me montrer la bonne voie,
Mes deux frères,
Ainsi qu'à toute ma famille et mes amis.

Amine

Je dédie ce travail à :

Mes parents,
Mon frère et ma sœur,
Toute ma famille,
Ainsi qu'à tous ceux qui me sont chers.

Choukri

Remerciements

Ce travail a été réalisé au sein du Laboratoire des Dispositifs de Communication et de Conversion Photovoltaïque (LDCCP) de l'Ecole Nationale Polytechnique.

Nous remercions le bon Dieu de nous avoir donné la volonté et la patience qui nous ont permis de mener à bien ce travail.

Nous tenons à exprimer notre profonde reconnaissance et nos chaleureux remerciements à notre promoteur Monsieur LARBES Chérif, qui nous a guidé et orienté tout au long de la réalisation de ce travail en prodiguant ses précieux conseils et ses vifs encouragements.

Nous tenons à remercier également Monsieur HADDADI Mourad, pour nous avoir honorés en acceptant de présider le jury, ainsi que Monsieur AIT CHEIKH Mohamed Salah et Madame HAMMAMI Latifa pour avoir accepté d'examiner ce travail.

Nous tenons aussi à exprimer nos plus vifs remerciements à Monsieur T.OBEIDI de l'Ecole Nationale Polytechnique pour sa précieuse aide et ses encouragements, ainsi qu'à Monsieur N.CHIKHI du Laboratoire de Microélectronique du Centre de Développement des Techniques Avancées (CDTA) pour ses éclaircissements,.

Nous ne saurons oublier de remercier toute personne qui, d'une manière ou d'une autre, nous a aidé dans l'élaboration de ce travail.

Table des matières

Introduction

Introduction

L'industrie moderne a des besoins de plus en plus importants en énergie. Les sources classiques d'énergie, qui sont les sources fossiles tel 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 telles que l'énergie solaire, éolienne, marémotrice 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, et 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 deux d'entre elles : La première est la méthode Perturbation et Observation (P&O), bien connue par les spécialistes de ce domaine, alors que la seconde utilise la logique floue dont les fonctions d’appartenance sont optimisées par les Algorithmes Génétiques.

D’autre part, les circuits FPGA (Field Programmable Gate Array), qui sont des circuits programmables standard, 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.

L’objet de ce projet de fin d’études consiste en l'implémentation des deux méthodes MPPT suscités sur circuit FPGA. Ce travail nécessite plusieurs étapes, il commence par la programmation en langage VHDL, la synthèse, le test et la simulation temporelle, et enfin le chargement du programme sur la carte FPGA.

Ce mémoire a été divisé en quatre 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 ensembles flous et l'optimisation par les Algorithmes Génétiques, explique le principe des deux méthodes MPPT qui nous intéressent (P&O et floue optimisée par les AGs) ;

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 ;

Le dernier chapitre est consacré à l'explication des programmes VHDL, à la visualisation et l'interprétation des résultats obtenus par simulation ainsi qu'à la comparaison entre les deux méthodes MPPT.

Chapitre I

Introduction à l'énergie photovoltaïque

Chapitre I

Introduction à l'énergie photovoltaïque

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 l'énergie solaire et de la production d'électricité grâce à l'effet photovoltaïque. Les principaux éléments du système photovoltaïque y sont étudiés et un survol de leurs caractéristiques est effectué.

I. L'énergie solaire

I.1. La ressource solaire

Le Soleil émet un rayonnement électromagnétique compris dans une bande de longueur d'onde variant de 0,22 à 10 microns (µm). L'énergie associée à ce rayonnement solaire se décompose approximativement ainsi :

– 9% dans la bande des ultraviolets (<0,4µm),
– 47% dans la bande visible (0,4 à 0,8µm),
– 44% dans la bande des infrarouges (>0,8µm).

L'atmosphère terrestre reçoit ce rayonnement à une puissance moyenne de 1,37 kilowatt au mètre carré (kW/m²) à plus ou moins 3 %. Elle en absorbe toutefois une partie, de sorte que la quantité d'énergie atteignant la surface terrestre dépasse rarement 1,2 kW/m² (1200W/m²). La rotation et l'inclinaison de la Terre font également que l'énergie disponible en un point donné varie selon la latitude, l'heure et la saison. Enfin, les nuages, le brouillard, les particules atmosphériques et divers autres phénomènes météorologiques causent des variations horaires et quotidiennes qui tantôt augmentent, tantôt diminuent le rayonnement solaire et le rendent diffus.

L'utilisation du rayonnement solaire comme source d'énergie pose donc un problème bien particulier. En effet, le rayonnement solaire n'est pas toujours disponible ; en outre, on ne peut l'emmagasiner ni le transporter. Le concepteur d'un système qui emploie le rayonnement solaire comme source d'énergie doit donc déterminer la quantité d'énergie solaire disponible à l'endroit visé et le moment où cette énergie est disponible.

Il faut d'abord comprendre l'effet de la rotation (moment de la journée) et de l'inclinaison (saison de l'année) de la Terre sur le rayonnement solaire. L'absorption atmosphérique est plus faible lorsque le Soleil se trouve à son point le plus haut dans le ciel, c'est-à-dire plein sud dans l'hémisphère nord. Dans ce cas, la distance que doit parcourir le rayonnement dans l'atmosphère est plus courte. C'est le « midi solaire », moment où le rayonnement solaire direct est le plus intense. Comme le Soleil est plus haut et que les journées sont plus longues en été, la quantité totale d'énergie reçue sur un plan horizontal y est plus grande qu'en hiver. La figure I.1 illustre ce phénomène, en reproduisant la trajectoire du Soleil dans le ciel au cours des quatre saisons de l'année.

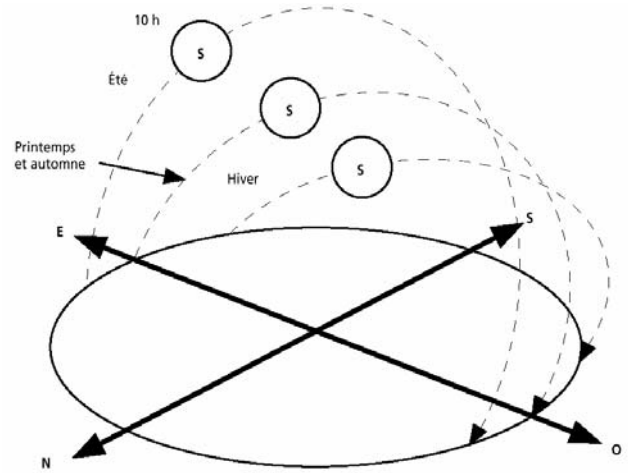


Figure 1.1 : Trajectoire du Soleil selon les saisons pour une latitude nord

Le rayonnement atteint son intensité maximale lorsque le plan est perpendiculaire aux rayons du Soleil, donc l'intensité du rayonnement solaire sur un plan quelconque augmente quand on l'incline vers le Soleil. On maximise par conséquent la quantité d'énergie solaire directe captée quand on change constamment l'inclinaison du plan pour le maintenir à angle droit avec les rayons du Soleil. Si le plan est fixe, la quantité d'énergie reçue sera moindre, car les rayons du Soleil le frapperont de biais la majorité du temps.

Lorsque l'inclinaison est égale à environ 35° par rapport à l'horizontale, le plan capte à peu près la même quantité d'énergie solaire toute l'année. Le rayonnement annuel capté est au maximum lorsque le plan est incliné à un angle égal à la latitude.

L'ombre projetée par les accidents du terrain (collines ou montagnes), par les immeubles et par les arbres peut également diminuer le rayonnement solaire frappant un plan quelconque. Pour les installations photovoltaïques, ce phénomène est particulièrement important parce que les cellules photovoltaïques et les modules sont branchés en série. L'obstruction d'une cellule peut causer une forte diminution de l'énergie produite et peut amener un phénomène de point chaud (*hot spot*), la cellule ombragée agissant comme récepteur et dissipant une certaine quantité d'énergie produite par les autres cellules.

Il est possible de représenter l'allure des courbes correspondant aux variations de l'ensoleillement selon différents paramètres (figure I.2) [3].

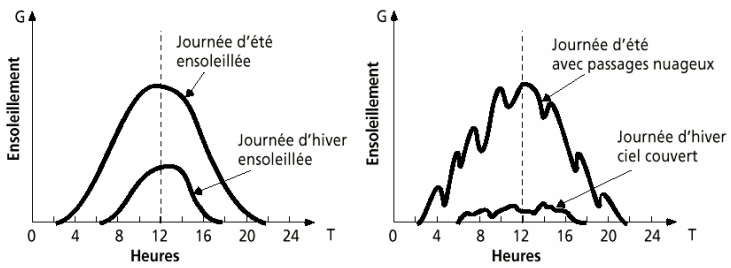


Figure 1.2 : Courbes d'ensoleillement typique par heure (latitude 45°)

I.2. Mesure du rayonnement solaire

L'ensoleillement (G) correspond à l'intensité du rayonnement solaire reçu sur un plan à un moment donné. Il s'exprime habituellement en watts par mètre carré (W/m²). L'ensoleillement varie de zéro, au lever du Soleil, à sa valeur maximale, typiquement au midi solaire.

L'ensoleillement peut également exprimer la quantité d'énergie solaire captée sur un plan pendant un intervalle déterminé. Il s'exprime habituellement en kilowattheure par mètre carré (kWh/m²), en «heure de soleil maximum», en mégajoule par mètre carré (MJ/m²) ou en calorie par centimètre carré (cal/cm²) pour l'intervalle déterminé, une journée ou une heure par exemple.

1 kWh/m² • jour = 1 heure de soleil maximum (1000W/m²) / jour
= 3,6 MJ/m² • jour
= 86 cal/cm² • jour

Le plus souvent, on exprime l'ensoleillement en «heures de soleil maximum», c'est-à-dire par le nombre équivalent d'heures par jour où l'éclairement est en moyenne de 1000 W/m². Les heures de soleil maximum sont un indice utile pour le dimensionnement des systèmes photovoltaïques, car ces systèmes sont habituellement évalués en watt-crête (Wc), c'est-à-dire sous un éclairement de 1000W/m².

La conception d'un système photovoltaïque exige des données sur l'ensoleillement. L'Organisation mondiale de la météorologie publie des cartes à l'échelle mondiale. La figure I.3 donne un exemple d'une carte indiquant la moyenne annuelle du rayonnement quotidien global sur une surface horizontale dans la région euro-méditerranéenne. En utilisant des données de rayonnement global sur un plan horizontal, il est possible de déterminer le rayonnement global pour une surface inclinée à un angle ø par rapport à l'horizontale.

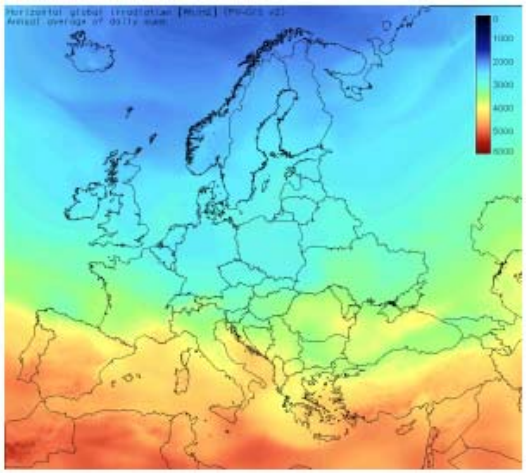


Figure 1.3 : Moyenne quotidienne du rayonnement solaire total sur une surface horizontale calculée en Wh/m² jour

II. Système photovoltaïque

II.1. L'effet photovoltaïque

L'énergie photovoltaïque (PV) est la transformation directe de la lumière en électricité. Elle n'est pas une forme d'énergie thermique, mais 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 Becquerel. 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.

II.2. La photopile

Cette photopile, qu'on appelle aussi cellule solaire ou photovoltaïque, est fabriquée à l'aide de matériaux semi-conducteurs (généralement le silicium). On peut la représenter comme une diode plate qui est sensible à la lumière.

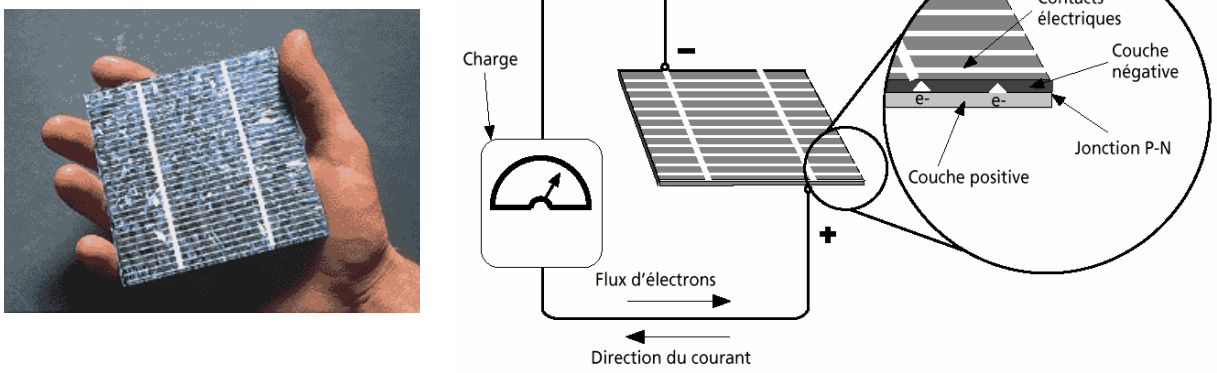


Figure 1.4 : Description d'une photopile ou cellule photovoltaïque

Quand un photon de lumière, d'énergie suffisante, heurte un atome sur la partie négative de cette diode, il excite un électron et l'arrache de sa structure moléculaire, créant ainsi un électron libre sur cette partie (zone N), et un trou sur l'autre partie (zone P). La création de paires électron-trou se traduit par la circulation d'un courant dans le circuit extérieur. La cellule photovoltaïque produira donc de l'électricité à courant continu (DC) et l'énergie qu'elle produit sera fonction principalement de la lumière reçue par la photopile.

Les premiers modules PV furent construits avec des cellules de silicium monocristallin. Le silicium étant le matériau le plus courant sur terre, mais un haut degré de pureté est requis pour en faire une pile photovoltaïque et le procédé est coûteux. Aujourd'hui encore, les cellules de silicium monocristallin sont toujours les plus efficaces avec un rendement de 16 % à 18 %, mais elles sont aussi les plus chères, avec un coût de production de modules de 3,50 à 4,00 $US/Wc.

Ces dernières années, l'arrivée de cellules de silicium polycristallin, avec une efficacité de 13 % à 15%, ainsi que la technologie du silicium amorphe (5 %) et l'emploi de semi-conducteurs en couches minces autres que le silicium ont révolutionné l'industrie photovoltaïque en diminuant les coûts de production. Aujourd'hui, la pile photovoltaïque a démontré dans les laboratoires une efficacité de 38 % grâce à la technologie des nouveaux matériaux (arséniure de gallium, tellurure de cadmium …).

II.2.1. Caractéristiques de la cellule photovoltaïque

Les caractéristiques du courant I = f (V) d'une cellule photovoltaïque peuvent être schématisées comme suit [3] :

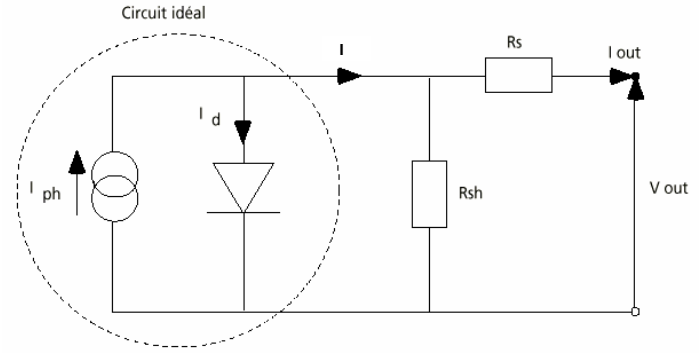


Figure 1.5 : Schéma équivalent d'une cellule photovoltaïque

Pour la cellule idéale :

$$I(V) = I_{ph}(\phi) - I_d(V) \quad \text{où}$$

I(V) = courant disponible
V = tension aux bornes de la jonction
$I_{ph}(ø)$ = courant produit par la photopile, ce courant est proportionnel au flux lumineux (*ø*)

$$I_d(V) = I_s \exp\left(\frac{qV}{JkT} - 1\right) \quad \text{où}$$

kT/q = 26 mV à 300 °K (27°C) pour le silicium
J = coefficient d'idéalité de la diode
I_s = courant de saturation de la diode

La photopile comporte en plus du circuit idéal une résistance série (R_s) et une résistance en shunt (R_{sh}).

- 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.

Une cellule photovoltaïque est caractérisée par les paramètres fondamentaux suivants :

- Courant de court-circuit ***Isc*** : c'est le courant maximal généré par la cellule lorsqu'elle est soumise à un court circuit : *V = 0* ;
- Tension à circuit ouvert ***Voc*** : c'est la tension aux bornes de la cellule sans charge, elle reflète la tension de seuil de la jonction *PN* ;
- Point de puissance maximale $\boldsymbol{P_{max}}$: c'est le point de fonctionnement *(Vmp, Imp)* où la cellule solaire génère sa puissance maximale.
- Facteur de remplissage (*Fill Factor)* ***FF*** : il correspond au rapport de puissance maximale sur le produit de *Voc* et *Isc* . et reflète la qualité de la cellule par rapport à une cellule idéale (*FF = 1).*

$$FF = \frac{V_{mp} \cdot I_{mp}}{V_{oc} \cdot I_{sc}}$$

- Le rendement de la cellule η : c'est le rapport de conversion de l'énergie lumineuse en énergie électrique, qui est égal au rapport de la puissance maximale de sortie sur la puissance des radiations lumineuses.

$$\eta = \frac{P_{\max}}{P_{in}} = \frac{V_{mp} \cdot I_{mp}}{P_{in}} = \frac{V_{oc} \cdot I_{sc} \cdot FF}{P_{in}}$$

La figure I.6 représente les caractéristiques courant-tension (*I=f(V)*) et puisance-tension (*P=f(V)*) d'une cellule photovoltaïque pour des conditions de travail constantes (température et ensoleillement fixes) [1].

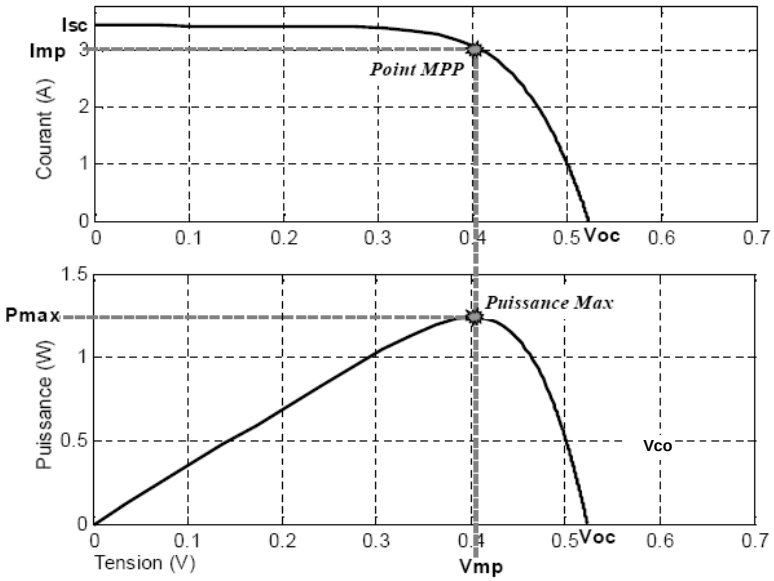


Figure I.6 : Caractéristiques I = f(V) et P= f(V) d'une cellule photovoltaïque

Une convention internationale définit la puissance d'une cellule en watt-crête. Le Wc est la puissance optimale fournie par la photopile sous des conditions de mesures normalisées c'est-à-dire pour un ensoleillement de 1 kW/m² et pour une température de jonction de la cellule de 25°C. Une simple cellule de silicium monocristallin ayant une surface de 100 mm x 100 mm aura un rendement d'environ 14% et produira environ 1,4 Wc à 0,5 v [3].

II.3. 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 un module photovoltaïque.

Figure I.7: Exemples de modules photovoltaïques

II.3.1. Association des modules photovoltaïques

Les modules peuvent également être connectés en série et en parallèle afin d'augmenter la tension et l'intensité du courant d'utilisation. Toutefois, il importe de prendre quelques précautions car l'existence de cellules moins efficaces ou l'occlusion d'une ou plusieurs cellules (dues à de l'ombrage, de la poussière, etc.) peuvent endommager les cellules de façon permanente.

II.3.1.1. Association en série

En additionnant des cellules ou des modules identiques en série, le courant de la branche reste le même mais la tension augmente proportionnellement au nombre de cellules (modules) en série.

Si les cellules des modules en série ne sont pas identiques ou si certaines cellules sont partiellement ombragées, la tension d'utilisation des modules en série sera légèrement diminuée. Pour une impédance de charge faible, les cellules moins efficaces peuvent devenir réceptrices si le courant d'utilisation est inférieur au courant produit par ces cellules. Ainsi, pour une impédance nulle (court-circuit), une cellule ombragée sera soumise à ses bornes à une tension inverse importante et la puissance qu'elle devra dissiper sera trop grande. En fonctionnant ainsi, on provoque l'échauffement de la cellule (*hot spot*), ce qui est susceptible de la détruire par claquage. Il convient donc de limiter la tension inverse maximale susceptible de se développer aux bornes d'une cellule en plaçant une diode parallèle (*by-pass*) au niveau de chaque module (voir figure I.7).

La diode parallèle limite la tension inverse par sa tension directe puisqu'elle devient passante. En court-circuit, la puissance dissipée par la cellule moins efficace se limite à l'ordre du watt, ce qui évite toute destruction. La diode parallèle est inopérante en fonctionnement normal et ne diminue donc pas le rendement des modules.

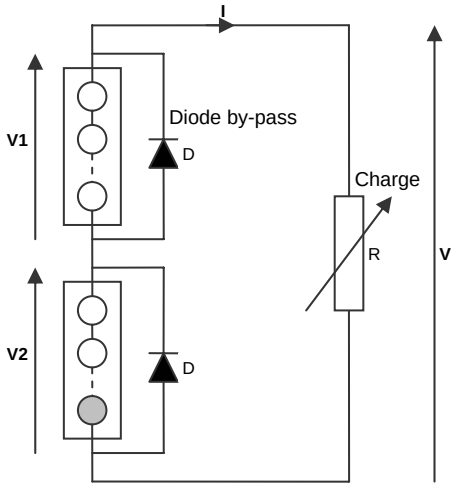


Figure I.8 : Modules en série avec diodes by-pass

II.3.1.2. Association en parallèle

En additionnant des modules identiques en parallèle, la tension de la branche est égale à la tension de chaque module et l'intensité augmente proportionnellement au nombre de modules en parallèle dans la branche.

Si les modules en parallèles ne sont pas identiques ou si quelques cellules d'un module sont ombragées, le courant d'utilisation total des modules sera plus faible. Pour une impédance de charge élevée, les modules moins performants deviendront récepteurs si la tension d'utilisation est supérieure à la tension produite par ces modules. Une dissipation de puissance importante peut devenir dangereuse au niveau de la cellule la plus faible de ces modules. Ainsi pour le cas le plus critique où la charge est nulle et le circuit ouvert, le courant des branches des modules performants se dissipera dans la branche la moins performante.

Bien que la cellule puisse dissiper un courant important sans être altérée, il est préférable de disposer d'une diode anti-retour. Celle-ci empêche aussi de gaspiller dans le module occulté une partie de la puissance produite par les modules fonctionnant normalement. Cette solution n'est valable que si la chute de tension provoquée par cette diode est négligeable devant la tension produite par les modules de la branche. En effet, cette diode est traversée, en fonctionnement normal, par le courant de la branche, ce qui introduit une perte de puissance permanente.

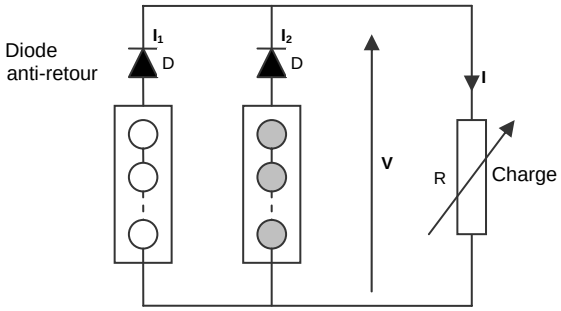


Figure I.9 : Modules en parallèle avec diodes anti-retour

II.3.1.3. Association série-parallèle

Généralement, on utilise un montage série-parallèle qui nous permet de régler à la fois la tension et le courant selon les caractéristiques de la charge. Les cellules photovoltaïques sont associées entre elles en série, et les modules sont associés en parallèle. On utilise alors les diodes by-pass pour éviter que les cellules les moins performantes deviennent consommatrices, et les diodes anti-retour pour éviter le retour du courant des autres modules lorsqu'un module est mal ensoleillé.

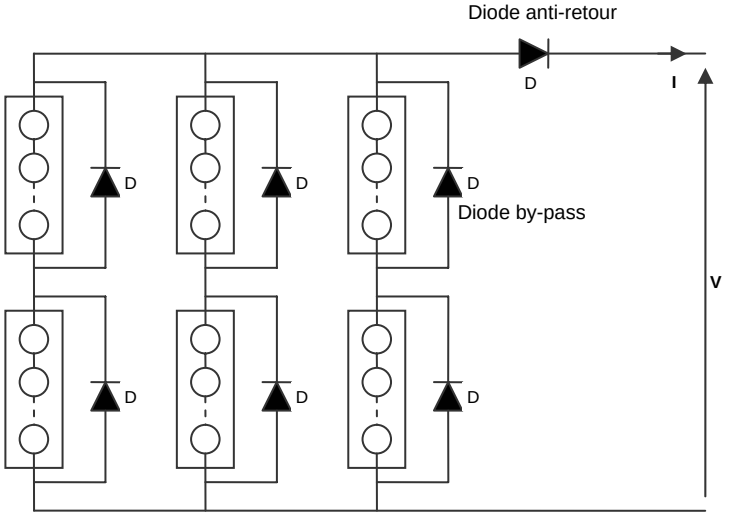


Figure I.10 : Montage série-parallèle de modules photovoltaïques

La figure I.11 montre les cinq courbes courant-tension possibles pour douze modules PV typiques de 50 Wc, ces courbes sont obtenues pour différentes commutations des modules dans le champ PV [3].

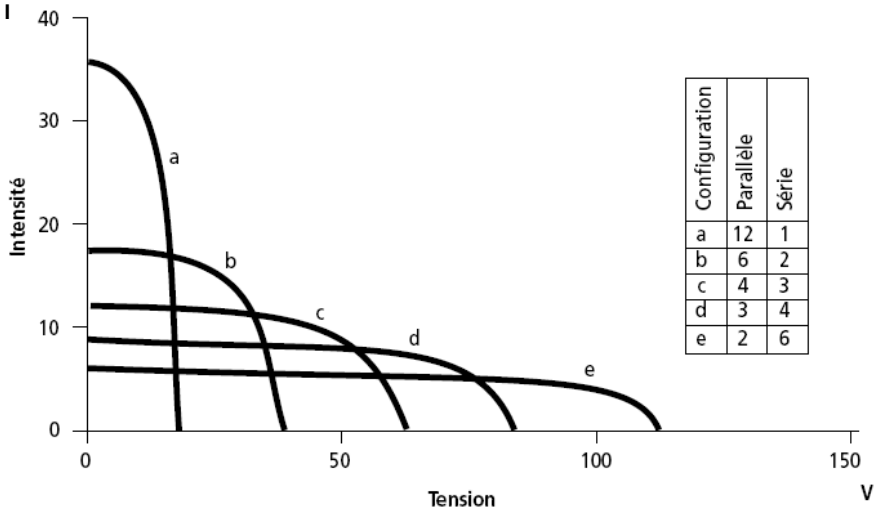


Figure I.11 : Commutation de12 modules de 50Wc dans un champ PV

II.3.2. Caractéristiques électriques des modules photovoltaïques

Il existe plusieurs indicateurs permettant de mesurer les performances des modules PV. La puissance crête (Wc) est un des indicateurs les plus significatifs, elle représente la puissance électrique maximale que le panneau peut fournir dans les conditions de mesures normalisées, c'est-à-dire lorsqu'il est connecté à une charge optimale, lorsque la température à la jonction des cellules est de 25 °C et lorsqu'il reçoit du soleil à une puissance de 1000W/m² (ceci correspond approximativement à une exposition perpendiculaire aux rayons du soleil le midi par temps clair).

La caractéristique courant-tension (*I=f(V)*) illustre la variation du courant de sortie en fonction de la tension de sortie. Elle représente aussi un indicateur important permettant de mesurer les performances des modules PV. A partir de cette caractéristique on peut tracer la caractéristique puissance-tension (*P=f(V)*).

Les conditions idéales conventionnelles sont très rarement remplies dans la pratique, et la variation de ces conditions est aléatoire et imprévisible. Le changement de l'ensoleillement et de la température influe directement sur les caractéristiques courant-tension et puissance-tension. Les figures I.12 et 1.13 montrent l'influence de ces deux paramètres sur les caractéristiques I-V et P-V [1]. D'abord, les deux caractéristiques sont tracées pour différents ensoleillements à température constante (figure I.12), ensuite elles sont tracées pour différentes températures à ensoleillement constant (figure I.13). On constate bien que les caractéristiques du panneau solaire dépendent fortement de l'ensoleillement et de la température.

Sur la figure I.12 on peut voir que le courant I_{sc} est très influencé par le changement de l'ensoleillement alors que la tension V_{oc} reste sensiblement constante. Par contre, on remarque sur la figure 13 que la tension V_{oc} est influencée par les changements de la température alors que le courant I_{sc} reste pratiquement inchangé.

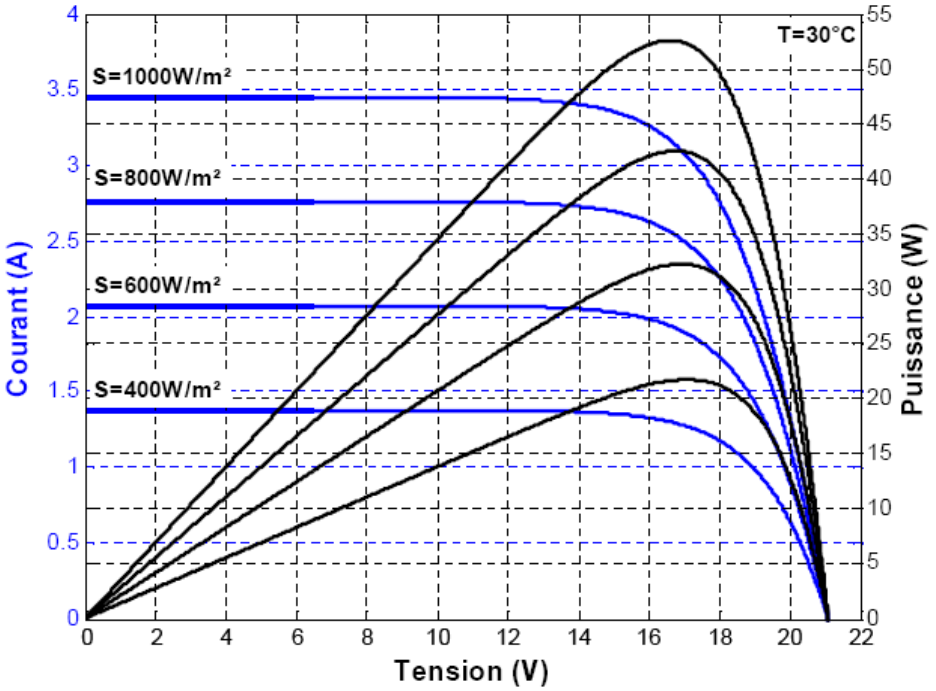


Figure I.12 : Caractéristiques I = f(V) et P = f(V) d'un panneau solaire constitué de 36 cellules en série pour différents ensoleillements S à température constante T = 30°C

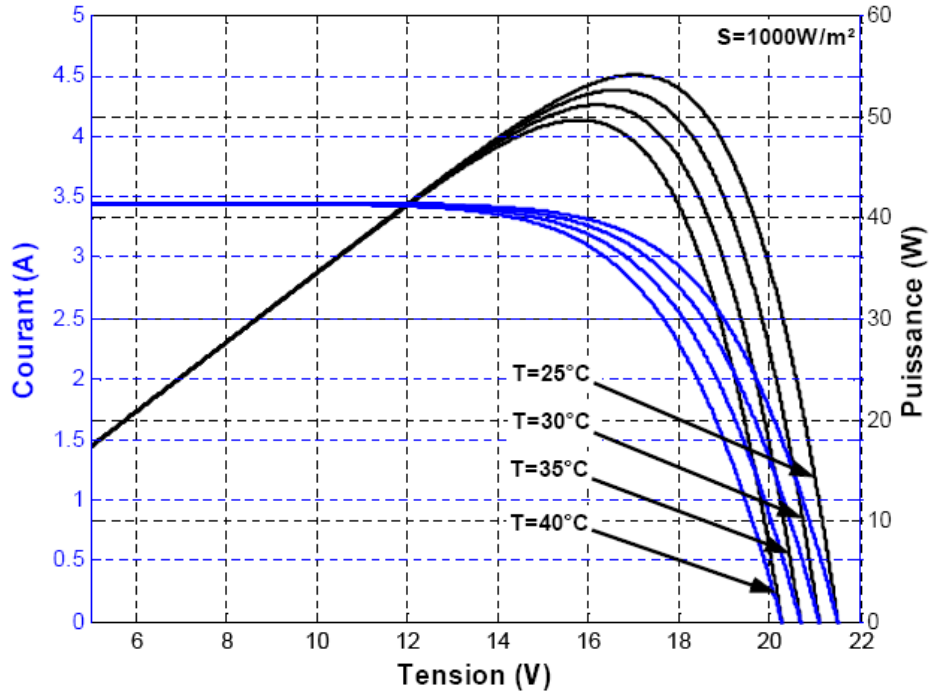


Figure I.13 : Caractéristique I = f(V) et P=f(V) d'un panneau solaire constitué de 36 cellules en série pour différentes températures T à ensoleillement constant S = 1000W/m²

Les figures I.12 et I.13 montrent aussi que la puissance de sortie ne dépend pas seulement de l'ensoleillement et de la température, mais aussi de la tension de fonctionnement *V* du panneau. En effet, la puissance de sortie est maximale seulement pour une certaine tension V_{mp}. C'est en ce point de fonctionnement que le panneau doit travailler pour qu'il ait un rendement maximal. Ce point est appelé MPP (Maximum Power Point).

II.4. Les différents types des systèmes photovoltaïques

Les modules PV sont les éléments de base de tout système photovoltaïque. Comme on vient de voir, ils peuvent être branchés en série pour augmenter leur tension d'utilisation et en parallèle pour augmenter leur courant. Cet ensemble est appelé le champ de modules PV. La figure I.14 montre des exemples de champs de modules photovoltaïques.

Figure I.14 : Exemples de champs de modules photovoltaïques

L'énergie fournie par le champ peut être utilisée pour charger des batteries qui fourniront l'électricité au moment voulu. Elle peut aussi être utilisée en reliant directement les modules à la charge sans les batteries (ex. : pompe solaire), ou en les branchant sur un réseau électrique. Il est également possible de combiner la sortie du champ PV avec d'autres sources d'énergie telles une génératrice ou une éolienne qui serviront d'appoint si l'ensoleillement n'est pas suffisant.

Bien qu'il existe une grande variété de systèmes photovoltaïques, on peut cependant les classer en deux groupes principaux. Le premier groupe inclut les systèmes autonomes, non reliés à un réseau électrique. L'autre groupe inclut des systèmes PV reliés différemment au réseau électrique. On cite dans ce qui suit les trois variantes des systèmes PV autonomes.

II.4.1. Système autonome sans batterie :

Ce type de système ne requiert pas de stockage d'électricité, soit parce que la production d'énergie des cellules est suffisante sous un faible éclairage (ex. : calculatrice), soit que le temps de fonctionnement de la charge n'est pas critique (ex. : pompe à eau : le réservoir d'eau sert de stockage).

II.4.2. Système autonome avec batterie :

C'est le système photovoltaïque le plus commun. Le champ PV sert de chargeur pour la batterie. L'électricité peut alors être utilisée en tout temps. Par exemple, ce système est bien adapté pour l'éclairage d'une maison où il faut de l'électricité lorsqu'il ne fait plus jour

II.4.3. Système hybride photovoltaïque/génératrice :

Ce système utilise les avantages de l'énergie photovoltaïque et de la génératrice au diesel, au propane ou à l'essence. Le système photovoltaïque fournit une énergie intermittente mais souvent moins coûteuse en régions éloignées. La génératrice sert d'énergie d'appoint selon la demande. Ce type de système s'applique particulièrement bien à des sites éloignés où il est important d'avoir de l'électricité à tout moment (ex. tours de communications, camps forestiers, etc.). Il peut également être couplé avec d'autres sources d'énergie telles les éoliennes et les microcentrales hydrauliques.

III. 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 I.15 illustre un système photovoltaïque simple avec batterie [1].

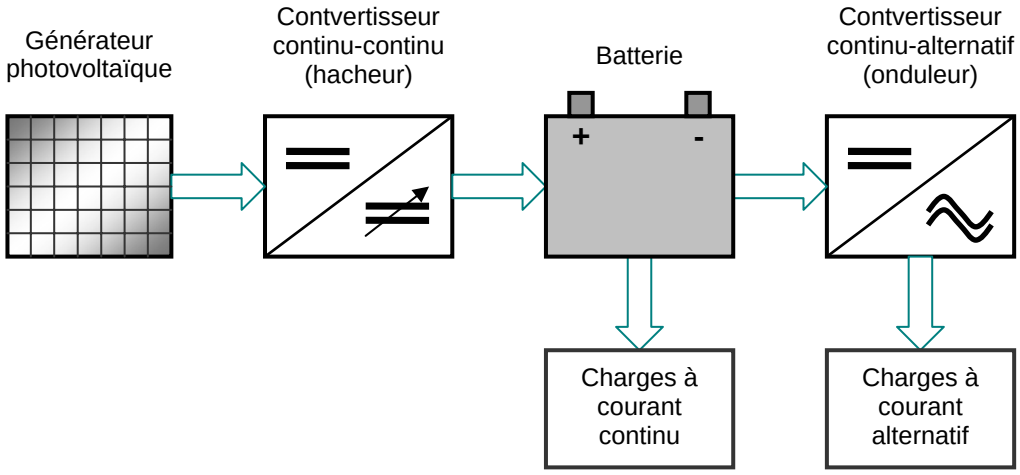


Figure I.15 : Composantes de base d'un système PV autonome avec batterie

III.1. Le générateur photovoltaïque

Le générateur photovoltaïque a été abordé plus haut. D'abord, on a vu que ce générateur se compose d'un ou de plusieurs modules photovoltaïques ; ces modules peuvent être assemblés suivant différents montages (série, parallèle, série-parallèle) afin d'obtenir la tension et le courant de travail voulues. Ensuite, on s'est intéressé aux caractéristiques de ces modules PV et particulièrement la caractéristique courant-tension ; on a finalement étudié l'effet des variations de la température et de l'ensoleillement sur cette caractéristique.

III.2. Le convertisseur continu-continu (hacheur)

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.

Le commutateur du convertisseur est commandé par un signal PWM (*Pulse Width Modulation*) ou MLI (*Modulation Largeur d'Impulsion*), avec une fréquence de commutation *Fs* fixe et un rapport cyclique *D* variable. La figure I.16 montre le signal de commande PWM du commutateur. D'abord on ferme le commutateur pendant un temps de fermeture égal à $D.T_s$, ensuite on l'ouvre durant un temps d'ouverture égal à *(1-D).Ts*, où :

- T_s est la période de commutation qui est égale à $1/F_s$;
- D est le rapport cyclique du commutateur $(D \in [0,1])$.

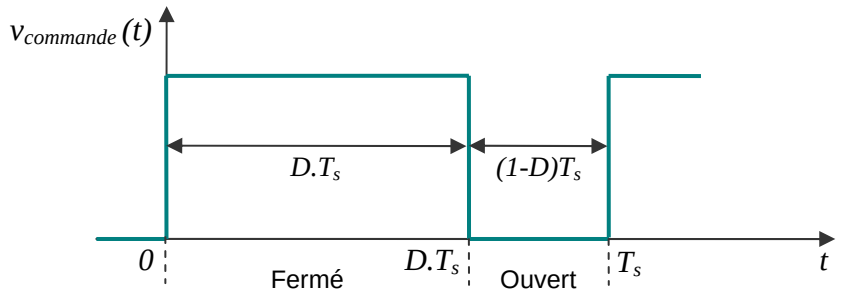


Figure I.16 : Tension de commande (PWM) du commutateur durant une période de commutation

Il existe différents types de convertisseurs continu-continu (DC-DC), les trois types 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 brève description de chacun d'eux. A noter que dans notre étude, nous avons utilisé le convertisseur Buck.

III.2.1. Hacheur dévolteur (Buck Converter)

Le convertisseur dévolteur a pour rôle principal de convertir la tension d'entrée en une tension de sortie inférieure. Il est aussi appelé convertisseur Buck, convertisseur abaisseur de tension ou hacheur série. La figure I.17 présente son circuit idéal.

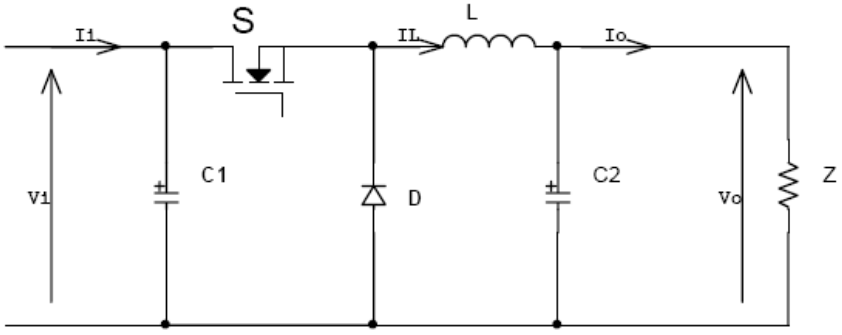


Figure I.17 : Circuit idéal du convertisseur dévolteur (Buck)

La figure I.18 explique le fonctionnement du convertisseur Buck durant une période de commutation T_s. Pendant le temps de fermeture $(t \in [0, D.T_s])$, la diode se bloque et un courant I_l circule dans la charge à travers l'inductance, cette dernière se charge alors d'énergie. Dès que le commutateur s'ouvre $(t \in [D.T_s, T_s])$, la source et la charge ne sont plus en contact, la diode se sature et l'inductance libère une énergie à la charge avec une diminution du courant I_l.

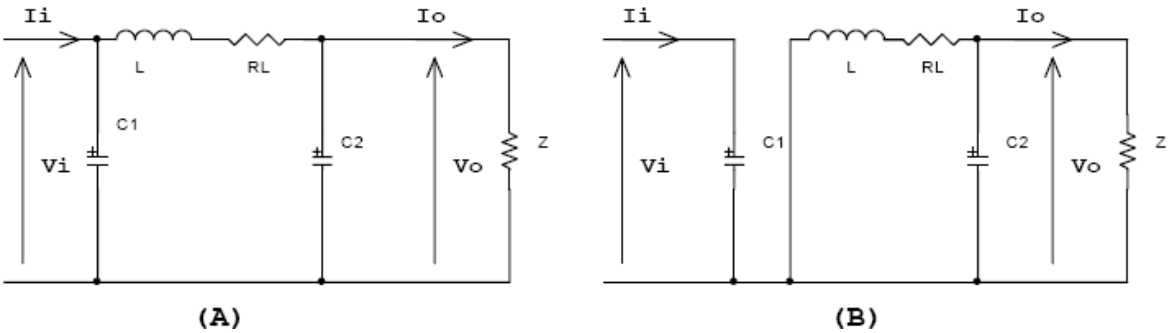


Figure I.18 : Circuits équivalents du convertisseur Buck pendant une période de commutation
(A) : pendant le temps de fermeture $(t \in [0, D.T_s])$
(B) : pendant le temps d'ouverture $(t \in [D.T_s, T_s])$

Le rapport de conversion d'un hacheur est le rapport entre la tension d'entrée V_i et la tension de sortie V_o. Pour un hacheur dévolteur, il est donné sous la forme suivante :

$$M(D) = \frac{V_o}{V_i} = D$$

La figure I.19 illustre la variation du rapport de conversion *M(D)* en fonction du rapport cyclique *D* pour un hacheur dévolteur. On voit bien que cette variation est linéaire. On remarque aussi que la valeur du rapport de conversion varie dans l'intervalle [0,1], ceci montre bien que le convertisseur Buck est un abaisseur de tension [2].

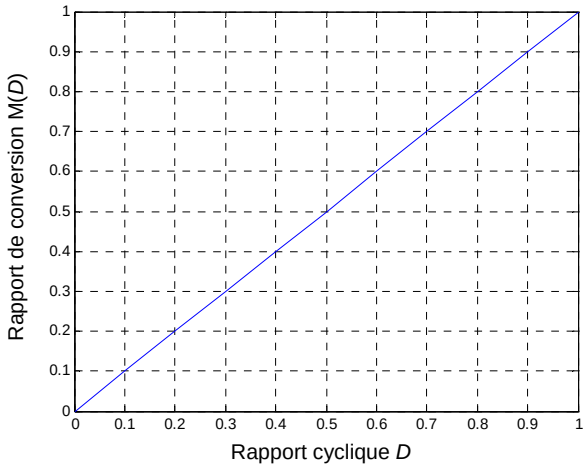


Figure I.19 : Variation du rapport de conversion pour un convertisseur Buck

III.2.2. Hacheur survolteur (Boost converter)

Le convertisseur survolteur a pour rôle principal de convertir la tension d'entrée en une tension de sortie supérieure. Il est aussi appelé convertisseur Boost, convertisseur élévateur de tension ou hacheur parallèle. Son circuit idéal est donné par la figure I.20.

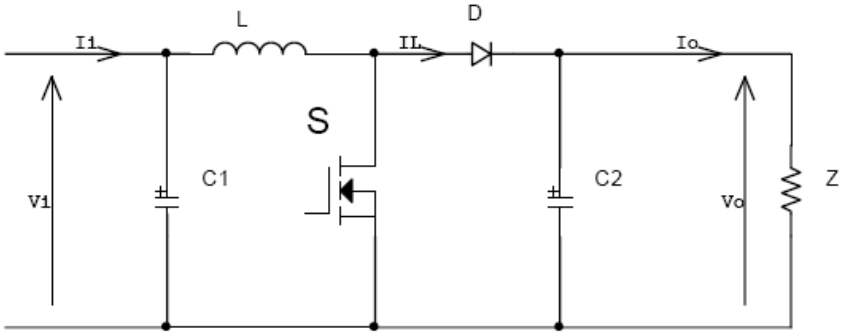


Figure I.20 : Circuit idéal d'un convertisseur survolteur (Boost)

La figure I.21 explique le fonctionnement du convertisseur Boost durant une période de commutation T_s. Pendant le temps de fermeture ($t \in [0, D.T_s]$), le courant dans l'inductance croit progressivement, au fur et à mesure elle emmagasine de l'énergie. Dès que le commutateur s'ouvre ($t \in [D.T_s, T_s]$), l'inductance L s'oppose à la diminution du courant I_L, ainsi elle génère une tension qui s'ajoute à la tension de source, qui s'applique sur la charge Z a travers la diode D.

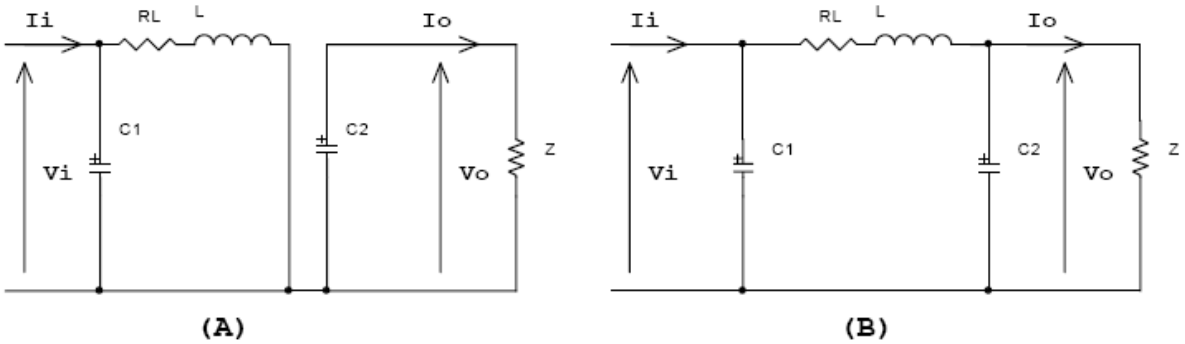


Figure I.21 : Circuits équivalents du convertisseur Boost pendant une période de commutation
(A) : pendant le temps de fermeture $(t \in [0, D.T_s])$
(B) : pendant le temps d'ouverture $(t \in [D.T_s, T_s])$

Le rapport de conversion d'un hacheur Boost est donné sous la forme suivante :

$$M(D) = \frac{V_o}{V_i} = \frac{1}{1-D}$$

La figure I.22 illustre la variation du rapport de conversion *M(D)* en fonction du rapport cyclique *D* pour un hacheur survolteur. On voit que cette variation est non linéaire. On remarque aussi que la valeur du rapport de conversion est toujours supérieure à 1, ceci montre bien que le convertisseur Boost est un élévateur de tension.

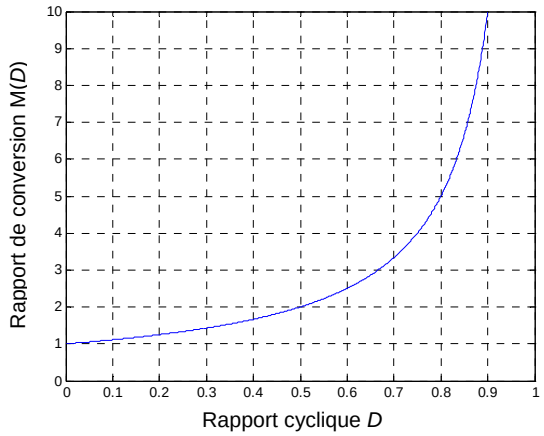


Figure I.22 : Variation du rapport de conversion pour un convertisseur Boost

III.2.3. Hacheur dévolteur-survolteur (Buck-Boost converter)

Le convertisseur dévolteur-survolteur (ou Buck-Boost) combine les propriétés des deux hacheurs déjà étudiés. Il peut être employé pour transformer idéalement n'importe quelle tension continue d'entrée en n'importe quelle tension continue de sortie. La figure I.23 illustre son circuit idéal.

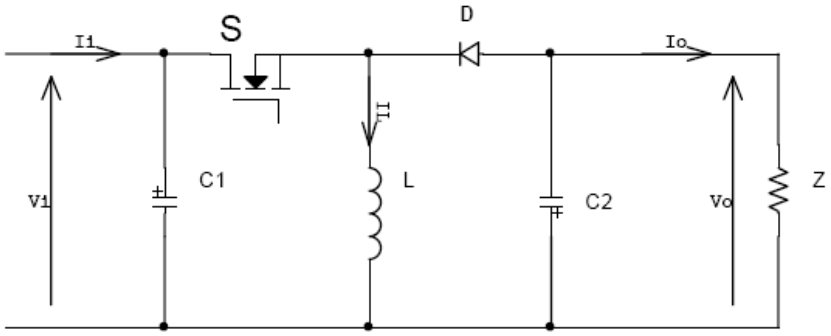


Figure I.23 : Circuit idéal d'un convertisseur dévolteur-survolteur (Buck-Boost)

La figure I.24 explique le fonctionnement du convertisseur Buck-Boost durant une période de commutation T_s. Pendant le temps de fermeture ($t \in [0, D.T_s]$), la tension de la source est appliquée à l'inductance qui se charge d'énergie. Durant la période d'ouverture ($t \in [D.T_s, T_s]$), la tension de l'inductance se trouve appliquée à la charge Z et son courant circule dans le sens inverse des aiguilles d'une montre à travers la diode D. Ainsi, la tension de sortie V_0 du convertisseur sera négative.

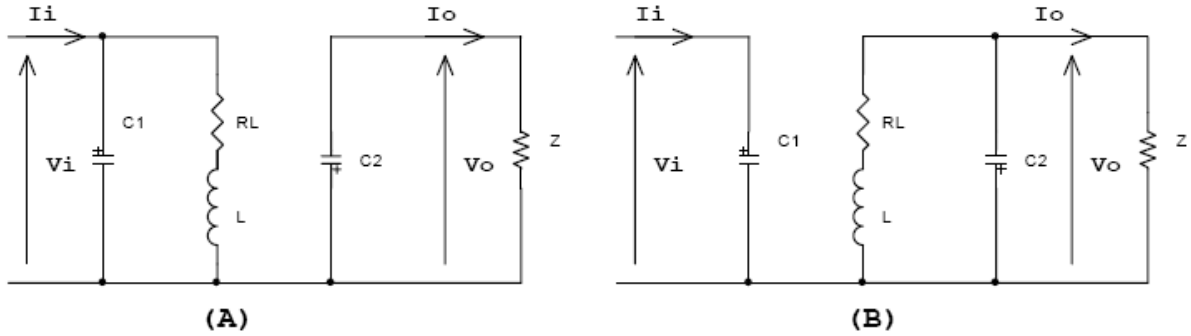


Figure I.24 : Circuits équivalents du convertisseur Buck-Boost pendant une période de commutation
(A) : pendant le temps de fermeture $(t \in [0, D.T_s])$
(B) : pendant le temps d'ouverture $(t \in [D.T_s, T_s])$

Le rapport de conversion d'un hacheur Buck-Boost est une relation non linéaire, il est donné par l'expression suivante :

$$M(D) = \frac{V_o}{V_i} = -\left(\frac{D}{1-D}\right)$$

La figure I.25 illustre les variations du rapport de conversion du convertisseur dévolteur-survolteur.

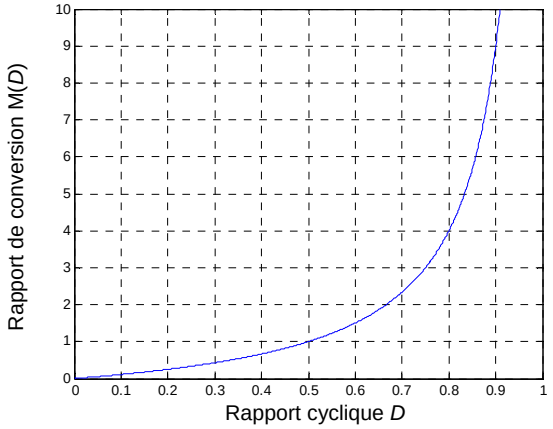


Figure I.25 : Variation du rapport de conversion pour un convertisseur Buck-Boost

III.3. Les batteries

La batterie sert à stocker l'énergie produite par le champ de modules PV. Il y a nécessité de stockage chaque fois que la demande énergétique est décalée dans le temps vis-à-vis de l'apport énergétique solaire. En effet, la demande énergétique est fonction de la charge à alimenter, les appareils utilisés fonctionnent soit en continu, soit à la demande ; de plus, l'apport énergétique solaire est périodique (alternance jour/nuit, été/hiver) et aléatoire (conditions météorologiques). Ce décalage entre la demande et l'apport énergétique nécessite un stockage d'électricité. Le système tampon utilisé le plus couramment pour les systèmes photovoltaïques est la batterie d'accumulateurs électrochimiques.

Une batterie utilisée dans un système PV doit remplir certaines conditions : par exemple un nombre important de charges et de décharges sans altération, une faible autodécharge, un rendement électrique élevé et une maintenance légère.

Dans un système photovoltaïque, la batterie remplit trois fonctions importantes :

- **Autonomie :** Une batterie permet de répondre aux besoins de la charge en tout temps, même la nuit ou par temps nuageux.
- **Courant de surcharge :** Une batterie permet de fournir un courant de surcharge pendant quelques instants. Ceci est nécessaire pour faire démarrer les moteurs et les autres appareils requérant un courant de démarrage de 3 à 5 fois supérieur au courant d'utilisation.
- **Stabilisation de la tension**. Une batterie permet de fournir une tension constante, en éliminant les écarts de tension du champ PV et en permettant aux appareils un fonctionnement à une tension optimisée.

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). Chacune a ses propres particularités et, selon les méthodes de construction, elles auront des caractéristiques de fonctionnement très différentes. La batterie au plomb-acide est la plus connue, étant utilisée depuis plus de 150 ans pour fournir le courant de démarrage des voitures et l'électricité des systèmes d'urgence entre autres. La batterie au nickel-cadmium a été conçue pour répondre à un besoin prolongé de stockage d'énergie dans des conditions de fonctionnement extrême et de maintenance minimale. Il existe plusieurs types de batteries au plomb-acide ainsi que quelques types de batteries au nickel-cadmium.

III.3.1. Caractéristiques des batteries

Les systèmes photovoltaïques exigent habituellement des batteries qui peuvent être chargées pendant le jour et déchargées durant la nuit. Ces batteries doivent pouvoir fonctionner ainsi pendant des années sans marquer plus qu'une détérioration minimale de leur rendement, tout en satisfaisant la demande, les jours où il n'y a que peu ou pas de soleil. Il y a donc deux paramètres importants à considérer lors de la conception du système :

- **Jours d'autonomie (réserve) :** nombre de jours pendant lesquels la batterie doit fournir la puissance requise sans être rechargée ni subir de dommages.
- **Fonctionnement à cycle de décharge faible ou profonde :** autonomie assurée soit par une batterie de grande capacité à faible décharge (cycle à faible décharge), soit par une batterie de capacité moindre à décharge profonde (cycle à décharge profonde).

La corrélation de ces paramètres permet d'estimer la capacité de la batterie, cette dernière s'exprime comme suit [3]:

$$C = Ch \times R / P_d$$

où Ch = charge quotidienne (en Ah)
R = réserve ou jours d'autonomie
P_d = profondeur de décharge pour chaque cycle
C = capacité de la batterie (en Ah)

Les systèmes qui fonctionnent selon un cycle à faible décharge sont dimensionnés de manière à employer au maximum de 15 % à 25 % de la capacité de la batterie chaque jour. En fonctionnant ainsi, on prolonge la durée de vie des batteries et on dispose d'une réserve automatique pour pallier la non-production d'énergie par mauvais temps. On emploie pour ce type de fonctionnement les batteries d'accumulateurs au plomb-calcium.

Dans le cas d'un système qui fonctionne selon un cycle à décharge profonde, la batterie d'accumulateurs est dimensionnée en fonction d'une consommation journalière de 50% à 80% de sa capacité. Les accumulateurs au plomb-antimoine peuvent supporter ce genre de traitement pendant plusieurs années. Presque toute la capacité est utilisée, de sorte qu'il faut moins d'accumulateurs ; mais on dispose aussi de peu de réserve pour alimenter la charge pendant les périodes prolongées de mauvais temps.

Chapitre II

Poursuite du point de puissance maximale (MPPT)

Chapitre II

Poursuite du point de puissance maximale (MPPT)

Comme il a été expliqué dans le chapitre I, le point de puissance maximale MPP (Maximum Power Point), qui correspond au point de fonctionnement optimal, se déplace en fonction des conditions de fonctionnement du module photovoltaïque. Les conditions de fonctionnement les plus signifiantes sont la température du module et l'ensoleillement qu'il subit.

Ces conditions atmosphériques subissent des variations au cours du temps et peuvent parfois changer brusquement. On peut rencontrer une variation brutale de l'ensoleillement dans un panneau fixe lors du passage de nuages par exemple, ou dans les véhicules solaires lorsqu'ils passent dans une zone ombrée par des arbres ou des constructions. On rencontre le cas d'une diminution rapide de la température lorsqu'il y'a des averses soudaines par exemple, mais le cas d'une augmentation brutale de la température ne se produit pratiquement jamais.

Pour que le panneau fournisse sa puissance maximale disponible, il faut donc une adaptation permanente de la charge avec le générateur photovoltaïque. Pour suivre les variations du point MPP, on doit utiliser un dispositif de poursuite automatique appelé MPPT (Maximum Power Point Tracker). Ce dispositif consiste en un contrôleur qui a pour rôle de rechercher continuellement le point MPP quelque soient les conditions de fonctionnement. Ce contrôleur MPPT doit satisfaire certaines conditions tel que la stabilité et la rapidité de la réponse.

La figure II.1 présente le schéma synoptique d'un système photovoltaïque avec MPPT.

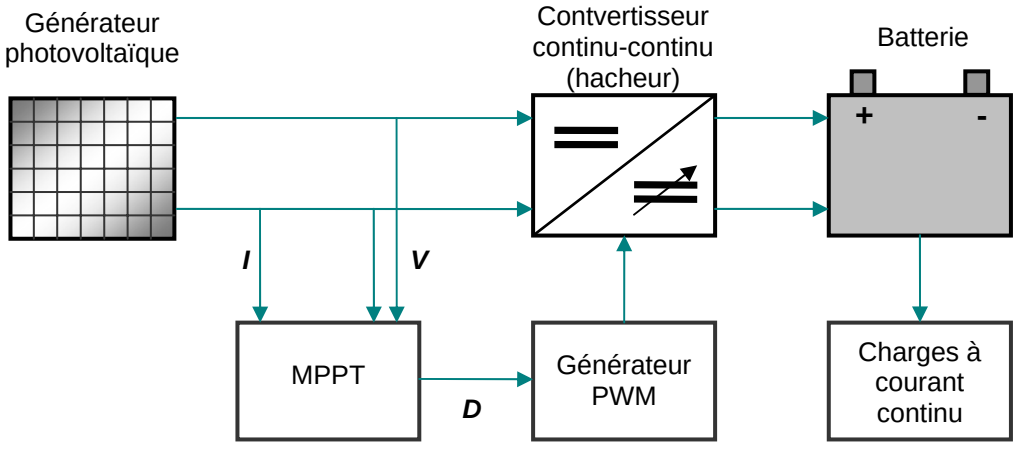


Figure II.1 : Schéma synoptique d'un système photovoltaïque avec MPPT

Il existe une panoplie de méthodes MPPT. Dans ce mémoire, on s'est intéressé à deux méthodes en particulier : la méthode Perturbation et Observation (P&O) et la méthode du contrôleur flou optimisé par les Algorithmes Génétiques. Ces deux méthodes utilisent le principe de la contre réaction de puissance.

Pour introduire la méthode du contrôleur flou optimisé, on commence par donner un survol de la logique floue et des Algorithmes Génétiques. Ces deux techniques font partie du champ de l'Intelligence Artificielle (IA), et constituent une approche originale et moderne permettant de résoudre un grand nombre de problèmes que les méthodes classiques n'arrivent pas ou mettent trop de temps à résoudre. Elles sont en train de s'imposer ces dernières années dans beaucoup de domaines allant de la finance jusqu'au domaine de l'électronique, du contrôle et de l'automatisation.

I. La logique floue

Le terme d'ensemble flou apparaît pour la première fois en 1965 lorsque le professeur Lotfi A. Zadeh, de l'université de Berkeley aux USA, publie un article intitulé « Ensembles flous » (Fuzzy sets). Il a réalisé depuis de nombreuses avancées théoriques majeures dans le domaine. En 1975, le professeur Mamdani à Londres présente les résultats très encourageants qu'il a obtenus sur la conduite d'un moteur à vapeur.

La première véritable application industrielle de la logique floue a été le contrôle d'un four à ciment réalisé par la société danoise F.L.Smidth en 1978. Mais c'est au Japon que la logique floue connaît son véritable essor à la fin des années 1980 avec de nombreuses applications dans l'électroménager et l'électronique grand public. Dans l'industrie, le traitement des eaux, les grues portuaires, les métros, les systèmes de ventilation et de climatisation sont aussi touchés.

Machines à laver sans réglage, caméscopes anti-bougé, contrôle de l'air conditionné et transmissions automatiques dans l'automobile font partie des nombreuses innovations qui ont fait connaître le terme « logique floue » à un large public. Dans le domaine des processus de production, continue et par lots, et dans les automatismes les applications se sont également multipliées.

La logique floue permet de systématiser ce qui est du domaine de l'empirisme, et donc difficile à maîtriser, c'est une approche essentiellement pragmatique, efficace et générique. La théorie des ensembles flous fournit une méthode pertinente et facilement réalisable dans des applications temps réel ; elle permet de transcrire et rendre dynamiques les connaissances des concepteurs ou des opérateurs. Cet aspect adaptable et universel de la logique floue permet de s'attaquer à l'automatisation de procédures telles que la mise en route, le réglage de paramètres, pour lesquelles peu d'approches existaient auparavant.

La logique floue ne remplace pas nécessairement les systèmes de régulation conventionnels, elle en est plutôt un complémentaire. Ses avantages viennent notamment de ses capacités à :

- formaliser et simuler l'expertise d'un opérateur ou d'un concepteur dans la conduite et le réglage d'un procédé ;
- donner une réponse simple pour les procédés dont la modélisation est difficile ;
- prendre en compte sans discontinuité des cas ou exceptions de natures différentes, et les intégrer au fur et à mesure dans l'expertise ;
- prendre en compte plusieurs variables et effectuer de la «fusion pondérée» des grandeurs d'influence.

I.1. Théorie des ensembles flous

I.1.1. Notion d'appartenance partielle

La théorie des ensembles flous repose sur la notion d'appartenance partielle : chaque élément appartient partiellement ou graduellement aux ensembles flous qui ont été définis. Les contours de chaque ensemble flou ne sont pas nets, mais flous ou graduels (figure II.2).

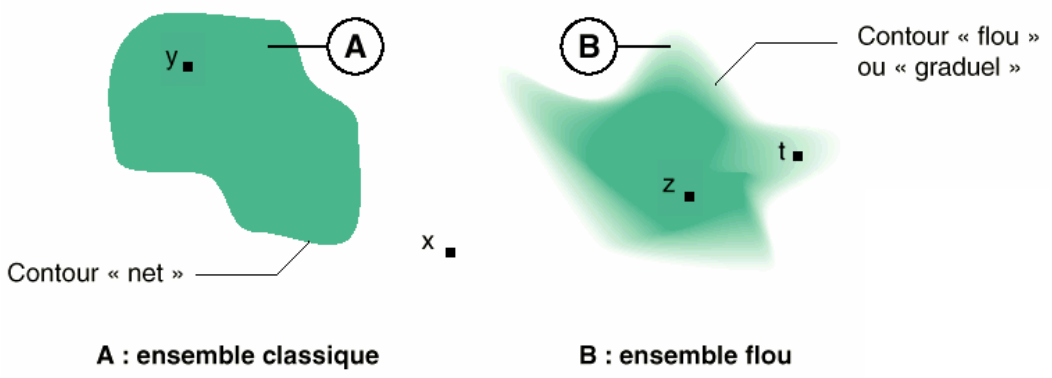


Figure II.2: comparaison d'un ensemble classique et d'un ensemble flou

La figure ci-dessus nous permet de dire que :

- x n'appartient ni à A ni à B,
- y appartient totalement à A,
- z appartient totalement B,
- t appartient partiellement à B.

I.1.2. Fonctions d'appartenance

Un ensemble flou est défini par sa « fonction d'appartenance », qui correspond à la notion de « fonction caractéristique » en logique classique. Supposons que nous voulions définir l'ensemble des personnes de « taille moyenne ». En logique classique, nous conviendrons par exemple que les personnes de taille moyenne sont celles dont la taille est comprise entre 1,60 m et 1,80 m. La fonction caractéristique de l'ensemble donne « 0 » pour les tailles hors de l'intervalle [1,60 m ; 1,80 m] et « 1 » dans cet intervalle.

L'ensemble flou des personnes de « taille moyenne » sera défini par une « fonction d'appartenance » qui diffère d'une fonction caractéristique par le fait qu'elle peut prendre n'importe quelle valeur dans l'intervalle [0,1]. A chaque taille possible correspondra un « degré d'appartenance » à l'ensemble flou des « tailles moyennes » compris entre 0 et 1. La figure II.3 fait une comparaison entre fonction caractéristique et fonction d'appartenance.

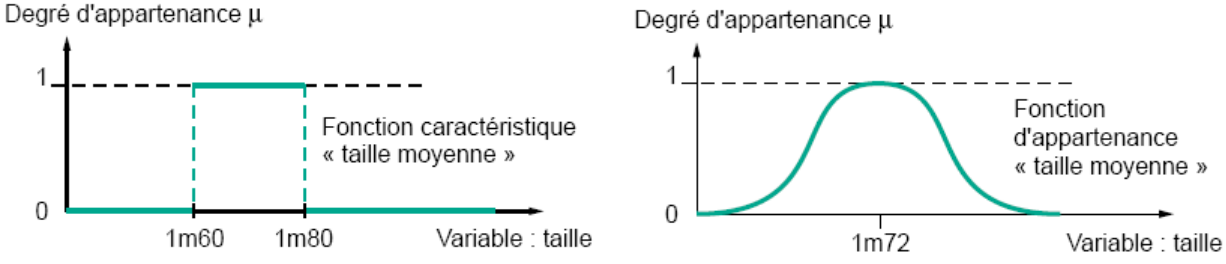


Figure II.3 : Comparaison entre fonction caractéristique et fonction d'appartenance

Les fonctions d'appartenance peuvent théoriquement prendre n'importe quelle forme. Toutefois, elles sont souvent définies par des segments de droites, et dites « linéaires par morceaux », elles peuvent alors être trapézoïdales ou triangulaires. Ces fonctions d'appartenance sont très utilisées car elles sont simples, elles comportent des points permettant de définir les zones où la notion est totalement vraie ou totalement fausse, ce qui simplifie le recueil d'expertise. On utilisera dans tout ce qui suit les fonctions d'appartenance triangulaires. On utilisera dans tout ce qui suit les fonctions d'appartenance de ce type.

Dans certains cas, les fonctions d'appartenance peuvent être égales à 1 pour une seule valeur de la variable et égales à 0 ailleurs, dans ce cas elles prennent le nom de « fonctions d'appartenance singletons ». Un singleton flou défini sur une variable réelle (taille) est la traduction dans le domaine flou d'une valeur particulière de cette variable (ex. : taille de Samir) [4].

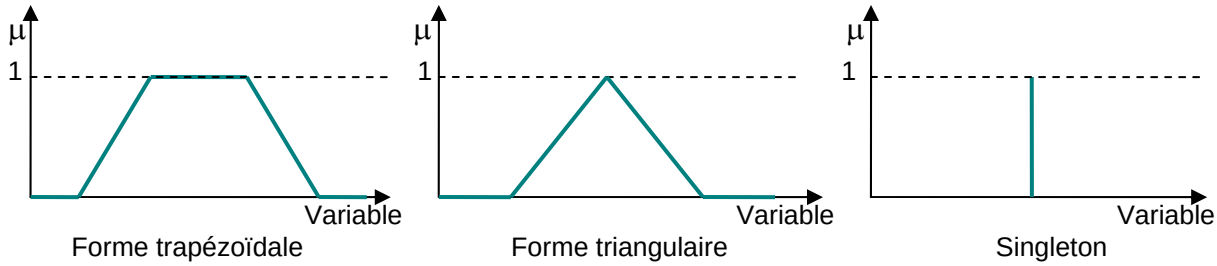


Figure II.4 : Différentes formes des fonctions d'appartenance

Plusieurs ensembles flous peuvent être définis sur la même variable, par exemple les ensembles « taille petite », « taille moyenne » et « taille grande », notions explicitées chacune par une fonction d'appartenance.

La figure II.5 montre la gradualité que permet d'introduire la logique floue. Une personne de 1,75m appartient à l'ensemble « taille grande » avec un degré 0,3 et à l'ensemble « taille moyenne » avec un degré de 0,7. En logique classique, le passage de moyen à grand serait brusque. Une personne de 1,80m serait par exemple de taille moyenne alors qu'une personne de 1,81m serait grande, ce qui choque l'intuition. La variable (par exemple : taille) ainsi que les termes (par exemple : moyenne, grande) définis par les fonctions d'appartenance portent respectivement les noms de variable linguistique et de termes linguistiques. Variables et termes linguistiques peuvent être utilisés directement dans des règles.

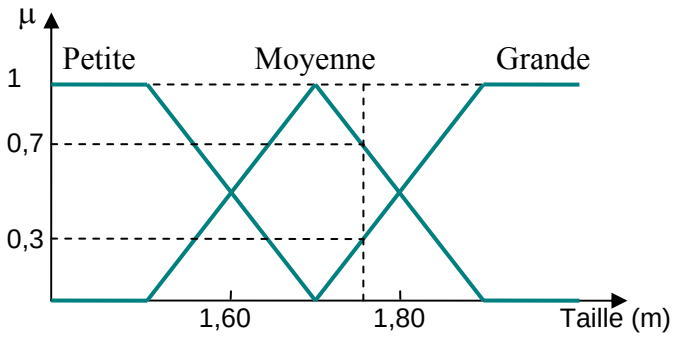


Figure II.5 : Fonctions d'appartenance de la variable Taille

L'opération de fuzzification permet de passer du domaine réel au domaine flou. Elle consiste à déterminer le degré d'appartenance d'une valeur à un ensemble flou. On peut aussi parler de degré de vérité, ces deux notions sont donc similaires. Le degré de vérité d'une proposition A est noté μ (A).

I.1.3. Opérateurs logiques flous

Ces opérateurs permettent d'écrire des combinaisons logiques entre notions floues, c'est-à-dire de faire des calculs sur des degrés de vérité. Comme pour la logique classique, on peut définir des opérateurs ET, OU, négation.

Exemple : Appartement Intéressant = Loyer Raisonnable ET Surface Suffisante

Il existe de nombreuses variantes dans ces opérateurs, cependant, les plus répandus sont ceux dits « de Zadeh » décrits ci-dessous.

● Intersection

L'opérateur logique correspondant à l'intersection d'ensembles est le ET. Le degré de vérité de la proposition « A ET B » est le minimum des degrés de vérité de A et de B :

$$\mu (A \text{ ET } B) = MIN [\mu (A), \mu (B)]$$

Exemple :

« Température Basse » est vraie à 0,7 ; « Pression Faible » est vraie à 0,5
« Température Basse ET Pression Faible » est donc vraie à 0,5 = MIN [0,7; 0,5]

On remarque que l'opérateur ET de la logique classique est bien respecté : 0 ET 1 donne bien 0.

● Union

L'opérateur logique correspondant à l'union d'ensembles est le OU. Le degré de vérité de la proposition « A OU B » est le maximum des degrés de vérité de A et de B :

$$\mu (A \text{ OU } B) = MAX [\mu (A), \mu (B)]$$

Exemple :

« Température Basse » est vraie à 0,7 ; « Pression Faible » est vraie à 0,5
« Température Basse OU Pression Faible » est donc vraie à 0,7 = MAX [0,7; 0,5].

On remarque que l'opérateur OU de la logique classique est bien respecté : 0 OU 1 donne bien 1.

● Complément

L'opérateur logique correspondant au complément d'un ensemble est la négation.

$$\mu (\text{NON } A) = 1 - \mu (A)$$

Exemple :

« Température Basse » est vraie à 0,7
« NON Température Basse», que l'on utilisera généralement sous la forme « Température NON Basse », est donc vraie à 0,3.

Là aussi, on remarque que l'opérateur négation de la logique classique est bien respecté: NON(0) donne bien 1 et NON(1) donne bien 0.

Le tableau II.1 récapitule les opérateurs logiques flous et fait une comparaison avec les opérateurs logiques classiques.

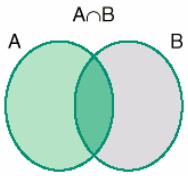
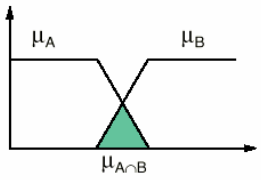
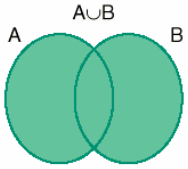
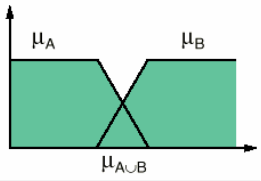
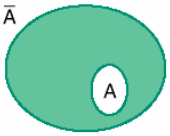
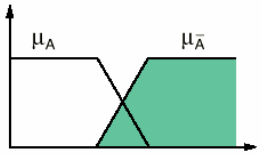
		Opérateur de ZADEH	Opération logique	
Intersection		$\mu_{A\cap B} = MIN(\mu_A, \mu_B)$	**ET**	
Union		$\mu_{A\cup B} = MAX(\mu_A, \mu_B)$	**OU**	
Négation		$\mu_{\bar{A}} = 1 - \mu_A$	**NON**	

Tableau II.1 : Tableau récapitulatif des opérateurs logiques flous

I.1.3. Règles floues

La logique floue a pour objectif de formaliser et de mettre en œuvre la façon de raisonner d'un être humain. En cela, elle peut être classée dans le domaine de l'intelligence artificielle. L'outil le plus utilisé dans les applications de logique floue est la base de règles floues. Une base de règles floues est composée de règles qui sont généralement utilisées en parallèle, mais peuvent également être enchaînées dans certaines applications.

Une règle est du type : SI « prédicat » ALORS « conclusion ».

Par exemple : « Si température élevée et pression forte ALORS ventilation forte et soupape grande ouverte ».

Les bases de règles floues, tout comme les systèmes experts classiques, fonctionnent en s'appuyant sur une base de connaissance issue de l'expertise humaine. Il y a néanmoins de grandes différences dans les caractéristiques et le traitement de cette connaissance.

● Prédicat

Un prédicat (encore appelé prémisse ou condition) est une combinaison de propositions par des opérateurs ET, OU, NON. Les propositions « température élevée » et « pression forte » de l'exemple précédent sont combinées par l'opérateur ET pour former le prédicat de la règle.

● Inférence

Le mécanisme d'inférence le plus couramment utilisé est celui dit « de Mamdani ». Il représente une simplification du mécanisme plus général basé sur « l'implication floue » et le « modus ponens généralisé ». Ces concepts sont explicités dans la référence [1]. Seules les bases de règles « de Mamdani » sont utilisées dans ce qui suit.

● **Conclusion**

La conclusion d'une règle floue est une combinaison de propositions liées par des opérateurs ET. Dans l'exemple précédent, « ventilation forte » et « soupape grande ouverte » sont la conclusion de la règle.

Les bases de règles floues, dans leur cas général, sont définies par des fonctions d'appartenance sur les variables du système, et par des règles qui peuvent être écrites textuellement, où chaque règle fait appel à des entrées et des sorties. Cependant, beaucoup d'applications définissent des tableaux de règles. Dans cette optique, l'espace est quadrillé, et à chaque case correspond une règle. Cette approche a l'avantage d'être systématique, mais elle ne permet pas toujours de traduire simplement (en un minimum de règles) l'expertise existante, et elle n'est applicable que pour deux voire trois entrées, alors que des bases de règles « libres » peuvent être bâties avec un nombre important de variables

I.1.4. Mécanisme d'inférence de Mamdani

Une base de règles floues de Mamdani comprend des règles linguistiques faisant appel à des fonctions d'appartenance pour décrire les concepts utilisés. Le mécanisme d'inférence comprend les étapes suivantes :

I.1.4.1. Fuzzification

La fuzzification consiste à évaluer les fonctions d'appartenance utilisées dans les prédicats des règles. La figure II.6 illustre ceci par nu exemple.

Exemple : Si «pression forte» ET «température élevée» ALORS «ouverture vanne grande»

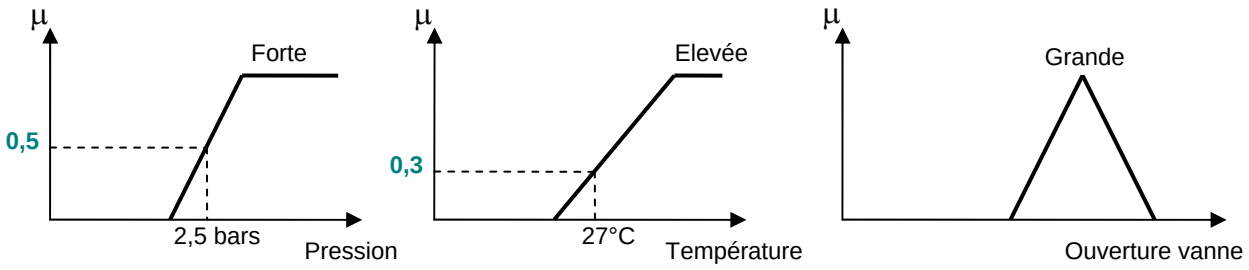


Figure II.6 : Fuzzification

I.1.4.2. Degré d'activation et implication

Le degré d'activation d'une règle est l'évaluation du prédicat de chaque règle par combinaison logique des propositions du prédicat. Le « ET » est réalisé en effectuant le minimum entre les degrés de vérité des propositions.

Le degré d'activation de la règle permet de déterminer la conclusion de la règle, c'est l'implication. Il existe plusieurs opérateurs d'implication, mais le plus utilisé est le « minimum ». L'ensemble flou de conclusion est construit en réalisant le minimum entre le degré d'activation et la fonction d'appartenance, sorte d'«écrêtage» de la fonction d'appartenance de conclusion.

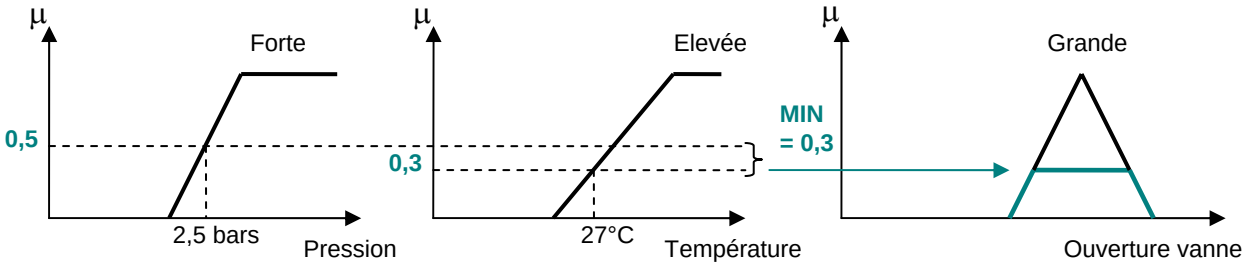


Figure II.7 : Implication

I.1.4.3. Agrégation

L'ensemble flou global de sortie est construit par agrégation des ensembles flous obtenus par chacune des règles concernant cette sortie. L'exemple suivant présente le cas où deux règles agissent sur une sortie. On considère que les règles sont liées par un « OU » logique, et on calcule donc le maximum entre les fonctions d'appartenance résultantes pour chaque règle.

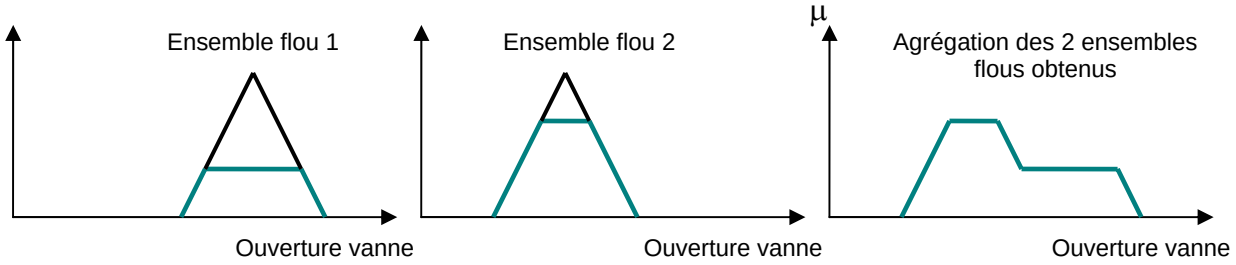


Figure II.8 : Agrégation

I.1.4.4. Défuzzification

A la fin de l'inférence, l'ensemble flou de sortie est déterminé mais il n'est pas directement utilisable pour donner une information précise à l'opérateur ou commander un actionneur. Il est nécessaire de passer du « monde flou » au « monde réel », c'est la défuzzification. Pour réaliser cette opération il existe plusieurs méthodes, la plus souvent rencontrée étant le calcul du « centre de gravité » de l'ensemble flou (figure I.9)

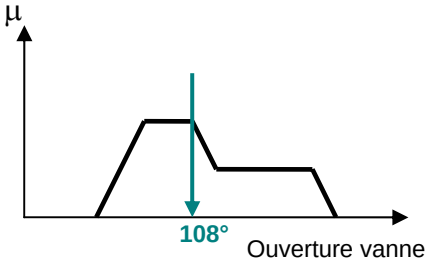


Figure II.9 : Défuzzification par centre de gravité

La formule qui permet d'obtenir le centre de gravité à partir de l'ensemble flou de sortie est la suivante :

$$x_0 = \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx}$$

II. Les Algorithmes Génétiques

Les Algorithmes Génétiques (AGs) appartiennent à la famille des Algorithmes Evolutionnaires qui sont inspirés du concept de sélection naturelle élaboré par Charles Darwin en 1859. Ils ont vu le jour dans les années 60 et ont commencé à sortir de leur isolement dans les années 90. Leur principe est de simuler l'évolution d'une population d'individus divers auxquels on applique différents opérateurs génétiques (croisement, mutations …) et que l'on soumet à chaque génération à une sélection. Ces algorithmes sont de plus en plus utilisés dans l'industrie car ils sont particulièrement adaptés aux problèmes d'optimisation comportant de nombreux paramètres.

Par rapport aux méthodes habituelles que l'on peut qualifier d' « analytiques », les AGs peuvent être qualifiés de « synthétiques ». Ils peuvent parfois synthétiser des solutions nouvelles et originales à des problèmes connus car ils expérimentent sans idées préconçues, si ce n'est les paramètres et l'espace de recherche qu'on leur impose.

Les AGs permettent d'obtenir des solutions à un problème n'ayant pas de méthode de résolution décrite précisément, ou dont la solution exacte, si elle est connue, est trop compliquée pour être calculée en un temps raisonnable. Dans ce cadre, on peut citer quelques problèmes complexes bien connus : optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), traitement d'images (alignement des images satellitaires, reconnaissance de formes…), contrôle de système industriels ou apprentissage des réseaux de neurones. Les AGs sont également utilisés pour optimiser des réseaux (câbles, fibres optiques, mais aussi eau, gaz…), des circuits VLSI, des antennes, des emplois du temps, des designs …etc. Ils peuvent aussi être utilisés pour contrôler un système évoluant dans le temps (chaîne de production, centrale nucléaire) car la population peut s'adapter à des conditions changeantes, ils supportent bien l'existence de bruit dans la fonction à optimiser.

II.1. Fonctionnement des Algorithmes Génétiques

Dans cette partie, on va expliquer brièvement le fonctionnement des Algorithmes Génétiques. La figure suivante donne l'organigramme simplifié d'un AG [6].

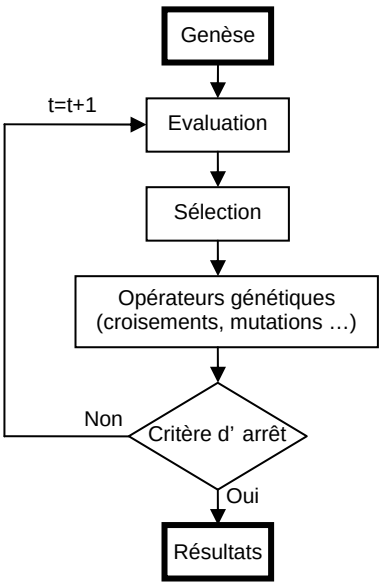


Figure II.10 : Organigramme d'un Algorithme Génétique

II.1.1. Codage des variables

La première étape est de définir et de coder convenablement le problème. A chaque variable d'optimisation x_i (à chaque paramètre du dispositif), nous faisons correspondre un *gène*. Nous appelons *chromosome* un ensemble de gènes. Chaque dispositif est représenté par un *individu* doté d'un génotype constitué d'un ou plusieurs chromosomes. Nous appelons *population* un ensemble de *N* individus que nous allons faire évoluer.

D'un point de vue informatique, nous utilisons dans notre algorithme un codage binaire. C'est-à-dire qu'un gène est un entier long (32 bits). Un chromosome est un tableau de gènes (figure II.11), un individu est un tableau de chromosomes et la population est un tableau d'individus. On aboutit à une structure présentant cinq niveaux d'organisation (figure II.12), d'où résulte le comportement complexe des AGs.

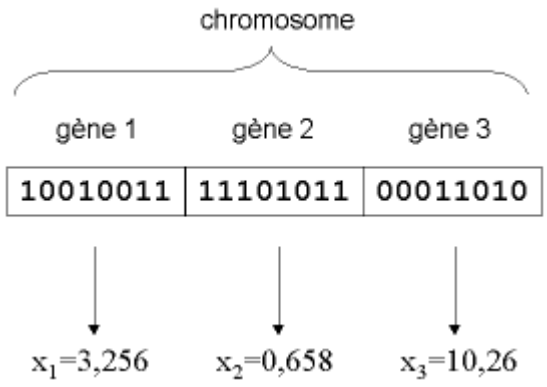


Figure II.11 : Illustration schématique du codage des variables d'optimisation x_i

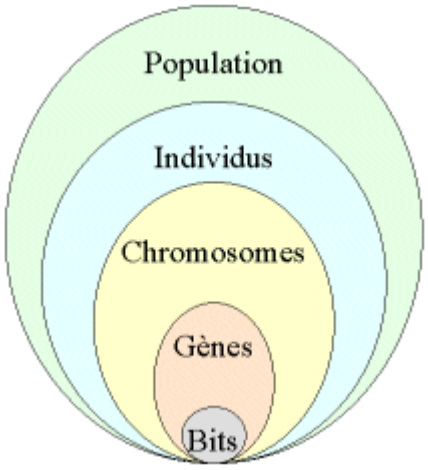


Figure II.12 : les cinq niveaux d'organisation de l'Algorithme Génétique

Un des avantages du codage binaire est que l'on peut ainsi facilement coder toutes sortes d'objets : des réels, des entiers, des valeurs booléennes ou des chaînes de caractères. Rappelons que dans cette étude les *n* variables sont supposées réelles. Nous considérons un espace de recherche fini :

$$x_{i\min} \le x_i \le x_{i\max} \qquad \forall i \in [1;n]$$

Afin de coder les variables réelles en binaire, il faut discrétiser l'espace de recherche. Ainsi un codage sur 32 bits implique une discrétisation des intervalles en $g_{max}=2^{32}-1$ = 4 294 967 295 valeurs discrètes.

A chaque variable réelle x_i on associe donc un entier long g_i :

$$0 \le g_i \le g_{\max} \qquad \forall i \in [1,n]$$

Les formules de codage et décodage sont alors les suivantes :

$$g_i = \frac{x_i - x_{i\min}}{x_{i\max} - x_{i\min}} . g_{\max}$$

$$x_i = x_{i\min} + (x_{i\max} - x_{i\min}) . \frac{g_i}{g_{\max}}$$

II.1.2. Genèse de la population

La première étape de l'algorithme est la genèse de la population, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer. On pourrait prendre des individus régulièrement répartis dans l'espace. Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs g_i des gènes sont tirées au hasard selon une distribution uniforme. Notons qu'on peut, si nécessaire, introduire des individus déjà calculés. Nous discuterons plus loin de la taille N de cette population, mais nous pouvons déjà dire qu'elle résultera d'un compromis entre temps de calcul et qualité de la solution.

II.1.3. Evaluation

L'évaluation de chaque dispositif est réalisée par le modèle utilisé. Les résultats obtenus sont alors utilisés pour calculer les fonctions objectif et la fonction d'adaptation. Notons que dans le cas d'un modèle physique, la majeure partie du temps de calcul sera probablement due à l'exécution de ce modèle. En effet, le reste de l'AG est essentiellement composé de manipulation d'entiers et de bits, donc très rapide.

II.1.4. Séléction-élimination

Nous appelons *génération* la population à un instant t donné. Une fois réalisée l'évaluation de la génération, on opère une sélection à partir de la fonction d'adaptation. Seuls les individus passant l'épreuve de sélection peuvent accéder à la *génération intermédiaire* (*mating pool*) et s'y reproduire. En fait, cette génération intermédiaire est deux fois plus petite (N/2 dispositifs) que la génération dont elle est issue. Notre algorithme étant conçu de façon à ce que chaque couple d'individus parents donne naissance à deux enfants, nous aboutissons à nouveau à une génération entière à l'instant t+1. Deux techniques de sélection sont utilisées :

● ***N*/2-élitisme :** Les individus sont triés selon leur fonction d'adaptation. Seule la moitié supérieure de la population, correspondant aux meilleurs composants, est sélectionnée. On constate que cette méthode induit une convergence prématurée de l'algorithme. Il est en effet nécessaire de maintenir une diversité génétique suffisante dans la population, celle-ci constituant un réservoir de gènes pouvant être utiles par la suite.

● **Sélection par tournoi :** deux individus sont choisis au hasard et combattent (on compare leurs fonctions d'adaptation) pour accéder à la génération intermédiaire. Le plus adapté l'emporte avec une probabilité $0{,}5 < p \leq 1$, qui est généralement prise égale à 1 (une valeur inférieure permet de réduire la pression de sélection si nécessaire). Cette étape est répétée jusqu'à ce que la génération intermédiaire soit remplie (N/2 composants).

II.2. Les opérateurs génétiques

L'algorithme génétique réalise l'optimisation par la manipulation d'une population de chromosomes. À chaque génération, l'AG crée un ensemble de nouveaux chromosomes au moyen de diverses opérations appelées opérateurs génétiques :

II.2.1. Opérateur croisement

Une fois la génération intermédiaire remplie, les individus sont aléatoirement répartis en couples. Les chromosomes des parents sont alors copiés et recombinés de façon à former deux descendants de la génération t+1 possédant des caractéristiques issues des deux parents.

L'opérateur croisement favorise l'exploration de l'espace de recherche en assurant le brassage du matériel génétique et l'accumulation des mutations favorables. Il permet de créer de nouvelles combinaisons des paramètres des composants.

Le phénomène de croisement est une propriété naturelle de l'ADN. C'est par analogie qu'ont été conçus les opérateurs de croisement dans les AGs. On utilise surtout deux méthodes de croisement classiques :

● **Croisement en un point :** on choisit au hasard un point de croisement pour chaque couple (figure II.13). Le croisement s'effectue directement au niveau binaire, et non pas au niveau des gènes. Un chromosome peut donc être coupé au milieu d'un gène.

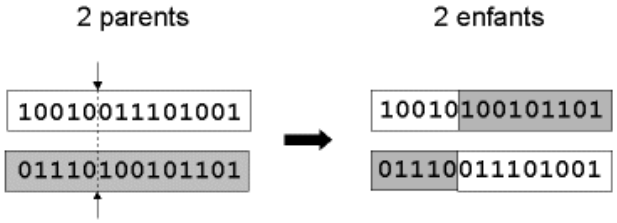


Figure II.13 : Représentation schématique du croisement en 1 point

● **Croisement un deux points :** on choisit au hasard deux points de croisement (figure II.14). Cet opérateur est généralement considéré comme plus efficace que le précédent.

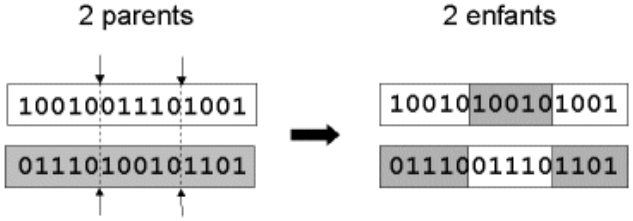


Figure II.14 : Représentation schématique du croisement en 2 points

II.2.2. Opérateur mutation

Nous définissons une *mutation* comme étant l'inversion d'un bit dans un chromosome (figure II.15). Cela revient à modifier aléatoirement la valeur d'un paramètre du dispositif. Les mutations jouent le rôle de bruit et empêchent l'évolution de se figer. Elles permettent d'assurer une recherche aussi bien globale que locale, selon le poids et le nombre des bits mutés. De plus, elles garantissent mathématiquement que l'optimum global peut être atteint.

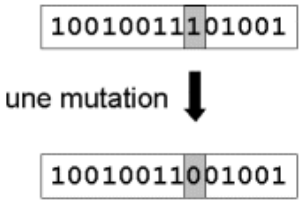


Figure II.15 : Représentation schématique d'une mutation dans un chromosome

D'autre part, une population trop petite peut s'homogénéiser à cause des erreurs stochastiques : les gènes favorisés par le hasard peuvent se répandre au détriment des autres. Cet autre mécanisme de l'évolution, qui existe même en l'absence de sélection, est connu sous le nom de *dérive génétique*. Du point de vue du dispositif, cela signifie que l'on risque alors d'aboutir à des dispositifs qui ne seront pas forcément optimaux. Les mutations permettent de contrebalancer cet effet en introduisant constamment de nouveaux gènes dans la population.

De nombreuses méthodes existent réaliser un opérateur mutation. Souvent la probabilité de mutation p_m par bit et par génération est fixée entre 0,001 et 0,01. On peut prendre également $p_m=1/l$ où l est le nombre de bits composant un chromosome. Il est possible d'associer une probabilité différente à chaque gène, et ces probabilités peuvent être fixes ou évoluer dans le temps.

II.3. Convergence de l'Algorithme Génétique

II.3.1. Convergence et temps de calcul

On constate que l'amélioration de la population est très rapide au début de l'exécution de l'algorithme *(recherche globale)* et devient de plus en plus lente à mesure que le temps passe *(recherche locale)*. On voit aussi que la valeur moyenne de la fonction d'adaptation a tendance à se rapprocher de celle de l'individu le plus adapté. Cela correspond à une uniformisation croissante de la population. L'utilisateur peut donc stopper l'algorithme dès qu'il voit que la population s'est à peu près uniformisée.

Un des intérêts des AGs est que le temps de calcul ne croît pas exponentiellement avec le nombre n de variables. D'autre part, ce temps de calcul est proportionnel au temps de calcul de la fonction d'adaptation, donc du modèle utilisé, et à la taille de la population.

II.3.2. Réglage des paramètres de l'AG

Nous abordons ici le délicat problème du réglage des paramètres de l'algorithme. Celui-ci doit être optimisé pour chaque type de problème traité, ce qui constitue une part importante du travail de l'utilisateur. L'ensemble problème-méthodes-paramètres constitue donc un tout. En témoigne certaines études où les paramètres d'un AG sont réglés et optimisés par un autre AG. Dans la pratique, les méthodes et paramètres des AGs sont tout d'abord réglés approximativement par tâtonnement avec des fonctions de n variables couramment utilisées pour tester les algorithmes d'optimisation (fonction sphérique, fonction fractale …). Le temps de calcul de ces fonctions étant minime, on peut ainsi régler rapidement les paramètres.

● Taille de la population :

- Trop faible : l'AG n'a pas assez d'échantillons de l'espace de recherche ;
- Élevée : l'AG est plus uniforme et prévenu contre la convergence prématurée dite aussi *stagnation locale* ou *dérive génétique* ;
- Trop élevée : le nombre élevé d'évaluations de la fonction d'adaptation par génération ralentit la convergence.

La taille de la population doit être choisie de façon à réaliser un bon compromis entre temps de calcul et qualité du résultat. Mais il faut être conscient que cette taille de population dépend de la puissance de calcul dont on dispose, des méthodes utilisées (sélection, opérateurs génétiques…), du nombre de variables considérées et de la fonction d'adaptation.

● Taux de croisement :

- Trop élevé : les bonnes structures risquent d'être cassées trop vite par rapport à l'amélioration que peut apporter la sélection ;
- Trop faible : la recherche risque de stagner à cause du faible taux d'exploration ;
- Le taux habituel est choisi entre 60% et 100%.

● Taux de mutation :

- Trop élevé : la recherche devient trop aléatoire.
- Trop faible : la recherche risque de stagner à cause du faible taux d'exploration.

Si la taille de la population est faible, un taux de croisement faible doit être combiné avec un taux de mutation élevé. Ces observations restent valables pour les fonctions continues sans contraintes avec codage binaire.

III. Méthodes MPPT

Il existe plusieurs méthodes de poursuite du point de puissance maximale. Les références [1] et [2] expliquent en détail un grand nombre de ces méthodes. On en donne dans ce qui suit un bref aperçu :

- Adaptation manuelle : Elle consiste à adapter manuellement la charge au générateur photovoltaïque, c'est-à-dire choisir la charge suivant la valeur du courant et de la tension obtenus expérimentalement pour des conditions normales de fonctionnement ;
- Méthodes à contre réaction de tension : la tension de référence peut être fixe, variable en fonction de la tension à circuit ouvert V_{oc}, ou externe par l'utilisation d'une cellule pilote ;
- Méthodes à contre réaction de courant : là aussi le courant peut être variable en fonction du courant de court circuit I_{sc}, ou externe par l'utilisation d'une cellule pilote ;
- Méthodes à contre réaction de puissance : Plusieurs algorithmes sont utilisés comme l'algorithme « Perturbation et Observation » (P&O) et l'algorithme « Incremental Conductance » ;
- Méthode de poursuite analogique ;
- Utilisation d'un contrôleur flou.

Comme on l'a déjà cité plus haut, les deux méthodes qui nous intéressent sont la méthode Perturbation et Observation (P&O), et la méthode du contrôleur flou optimisé par les AGs. Nous allons expliquer dans ce qui suit le principe de chacune d'elles.

III.1. Méthode Perturbation et Observation (P&O)

C'est une méthode à contre réaction de puissance, et c'est la méthode MPPT la plus utilisée vu la simplicité de son principe et la facilité de son implémentation. Elle est basée, comme son nom l'indique, sur l'introduction d'une perturbation sur le système et l'observation de son effet sur la puissance. La perturbation du système est introduite par l'augmentation ou la diminution de la tension de référence (V_{ref}) en agissant directement sur le rapport cyclique du convertisseur DC-DC. L'observation de la puissance permet ensuite de prendre une décision sur la prochaine perturbation à introduire, s'il y'a eu une augmentation de la puissance, la perturbation continuera dans le même sens, sinon elle est inversée.

L'algorithme de cette méthode est donné dans la figure II.16. On explique dans ce qui suit son fonctionnement brièvement :

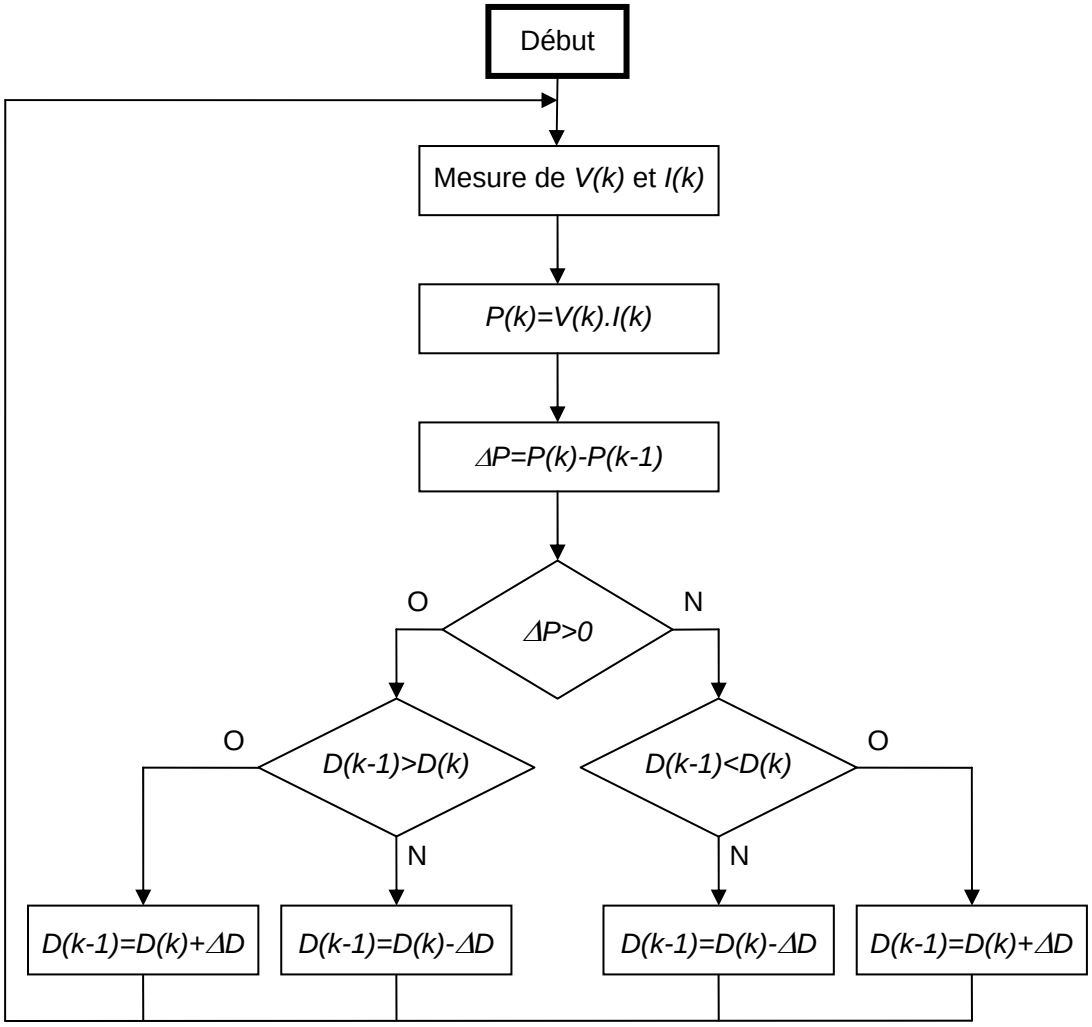


Figure II.16 : Organigramme de l'algorithme P&O

Tout d'abord, une mesure du courant I et de la tension V est effectuée pour calculer la puissance à l'instant actuel $P(k)$, cette valeur est ensuite comparée à celle de l'instant passé $P(k-1)$. La différence ΔP permet de déterminer le sens de variation de la puissance, si celle-ci est positive la prochaine perturbation doit suivre le même sens que la précédente, sinon elle doit être dans le sens opposé. Pour cela un test est effectué sur $\Delta V=V(k)-V(k-1)$ et la décision est alors prise sur la prochaine valeur du rapport cyclique, soit par son incrémentation ($D(k+1)=D(k)+\Delta D$), ou au contraire par sa décrémentation ($D(k+1)=D(k)-\Delta D$).

Par exemple, si ΔP est positive alors la puissance a augmenté au cours du dernier cycle, et si en plus ΔV est négative, cela veut dire qu'il y'a eu une décrémentation du rapport cyclique durant le dernier cycle, la décision à prendre dans ce cas sera alors une autre décrémentation du rapport cyclique : $D(k+1)=D(k)-\Delta D$.

On doit noter qu'à chaque cycle d'horloge une perturbation est introduite, et une fois le MPP atteint, l'algorithme ne s'arrêtera pas et V continuera d'osciller autours de la tension de fonctionnement idéal V_{mp}. Ces oscillations causent des pertes d'énergie et leur amplitude est proportionnelle au pas de perturbation ΔD.

Si la largeur du pas ΔD est grande, l'algorithme convergera rapidement vers le MPP, surtout pour des changements brusques des conditions de fonctionnement, mais l'amplitude des oscillations autours de ce point seront importantes. Par contre si ce pas est trop petit l'amplitude des oscillations sera faible mais l'algorithme aura une réponse très lente, ce qui diminuera son efficacité pour des variations rapides de l'insolation et de la température.

Le choix idéal de ΔD est donc un compromis à faire entre la rapidité de la réponse de l'algorithme et les pertes de puissance générées par les oscillations autour de l'état stable. Cette valeur est déterminée la plupart du temps expérimentalement ou par la simulation.

Un des inconvénients de la méthode P&O est que son algorithme ne répond pas instantanément aux variations de puissance dues aux changements brusques de l'ensoleillement. En effet, une augmentation brutale de l'insolation produit une augmentation de la puissance, l'algorithme considérera cette augmentation comme étant due à la perturbation introduite durant le dernier cycle, il continuera alors dans la même direction qui peut être la mauvaise direction, ce qui l'éloigne du point MPP. L'algorithme ne commencera à rechercher normalement le point MPP qu'une fois l'état de stabilité de l'ensoleillement atteint. Ce phénomène cause un retard de la réponse de l'algorithme et produit des pertes de puissance, il représente le plus grand inconvénient de la méthode P&O.

III.2. Méthode du contrôleur flou optimisé par les AGs

Cette méthode utilise la logique floue afin de rendre plus rapide la réponse du contrôleur et d'augmenter la stabilité du système une fois le point MPP atteint. Les fonctions d'appartenance des variables d'entrée et de sortie sont optimisés par les Algorithmes Génétiques afin d'améliorer les qualités du contrôleur.

La poursuite du point MPP sera divisée en deux phases : la première sera la phase de recherche rude, avec un pas de recherche important pour améliorer la réponse du contrôleur MPPT, la seconde sera une phase fine où le pas sera très petit, ce qui garantira une stabilité du système et la diminution au maximum des oscillations autours du point MPP. Cette caractéristique du contrôleur flou illustre son efficacité et le place parmi les meilleurs dispositifs de poursuite MPPT.

Le contrôleur flou se compose de trois blocs : l'opération de fuzzification des variables d'entrée est effectué dans le premier bloc, elle permet le passage du domaine réel au domaine flou, le deuxième bloc est consacré aux règles d'inférence, alors que le dernier bloc est le siège de l'opération de défuzzification permettant le retour vers le domaine réel. Cette dernière opération utilise la méthode du centre de masse pour déterminer la valeur de la sortie. La figure II.17 donne la structure de base du contrôleur flou utilisé.

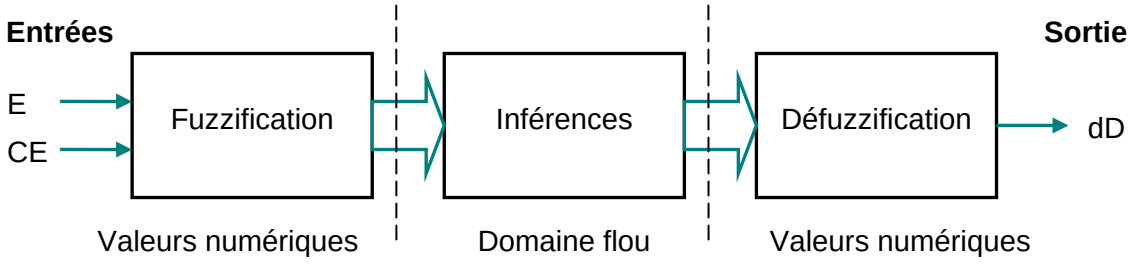


Figure II.17 : Structure de base du contrôleur flou MPPT

III.2.1. Fuzzification

Tout d'abord, les valeurs instantanées du courant et de la tension de sortie du générateur photovoltaïque sont mesurées par un convertisseur analogique numérique (CAN), la puissance à l'instant actuel est obtenue par la multiplication de ces deux grandeurs :

$$P(k) = I(k).V(k)$$

Les variables d'entrées sont l'erreur sur la puissance (E) et la dynamique de l'erreur (CE), on donne leurs expressions respectives :

$$E(k) = \frac{P(k) - P(k-1)}{V(k) - V(k-1)}$$

$$CE(k) = E(k) - E(k-1)$$

La valeur de l'entrée *E(k)* nous indique de quel coté est situé le point de fonctionnement par rapport au point de puissance maximale MPP. Si cette valeur est positive, cela veut dire que le point de fonctionnement se trouve à gauche du point MPP, si par contre elle est négative alors le point de fonctionnement est à droite du point MPP.

L'entrée *CE(k)* nous permet d'estimer le degré d'éloignement entre le point de fonctionnement et le point MPP. Si sa valeur est petite, cela veut dire que le point de fonctionnement est encore loin du point MPP, si par contre cette valeur est élevée cela indique que le point de fonctionnement est très proche du point MPP.

Ces deux entrées nous permettent de prendre la décision appropriée sur la valeur de la sortie *dD* du pas du rapport cyclique. Comme cité au chapitre I, augmenter le rapport cyclique du convertisseur DC-DC permet d'augmenter la tension du point de fonctionnement et inversement.

On applique la théorie des ensembles flous aux entrées-sorties, et on exprime chacune d'entre elles par cinq fonctions d'appartenance qui sont :

NG : négatif grand,
NP : négatif petit,
ZE : zéro,
PP : positif petit,
PG : positif grand.

C'est là qu'intervient la phase d'optimisation par les Algorithmes Génétiques. En effet, comme détaillé dans la référence [5], on a appliqué l'AG sur les valeurs délimitant ces fonctions d'appartenance en utilisant les paramètres suivants :

- Nombre de gènes = 12 (4 gènes par variable floue, les extrémités de l'intervalle de recherche et le zéro étant fixes)
- Nombre d'individus = 100
- Nombre de générations = 20
- Taux de croisement = 100%
- Taux de mutation = 1/12 = 8,33%

- Espace de recherche : $E \in [-32,32]$; $CE \in [-100,100]$; $dD \in [-0.64,0.64]$
- Indice de distribution = 5 (cet indice permet de fixer le degré de ressemblance entre les parents et les enfants)
- Technique de sélection : *N*/2-élitisme
- Critère de sélection : minimisation de l'erreur quadratique sur la puissance, cette erreur est donnée par la formule suivante :

$$\int (P_{\max} \_ P_{désirée})^2 dt$$

L'exécution de l'AG a été effectuée sur un PC et a duré quelques jours. La figure II.18 illustre la forme des fonctions d'appartenance des variables d'entrée et de sortie obtenues après optimisation par l'AG. On remarque que ces fonctions d'appartenance ne sont pas uniformément réparties.

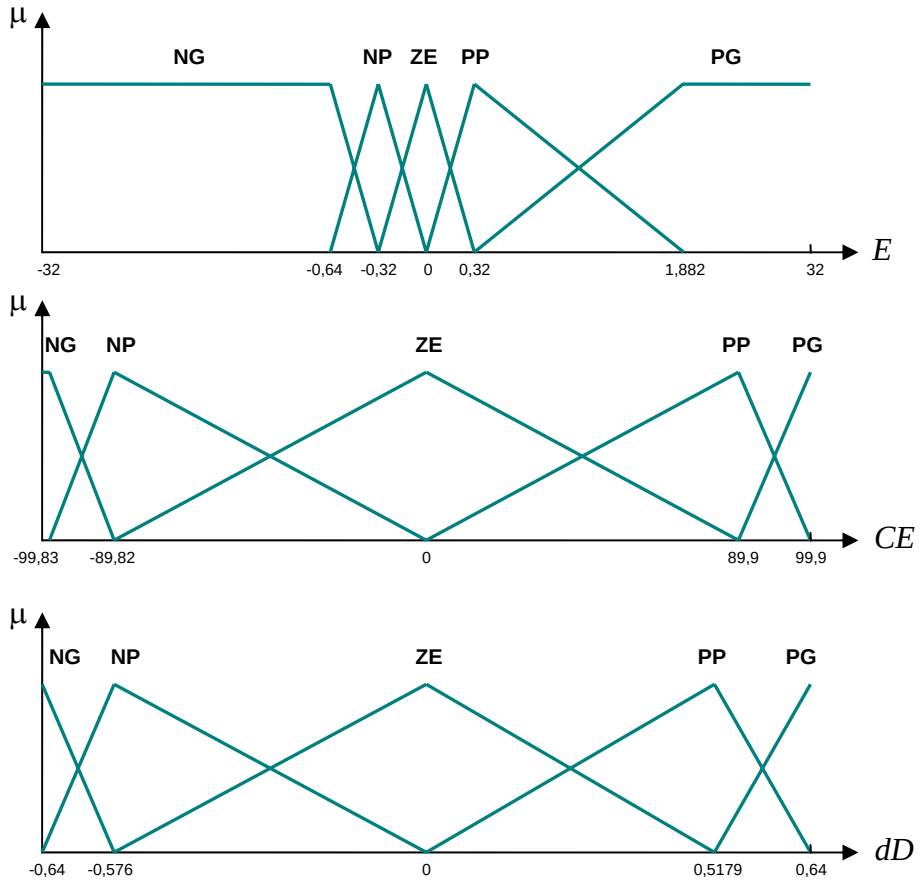


Figure II.18 : Fonctions d'appartenance optimisées des variables d'entrée et de sortie

III.2.2. Inférences

La méthode d'inférence utilisée pour ce contrôleur est celle de « Mamdani », c'est généralement la méthode la plus utilisée. Elle utilise l'opérateur MIN pour le « ET » et l'opérateur MAX pour le « OU ».

Les règles d'inférence permettent de prendre la bonne décision sur la sortie *dD* à partir des valeurs des entrées *E* et *CE*. Le tableau II.3 résume les règles d'inférences du contrôleur flou MPPT utilisé [2].

E \ CE	NG	NP	ZE	PP	PG
NG	ZE	ZE	PG	PG	PG
NP	ZE	ZE	PP	PP	PP
ZE	PP	ZE	ZE	ZE	NP
PP	NP	NP	NP	ZE	ZE
PG	NG	NG	NG	ZE	ZE

Tableau II.2 : Table d'inférences du contrôleur MPPT flou optimisé

On prend comme exemple la règle suivante : Si ***E*** est **NG** et ***CE*** est **PP** alors ***dD*** est **PG**.

On explique dans ce qui suit cette règle : si E(k) est négatif grand (NG) et CE(k) est positif petit (PP), cela veut dire que le point de fonctionnement est à droite du point MPP, et en plus il est éloigné de ce dernier. La décision à prendre sera alors d'incrémenter le rapport cyclique d'un grand pas, c'est-à-dire dD sera positif grand (PG).

Les règles sont liées entre elles par un « OU » logique, et on calcule donc le maximum entre les fonctions d'appartenance résultantes pour chaque règle. L'ensemble flou global de la sortie *dD* est construit par agrégation des ensembles flous obtenus par chacune de ces règles.

III.2.3. Défuzzification

A partir de l'ensemble flou obtenu par les règles d'inférences, on doit obtenir une seule valeur de la variable de commande *dD*. L'opération de défuzzification est réalisée par la méthode du centroïde (Figure II.19). Elle permet de calculer le centre de gravité de la surface obtenue par l'expression suivante :

$$dD_0 = \frac{\sum_{i=1}^{n} D_i . \mu(D_i)}{\sum_{i=1}^{n} \mu(D_i)}$$

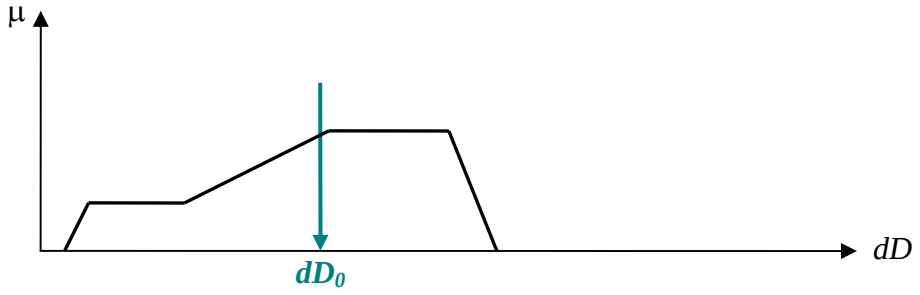


Figure II.19 : Défuzzification de la sortie dD par la méthode du centre de masse

Chapitre III

Développement d'un projet sur circuit FPGA

Chapitre III

Développement d'un projet sur circuit FPGA

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 de défense. 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 les méthodes MPPT déjà cités dans le chapitre II sur un circuit FPGA. La programmation a été faite en utilisant le langage de description VHDL. Nous allons d'abord faire une description des circuits FPGA, ensuite on va introduire le langage VHDL, et on terminera par donner les étapes nécessaires au développement d'un projet sur un circuit FPGA, de la programmation jusqu'au chargement sur la carte.

I. 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 III.1 donne une classification possible des circuits numériques en précisant où se situent les circuits FPGA dans cette classification.

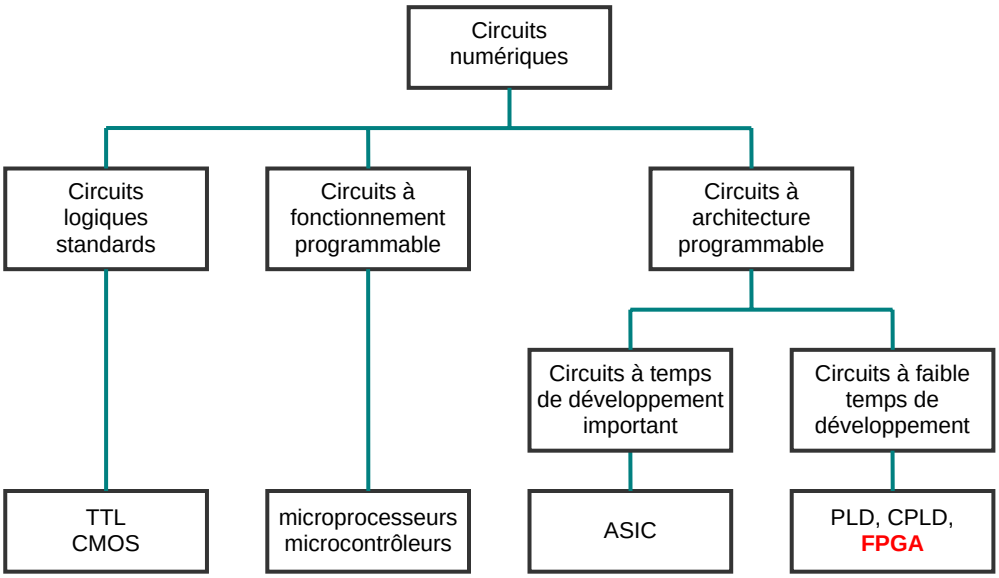


Figure III.1 : Classification des circuits numériques

Les FPGA sont utilisés dans 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.

I.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 qu'on va implémenter nos programmes.

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 III.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.

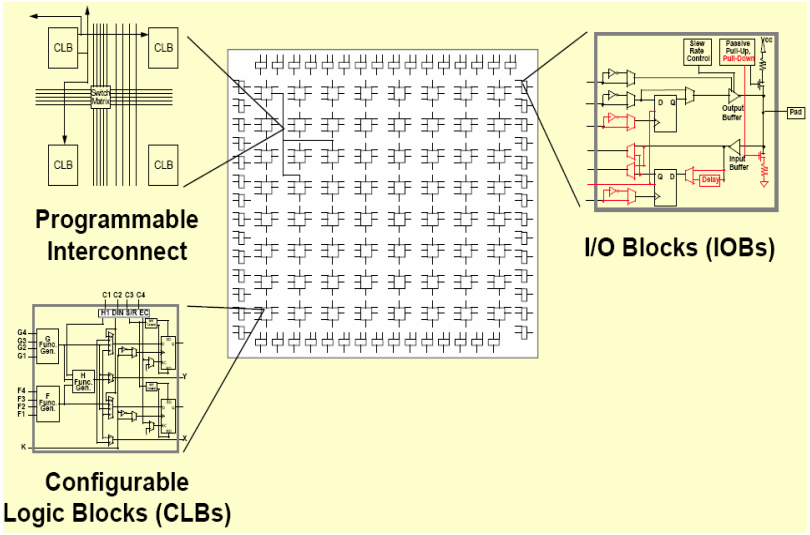


Figure III.2 : Architecture interne du FPGA

● 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 III.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.

La mise au point d'une configuration s'effectue en deux temps : Une première étape purement logicielle va consister à dessiner puis simuler logiquement le circuit fini. Dans la seconde étape, on effectuera une simulation matérielle en configurant un circuit réel. On pourra alors vérifier si le fonctionnement réel correspond bien à l'attente du concepteur, et si besoin est identifier les anomalies liées généralement à des temps de transit réels légèrement différents de ceux supposés lors de la simulation logicielle, ce qui peut conduire à des états instables voire même erronés.

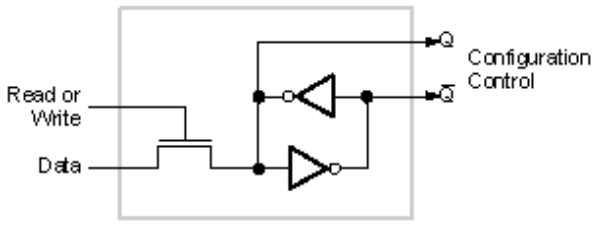


Figure III.3 : Structure d'un cellule SRAM

Les circuits FPGA du fabricant Xilinx utilisent deux types de cellules de base : les cellules d'entrées/sorties appelés IOB (Input Output Bloc), et les cellules logiques appelées CLB (Configurable Logic Bloc). Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions configurable. On décrit dans ce qui suit chacun de ces composants.

I.1.1. Les CLB (Configurable Logic Bloc)

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

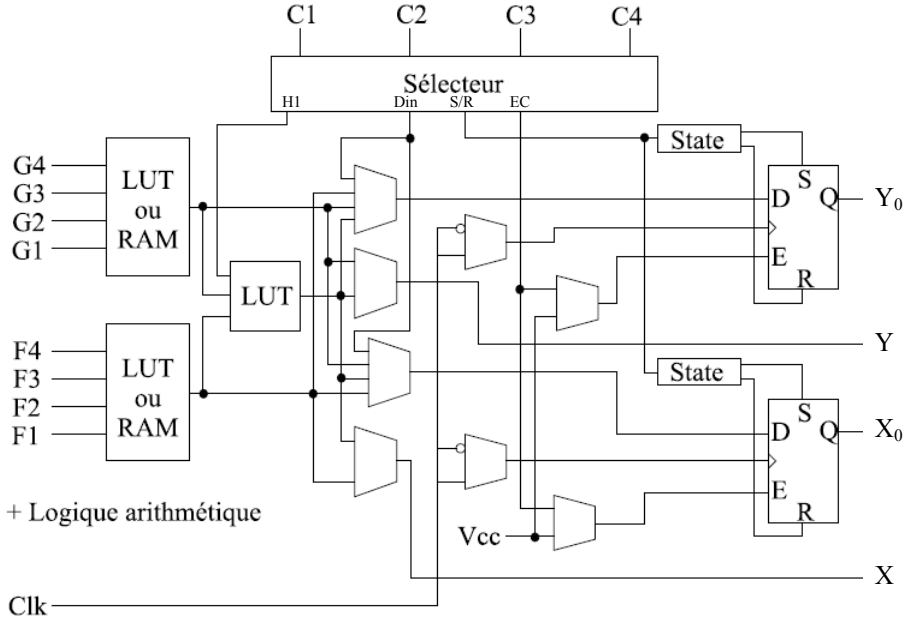


Figure III.4 : Schéma d'une cellule logique (XC4000 de Xilinx)

Dans cette famille, la cellule de base contient deux LUT à 4 entrées qui peuvent réaliser deux fonctions quelconques à 4 entrées. Une troisième LUT peut réaliser une fonction quelconque à 3 entrées à partir des sorties des deux premières LUT (F' et G' qui deviennent H2 et H3) et d'une troisième variable d'entrée H1 sortant du bloc « sélecteur ». Le bloc sélecteur contient 4 signaux de contrôle : 3 signaux dédiées pour les registres : une donnée "Din", un signal de validation "Ec" et une remise à un ou à zéro asynchrone "S/R", et le 4ème signal représente l'entrée H1 de la LUT à 3 entrées.

Les signaux des générateurs de fonction peuvent sortir du CLB, soit par la sortie X pour les fonctions F' et G', soit Y pour les fonctions G' et H'. Ainsi un CLB peut être utilisé pour réaliser :

- deux fonctions indépendantes à 4 entrées indépendantes ;
- ou une seule fonction à 5 variables ;
- ou deux fonctions, une à 4 variables et une autre à 5 variables.

Les sorties de ces blocs logiques peuvent être appliquées à des bascules au nombre de deux ou directement à la sortie du CLB (sorties X et Y). Chaque bascule présente deux modes de fonctionnement : un mode « flip-flop » avec comme donnée à mémoriser, soit l'une des

fonctions F', G', H' ; soit l'entrée directe Din. La donnée peut être mémorisée sur un front montant ou descendant de l'horloge (Clk). Les sorties de ces deux bascules correspondent aux sorties du CLB X_0 et Y_0.

Un mode dit de « verrouillage » exploite l'entrée S/R qui peut être programmée soit en mode SET, mise à 1 de la bascule, soit en Reset, mise à zéro de la bascule. Ces deux entrées coexistent avec une autre entrée, qui n'est pas représentée sur la figure III.4, et qui est appelée le global Set/Reset. Cette entrée initialise le circuit FPGA à chaque mise sous tension, à chaque configuration, en commandant toutes les bascules au même instant soit à '1', soit à '0'. Elle agit également lors d'un niveau actif sur le fil RESET lequel peut être connecté à n'importe quelle entrée du circuit FPGA.

L'idée de cette architecture consiste à pouvoir modifier le contenu des mémoires des LUT en cours de fonctionnement. En effet, les LUT ne sont rien d'autre que des petites RAM qui étaient configurées au démarrage ; on peut alors les utiliser comme des petites mémoires de 16x1 bits. Un mode optionnel des CLB est donc la configuration en mémoire RAM de 16x2 bits ou 32x1 bit. Les entrées F1 à F4 et G1 à G4 deviennent des lignes d'adresses sélectionnant une cellule mémoire particulière. La fonctionnalité des signaux de contrôle est modifiée dans cette configuration : les lignes H1, Din et S/R deviennent respectivement les deux données D_0 et D_1 d'entrée (RAM 16x2bits) et le signal de validation d'écriture WE. Le contenu de la cellule mémoire (D_0 et D_1) est accessible aux sorties des générateurs de fonctions F' et G'. Ces données peuvent sortir du CLB à travers ses sorties X et Y ou alors en passant par les deux bascules.

L'intégration de fonctions à nombreuses variables diminue le nombre de CLB nécessaires et les délais de propagation des signaux ; par conséquent, elle augmente la densité et la vitesse du circuit. Le plus large circuit de cette famille (le circuit XC40250) dispose d'un réseau de 92x92 cellules de base et est équivalent à environ 250.000 portes logiques.

Xilinx propose également des composants "haute densité" avec les familles Virtex (4 millions de portes) et Virtex II (6 millions de portes). La famille Virtex apporte plusieurs nouveautés par rapport à la famille XC4000 :

- l'adjonction de blocs mémoires de 4 Kbits au coeur de la logique et même de plus larges mémoires dans la famille "Extended Memory".
- l'utilisation de boucles à verrouillage de phase améliorées : DLL (Digital Delay Locked Loop) qui permettent de synchroniser une horloge interne sur une horloge externe, de travailler en quadrature de phase et de multiplier ou diviser la fréquence.
- La présence d'un anneau de connexions autour du circuit pour faciliter le routage des entrées-sorties.
- La compatibilité avec de nombreux standards de transmission de données et de niveaux logiques.

L'architecture des cellules logiques est toujours basée sur des LUT à 4 entrées configurables également en petites mémoires RAM 16 bits. De plus, de la logique a été ajoutée pour permettre la synthèse de plus larges LUT comme une combinaison des LUT existantes. Le plus large circuit de cette famille (le circuit XCV3200E) contient un réseau de 104x156 cellules logiques et est équivalent à environ 4 millions de portes logiques.

Finalement, la famille Virtex II Pro améliore encore un peu le modèle précédent :

- Des blocs de mémoire de 18 Kbits.
- Des multiplieurs signés de 18x18 bits vers 36 bits.

- Des buffers Tri-state en interne pour réaliser des bus.
- Le contrôle des impédances de sortie pour adapter chaque impédance à celle de la piste du circuit imprimée.
- Le DCM (Digital Clock Manager) qui est une évolution du DLL et qui affine encore le déphasage et la synthèse des horloges internes. Le plus large circuit de cette famille (le circuit XC2V10000) contiendra 192 blocs mémoire de 18Kbits, 192 multiplieurs, un réseau de 128x120 cellules logiques et sera équivalent à environ 10 millions de portes logiques (selon Xilinx).

I.1.2. 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 III.5 présente la structure de ces blocs.

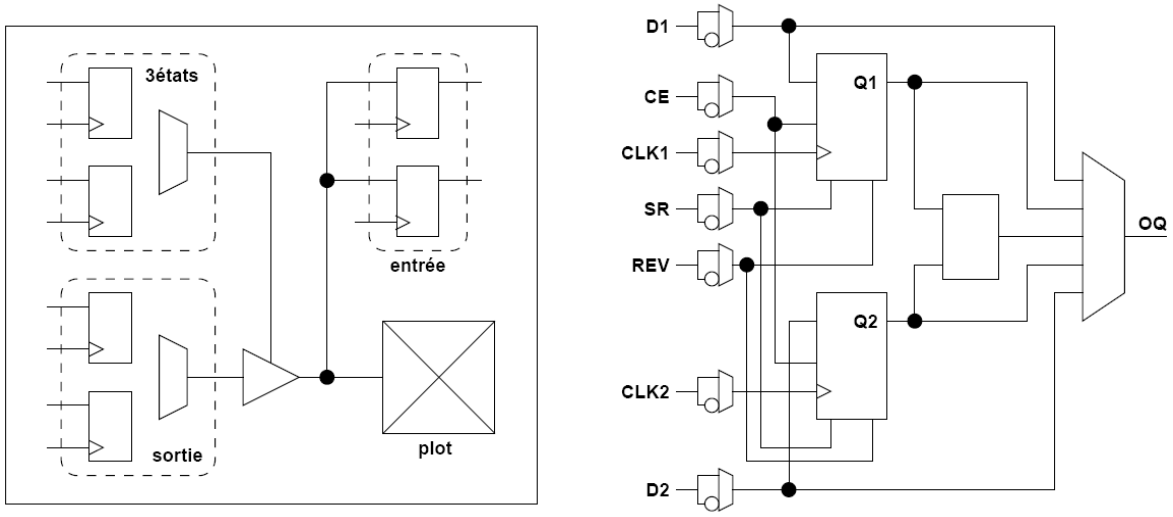


Figure III.5 : Schéma d'un bloc d'entrée/sortie (IOB)

● Configuration en entrée

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

● Configuration en sortie

Nous distinguons les possibilités suivantes :

- inversion ou non du signal avant son application à l'IOB ;
- synchronisation du signal sur des fronts montants ou descendants d'horloge ;
- mise en place d'un "pull-up" ou "pull-down" dans le but de limiter la consommation des entrées sorties inutilisées ;

- 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.

I.1.3. Les différents types d'interconnexions

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'interconnexions selon la longueur et la destination des liaisons. Nous disposons :

- d'interconnexions à usage général,
- d'interconnexions directes,
- de longues lignes.

I.1.3.1. Les interconnexions à usage général

Ce système fonctionne en une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes des CLB et des IOB.

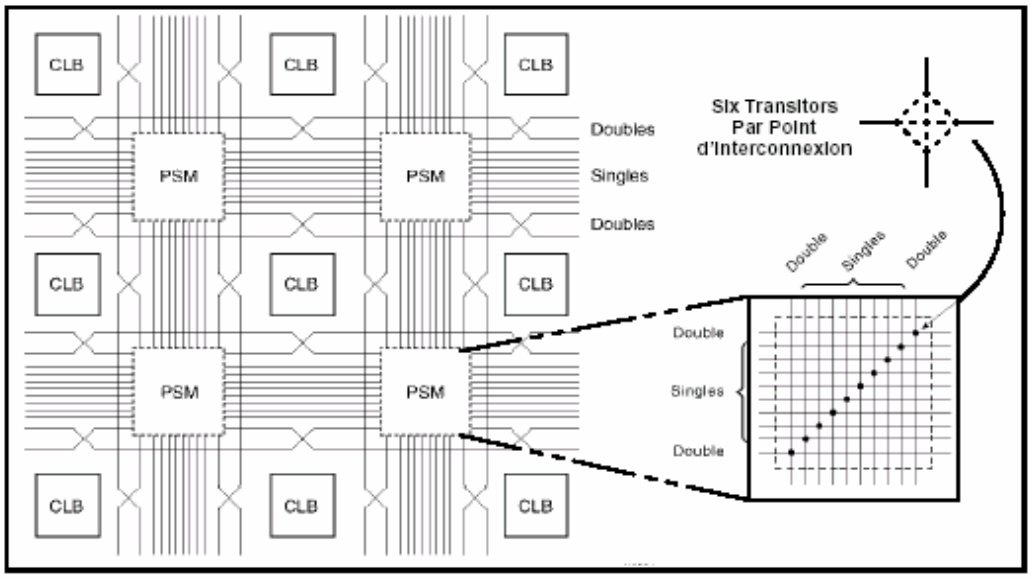


Figure III.6 : Connexions à usage général et détail d'une matrice de commutation

Des aiguilleurs appelés aussi « matrices de commutation » (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.

I.1.3.2. Les interconnexions directes

Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en terme 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.

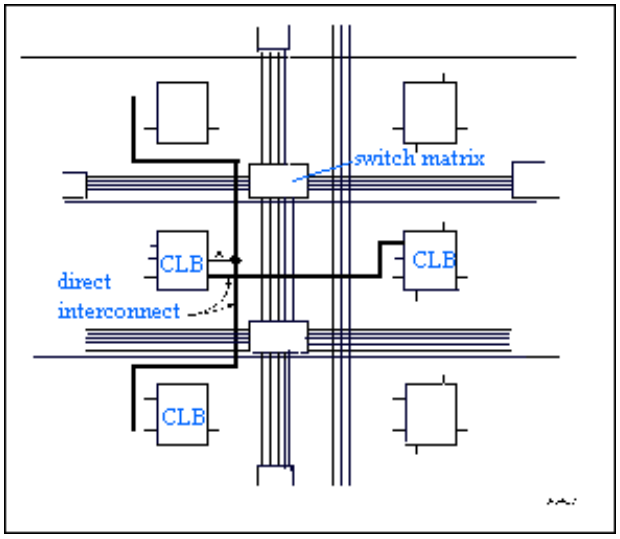


Figure III.7 : Les interconnexions directes

Pour chaque bloc logique configurable, la sortie X peut être connectée directement aux entrées C ou D du CLB situé au-dessus et les entrées A ou B du CLB situé au-dessous. Quant à la sortie Y, elle peut être connectée à l'entrée B du CLB placé immédiatement à sa droite. Pour chaque bloc logique adjacent à un bloc entrée/sortie, les connexions sont possibles avec les entrées I ou les sorties O suivant leur position sur le circuit.

I.1.3.3. 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.

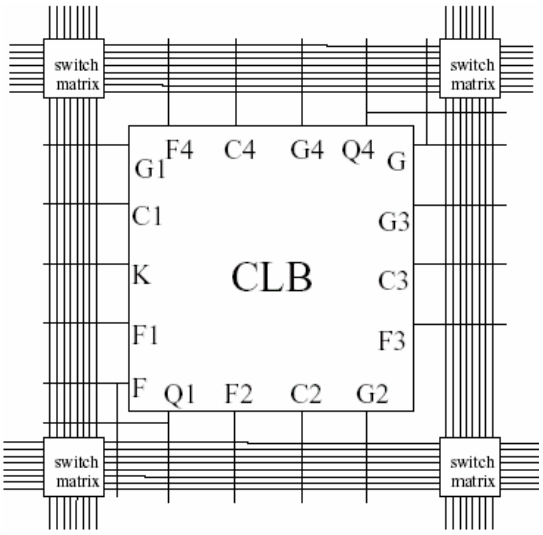


Figure III.8 : Les longues lignes

I.1.3.4. Performances des interconnexions

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions à usage général, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connectique de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ils possèdent une faible résistance mais une capacité importante. De plus, si on utilise un aiguilleur, sa résistance s'ajoute à celle existante.

I.2. Kit de développement Virtex-II V2MB1000 de Memec Design

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 inclue 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 à des 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 III.9 présente une photo de la carte de développement et de ses outils [8].

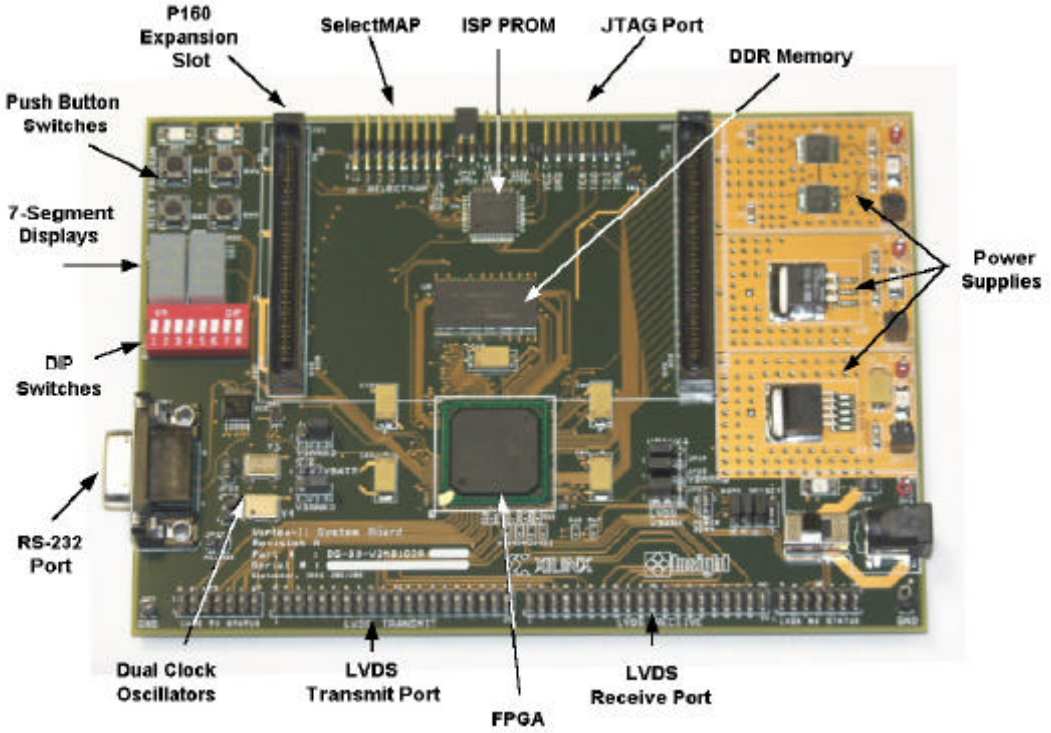


Figure III.9 : Carte de développement « Memec Design Virtex-II »

I.2.1. Description de la carte de développement

Un diagramme simplifié de la carte de développement Virtex-II est illustré à la figure III.10.

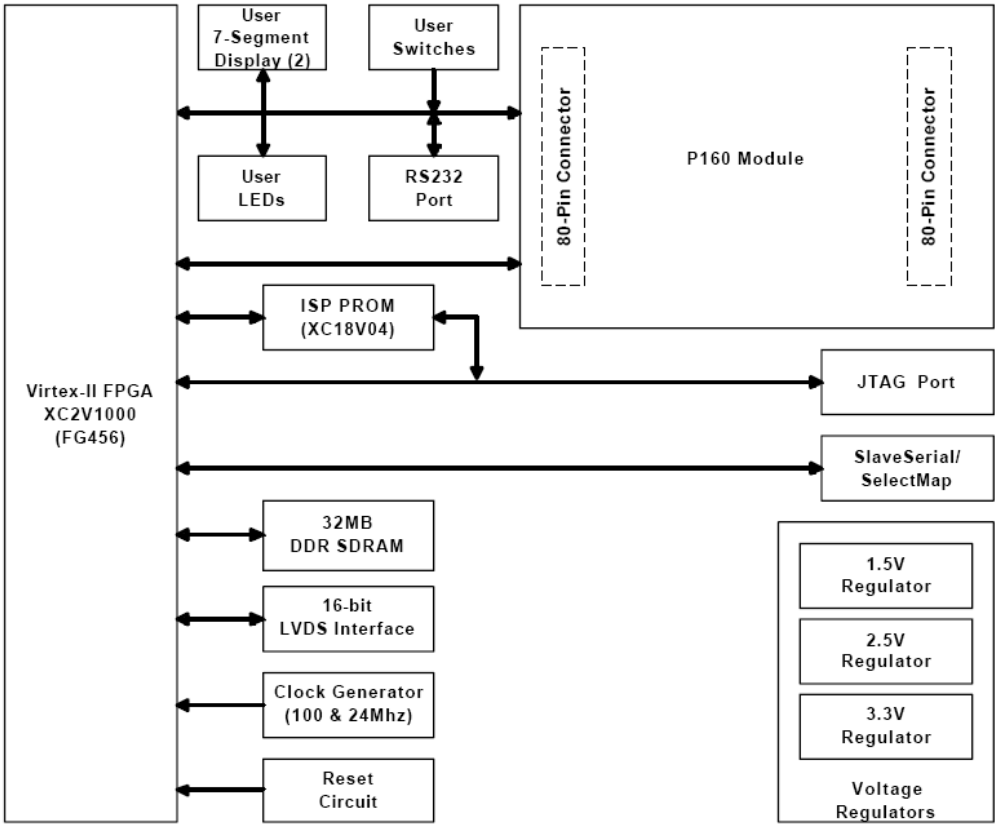


Figure III.10 : Diagramme de la carte de développement

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 développés pour des applications haute performance telles que 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 [9].

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 p8résents 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.

I.2.2. 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 suivante montre l'installation pour toutes les configurations de modes supportés par la carte de développement Virtex-II.

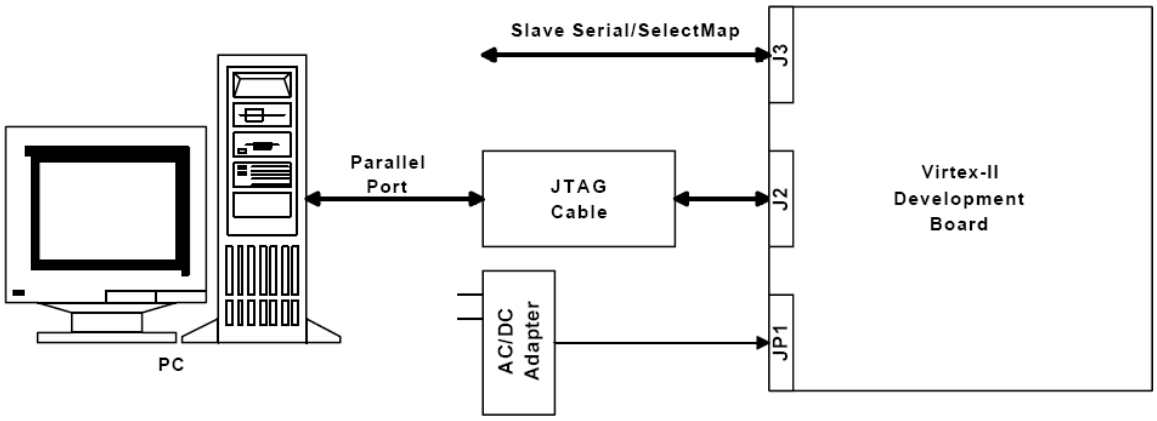


Figure III.11 : Chargement du programme sur la carte

● Utilisation de l'interface JTAG

Le câble Memec Design JTAG est connecté d'un coté 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.

II. 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).

Le VHDL est un langage permettant de faire :

● **La spécification** : le langage VHDL est très bien adapté à la modélisation des systèmes numériques complexes grâce à son niveau élevé d'abstraction. Le partitionnement en plusieurs sous ensembles permet de subdiviser un modèle complexe en plusieurs éléments prêts à être développés séparément.

● **La simulation** : la notion du temps, présente dans le langage, permet son utilisation pour décrire des fichiers de simulation (Test-Bench). Le modèle comportemental avec les fichiers de simulation peuvent constituer, ensemble, un cahier de charges. Les fichiers de simulation peuvent également être utilisés avec un banc de tests de production.

● **La synthèse logique** : les logiciels de synthèse permettent de traduire la description VHDL en logique. Il est ainsi possible d'intégrer la description dans un composant programmable (CPLD, FPGA) ou dans un circuit ASIC.

● **La preuve formelle** : le langage permet de prouver formellement que 2 descriptions sont parfaitement identiques au niveau de leur fonctionnalité [10].

II.1. 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.

Le schéma ci-dessous représente un exemple de descriptions VHDL ou de blocs fonctionnels implantés dans un FPGA [11]. 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.

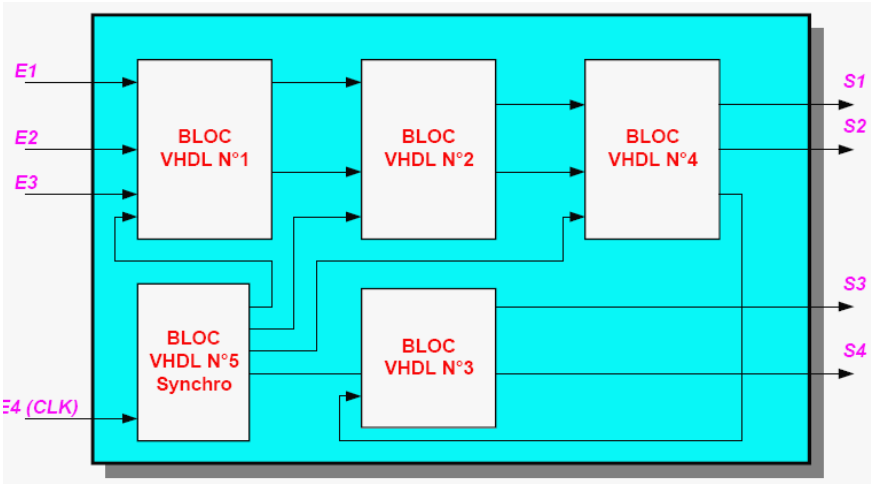


Figure III.12 : Schéma fonctionnel d'implantation d'une description VHDL dans un circuit FPGA

II.2. Structure d'une description VHDL simple

La structure typique d'une description VHDL est composée de 2 parties indissociables qui sont : l'entité et l'architecture. La figure III.13 illustre ceci.

● L'entité

L'entité (ENTITY) est vue comme une boite noire avec des entrées et des sorties caractérisée par des paramètres. Elle représente une vue externe de la description.

● L'architecture

L'architecture (ARCHITECTURE) contient les instructions VHDL permettant de décrire et de réaliser le fonctionnement attendu. Elle représente la structure interne de la description.

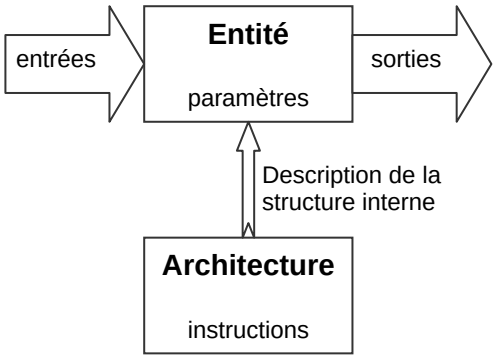


Figure III.13 : Structure de base d'une description VHDL

A une entité peuvent correspondre plusieurs architectures. De cette manière, il devient possible de modifier la structure interne d'une description sans modifier son aspect externe. Cette propriété du VHDL devient très intéressante dès lors que plusieurs descriptions sont imbriquées les unes dans les autres.

Les entités et architectures sont des unités de conception dites primaire et secondaire. Un couple entité-architecture donne une description complète d'un élément ; ce couple est appelé modèle.

II.2.1. Déclaration des bibliothèques

Toute description VHDL utilisée pour la synthèse a besoin de bibliothèques. L'IEEE les a normalisées et plus particulièrement la bibliothèque IEEE1164. Elles contiennent les définitions des types de signaux électroniques, des fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques, etc. La directive **use** permet de sélectionner les bibliothèques à utiliser.

Exemple :
```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
```

II.2.2. Déclaration de l'entité et des entrées/sorties

Cette opération permet de définir le nom de la description VHDL ainsi que les entrées et sorties utilisées, l'instruction qui les définit est **port** :

Exemple :
```
entity DEC7SEG4 is
port (DEC :in std_logic_vector(3 downto 0);
      SEG:out std_logic_vector(6 downto 0)
     );
end DEC7SEG4;
```

Le nom du signal est composé d'une chaîne de caractères dont le premier est une lettre. Le langage VHDL n'est pas sensible à la « casse », c'est à dire qu'il ne fait pas la distinction entre les majuscules et les minuscules.

Le signal peut être défini en entrée (**in**), en sortie (**out**), en entrée/sortie (**inout**) ou en **buffer**, c'est-à-dire qu'il est en sortie mais il peut être utilisé comme entrée à l'intérieur de la description (figure III.14). L'affectation des broches d'entrées/sorties se fait par la définition d'attributs supplémentaires qui dépendent du logiciel de développement utilisé.

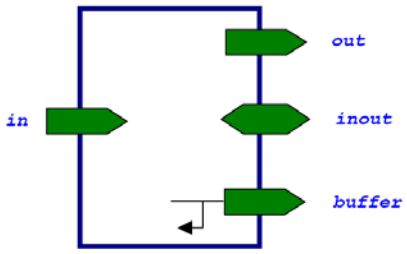


Figure III.14 : Sens des signaux d'entrées/sorties dans une description VHDL

Les types utilisés pour les signaux d'entrées/sorties sont :

- le **std_logic** pour un signal à un bit ;
- le **std_logic_vector** pour un bus composé de plusieurs bits.

Exemple : `LATCH : inout std_logic_vector (4 downto 0) ;`

Ce signal est un bus bidirectionnel de 5 bits, où `LATCH(4)` correspond au MSB et `LATCH(0)` correspond au LSB.

Les valeurs que peut prendre un signal de type `std_logic` sont :

- **'0'** ou **'L'** pour un niveau bas.
- **'1'** ou **'H'** pour un niveau haut.
- **'Z'** pour état haute impédance.
- **'-'** pour un état quelconque, c'est à dire n'importe quelle valeur ('0' ou '1').

II.2.3. Déclaration de l'architecture correspondante à l'entité

L'architecture décrit le fonctionnement souhaité pour un circuit ou une partie du circuit. En effet le fonctionnement d'un circuit est généralement décrit par plusieurs modules VHDL. Il faut comprendre par module le couple entité/architecture. Dans le cas de simples PLDs on trouve souvent un seul module. L'architecture établit les relations entre les entrées et les sorties à travers les instructions. On peut avoir un fonctionnement purement combinatoire, séquentiel voire les deux : séquentiel et combinatoire.

Exemple :

```
architecture DESCRIPTION of DEC7SEG4 is
begin
      SEG <= "1111110" when DEC = 0
        else "0110000" when DEC = 1
        else "1101101" when DEC = 2
        else "1111001" when DEC = 3
        else "0110011" when DEC = 4
        else "1011011" when DEC = 5
        else "1011111" when DEC = 6
        else "1110000" when DEC = 7
        else "1111111" when DEC = 8
        else "1111011" when DEC = 9
        else "-------";
end DESCRIPTION;
```

II.2.4. Opérateurs VHDL

Dans une description VHDL, l'opérateur le plus utilisé est l'affectation (**<=**). Il permet de modifier l'état d'un signal en fonction d'autres signaux et/ou d'autres opérateurs. L'opérateur concaténation (**&**) permet de joindre des signaux entre eux pour former un nouveau signal dont le nombre de bits égal à la somme des nombres de bits des deux signaux.

Le VHDL dispose également d'opérateurs logiques (**and**, **or**, **not**, **nor**, **xor**, …), arithmétiques (**+**,**-**,*****,**/**) et relationnels (**=**, **>**, **<**, **>=**, **<=**, **/=**) qui sont prédéfinis dans les bibliothèques `ieee.std_logic_1164.all` et `ieee.numeric_std.all`.

Exemple :
```
if (A>=F or B=C) then
  S1 <= A&B;
  S2 <= (A*C)+D;
end if;
```

II.2.5. Exemple d'une description VHDL simple

La figure suivante montre un exemple simple d'une description VHDL d'un décodeur 1 parmi 4. Les différentes parties de la description sont illustrées par des commentaires.

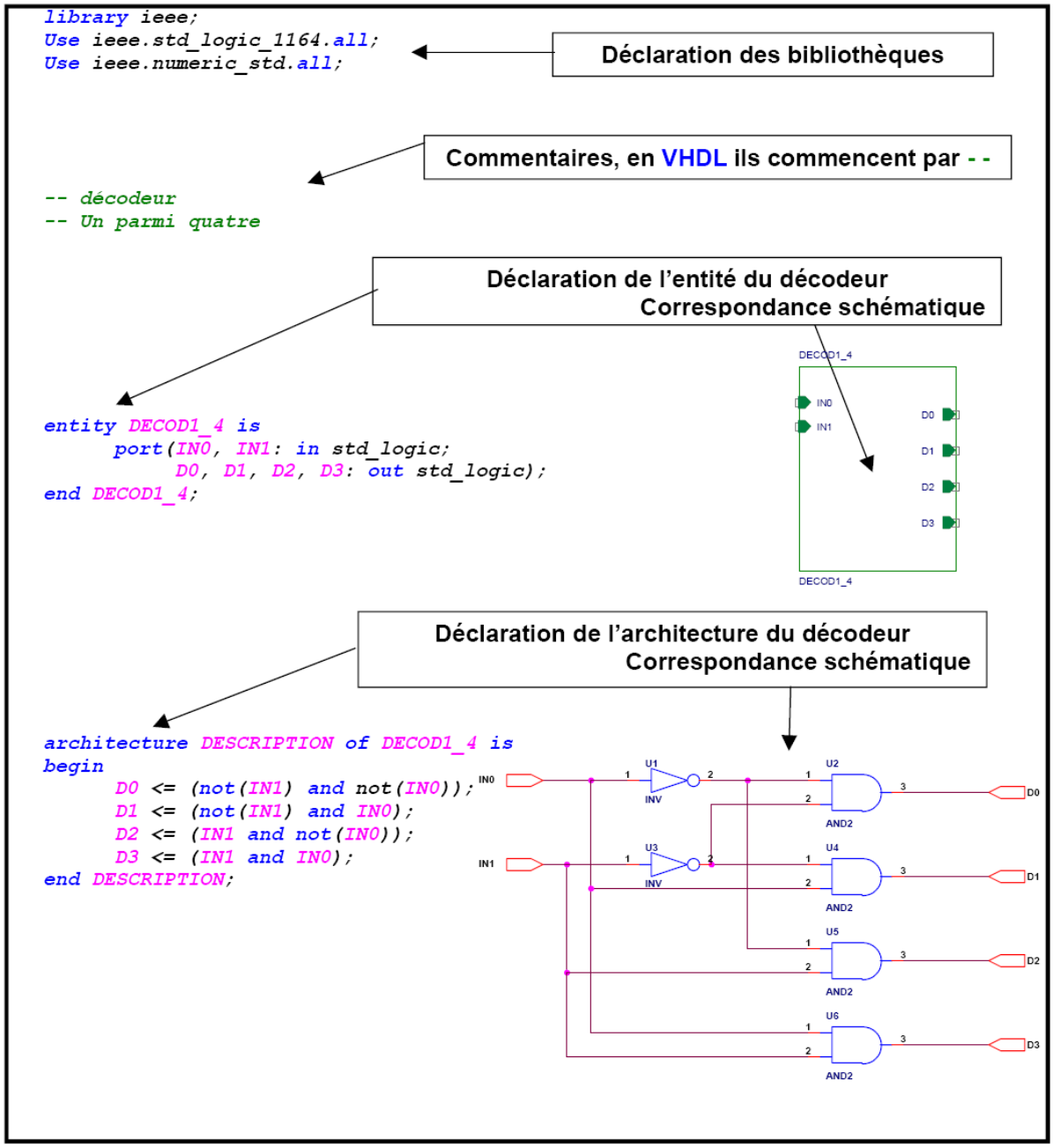


Figure III.15 : Exemple d'une description VHDL simple

II.3. Les deux modes de travail en VHDL

Le VHDL utilise deux modes de fonctionnement : le mode combinatoire (ou concurrent) et le mode séquentiel. Chacun de ces modes est utilisé dans des cas bien précis.

II.3.1. Le mode combinatoire

En mode combinatoire (ou concurrent), toutes les instructions d'une description VHDL sont évaluées et affectent les signaux de sortie en même temps, l'ordre dans lequel les instructions sont écrites n'a donc aucune importance. En effet la description génère des structures électroniques, c'est la grande différence entre une description VHDL et un langage informatique classique.

Dans un système à microprocesseur, les instructions sont exécutées les unes à la suite des autres. Avec VHDL il faut essayer de penser à la structure qui va être générée par le synthétiseur pour écrire une bonne description, cela n'est pas toujours évident.

Il existe deux instructions de base utilisés en mode combinatoire :

● L'affectation conditionnelle :

Cette instruction permet de modifier l'état d'un signal suivant le résultat d'une condition logique entre des signaux, des valeurs et/ou des constantes. Sa syntaxe est la suivante :

```
SIGNAL <= expression when condition
          [else expression when condition]
          [else expression];
```

● L'affectation sélective :

Cette instruction permet d'affecter différentes valeurs à un signal, selon les valeurs prises par un signal dit de sélection. Sa syntaxe se présente comme suit :

```
with SIGNAL_DE_SELECTION select
SIGNAL <= expression when valeur_de_selection,
          [expression when valeur_de_selection,]
          [expression when others];
```

II.3.2. Le mode séquentiel

Le mode séquentiel utilise les process dans lesquels le temps est une variable essentielle. Un process est une partie de la description d'un circuit dans laquelle les instructions sont exécutées séquentiellement, c'est-à-dire les unes à la suite des autres. Il permet d'effectuer des opérations sur les signaux en utilisant les instructions standard de la programmation structurée comme dans les systèmes à microprocesseurs.

Dans le mode séquentiel, on distingue :

- **Le mode séquentiel asynchrone,** dans lequel les changements d'état des sorties peuvent se produire à des instants difficilement prédictibles (retards formés par le basculement d'un nombre souvent inconnu de fonctions),

- **Le mode séquentiel synchrone,** dans lequel les changements d'état des sorties se produisent tous à des instants quasiment connus (un temps de retard après un front actif de l'horloge).

Avant, pour des raisons de simplification de câblage, on utilisait la logique asynchrone pour réaliser des fonctions complexes mettant en oeuvre de nombreux boîtiers. Mais du fait de la difficulté qu'il y a à prévoir les instants de changement d'état des signaux, il apparaissait souvent des impulsions indésirables (Glitch) pouvant générer des résultats non conformes aux attentes ; d'où des difficultés souvent importantes dans la mise au point.

Les FPGA permettent maintenant de réaliser ces mêmes fonctions logiques complexes en fonctionnement complètement synchrone. La bonne prédictibilité des instants de changement d'état permet de réduire considérablement les aléas de fonctionnement. Maintenant, de l'avis unanime de tous les utilisateurs expérimentés de ce type de circuit, il faut utiliser impérativement des méthodes de synthèse synchrone.

L'exécution d'un process est déclenchée par un ou des changements d'états de signaux logiques. Le nom de ces signaux est défini dans la liste de sensibilité lors de la déclaration du process. Cette dernière suit le modèle suivant :

```
[Nom_du_process :] process(signaux_liste_de_sensibilité)
Begin
    instructions du process
end process [Nom_du_process] ;
```

L'exécution d'un process a lieu à chaque changement d'état d'un signal de la liste de sensibilité. Les changements d'état des signaux par les instructions du process ne sont pris en compte qu'à la fin du process.

Le mode séquentiel utilise également deux structures d'instructions de base :

● L'assignation conditionnelle :

Elle permet d'exécuter des instructions séquentiellement et uniquement quand la condition est remplie. Elle suit la syntaxe suivante :

```
if condition then
    instructions
   [elsif condition then instructions]
   [else instructions]
end if ;
```

● L'assignation sélective :

Elle permet d'exécuter les instructions séquentiellement suivant les valeurs prises par le signal de sélection. Sa syntaxe se présente comme suit :

```
case signal_de_slection is
     when valeur_de_sélection => instructions
    [when others => instructions]
end case;
```

III. Etapes nécessaires au développement d'un projet sur FPGA

Le développement en VHDL nécessite l'utilisation de deux outils : le simulateur et le synthétiseur. Le premier va nous permettre de simuler notre description VHDL avec un fichier de simulation appelé « test-bench » ; cet outil interprète directement le langage VHDL et il comprend l'ensemble du langage. L'objectif du synthétiseur est très différent : il doit traduire le comportement décrit en VHDL en fonctions logiques de bases, celles-ci dépendent de la technologie choisie ; cette étape est nommée « synthèse ». L'intégration finale dans le circuit cible est réalisée par l'outil de placement et routage. Celui-ci est fourni par le fabricant de la technologie choisie.

Le langage VHDL permet d'écrire des descriptions d'un niveau comportemental élevé. La question est de savoir si n'importe quelle description comportementale peut être traduite en logique ?

Avec les outils actuels, il est possible de disposer de fichiers VHDL à chaque étape. Le même fichier de simulation (test-bench) est ainsi utilisable pour vérifier le fonctionnement de la description à chaque étape de la procédure de développement. La figure III.16 donne les différentes étapes nécessaires au développement d'un projet sur circuit FPGA.

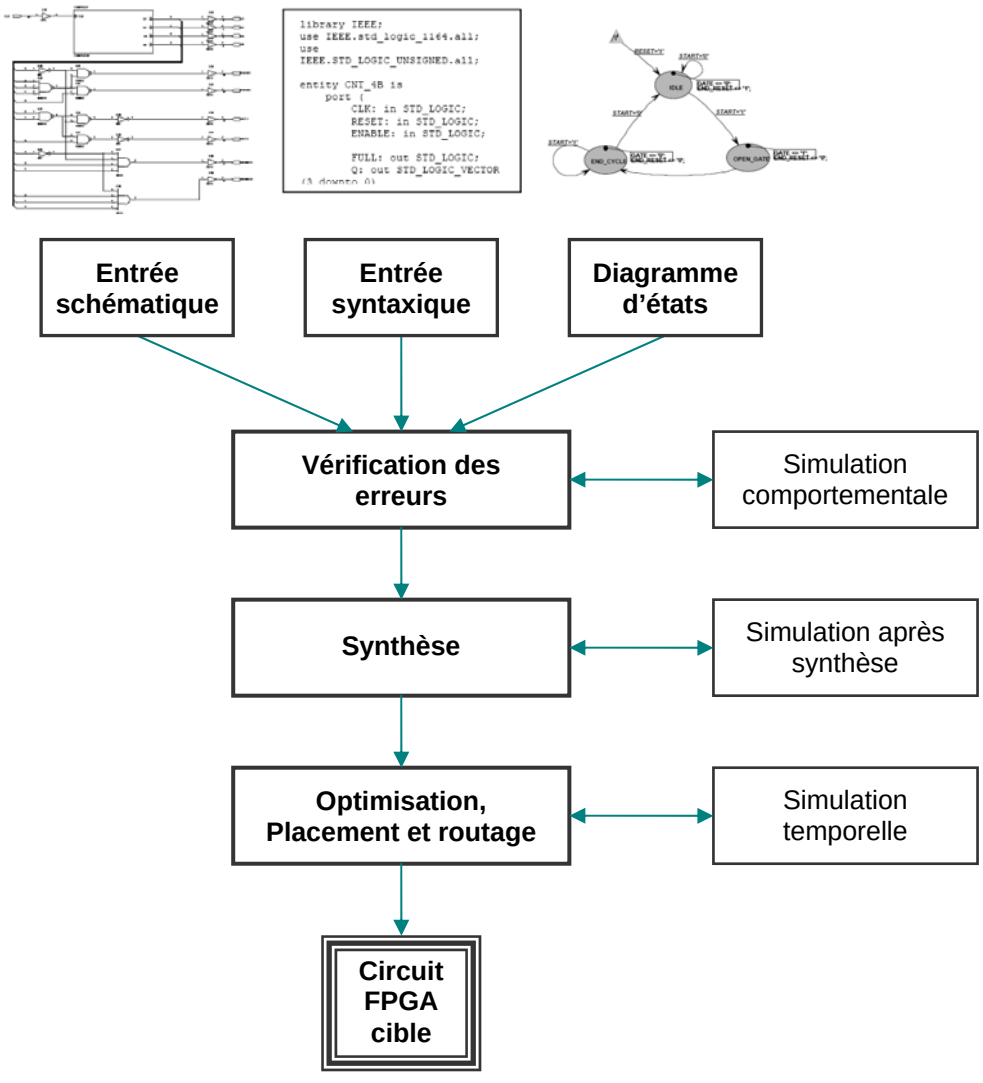


Figure III.16 : Organisation fonctionnelle de développement d'un projet sur circuit FPGA

III.1. Saisie du texte VHDL

La saisie du texte VHDL se fait sur le logiciel « ISE Xilinx Project Navigator ». Ce logiciel 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.

La figure III.17 montre comment se présente le logiciel «ISE Xilinx Project Navigator». La saisie du texte se fait sur la partie droite de l'écran, on voit en haut à gauche la hiérarchie du projet, et en bas à gauche les nombreux outils nécessaires tout au long du développement du projet.

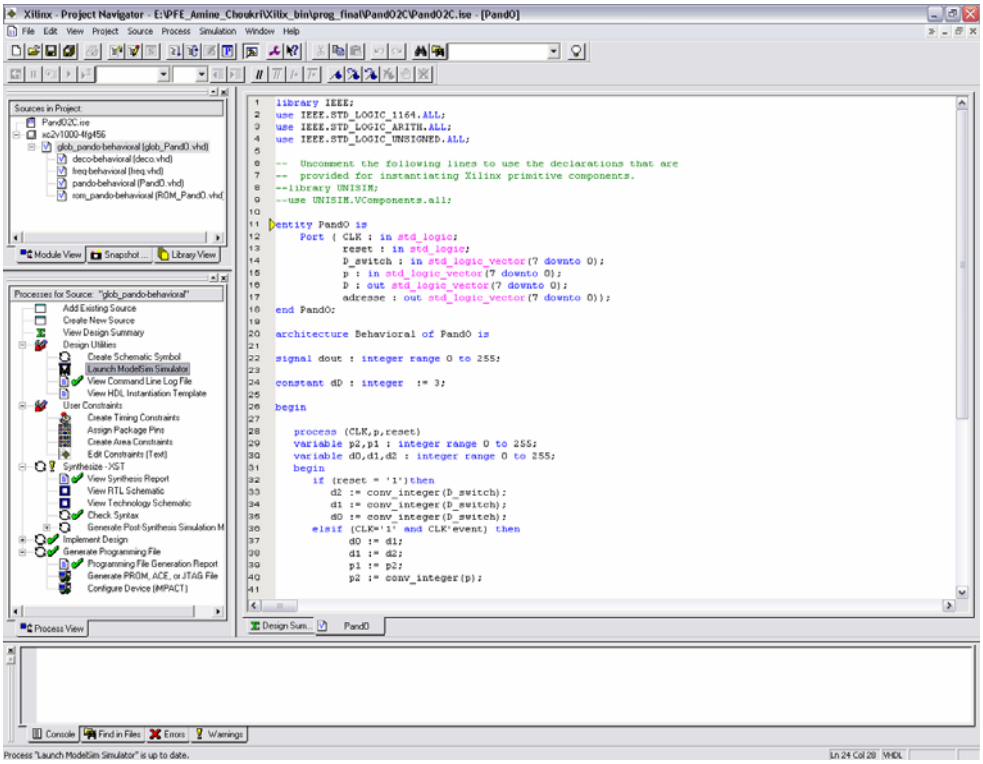


Figure III.17 : Vue d'ensemble du logiciel « Xilinx Project Navigator »

Il faut commencer par créer un projet, ensuite inclure des fichiers sources dans lesquels il faut saisir le texte VHDL désiré. On peut inclure autant de sources qu'on veut dans un projet.

III.2. Vérification des erreurs

Cette étape est effectuée en appuyant sur le bouton « check syntax ». Elle permet de vérifier les erreurs (errors) de syntaxe du texte VHDL et d'afficher les différentes alarmes (warnings) liées au programme, par exemple des signaux déclarés mais non utilisés dans le programme. S'il y'a des erreurs dans le programme, il ne peut pas être synthétisé, mais la présence d'alarmes n'empêche pas de poursuivre normalement les autres étapes du développement.

Cette étape permet donc de valider la syntaxe du programme et de générer la « netlist », qui est un fichier contenant la description de l'application sous forme d'équations logiques.

III.3. 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 (figureIII.18).

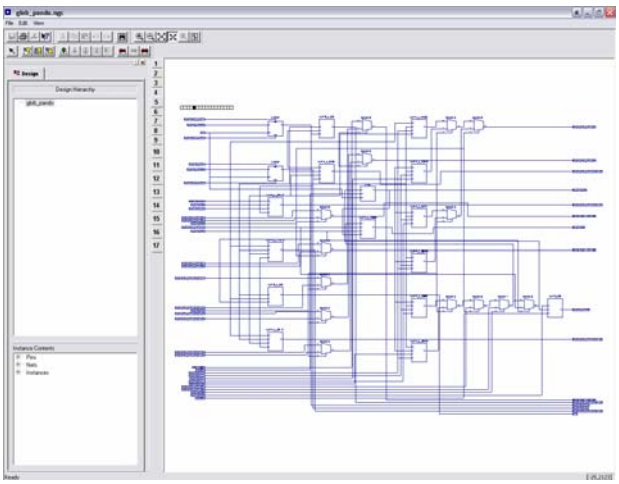


Figure III.18 : Aperçu de l'outil « View RTL Schematic »

De plus, le synthétiseur permet à l'utilisateur d'imposer des contraintes de technologie (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 III.19 montre un aperçu de l'outil d'assignation des broches d'entrées/sorties.

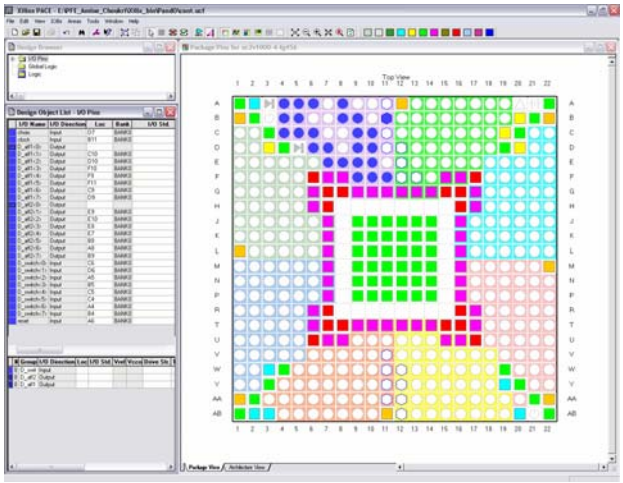


Figure III.19 : Aperçu de l'outil d'affectation des broches d'entrées/sorties

III.4. Simulation

Le simulateur utilisé est le « ModelSim Simulator » (figure III.20). La simulation permet de vérifier le comportement d'un design avant ou après implémentation dans le composant cible. Elle représente une étape essentielle qui nous fera gagner du temps lors de la mise au point sur la carte. Il faut juste noter qu'un projet peut être simulé même s'il n'est pas synthétisable.

Lors de l'étape de simulation comportementale, on valide l'application indépendamment de l'architecture et des temps de propagation du futur circuit cible. La phase de simulation après synthèse valide l'application sur l'architecture du circuit cible, et enfin la simulation temporelle prend en compte les temps de propagation des signaux à l'intérieur du circuit FPGA cible.

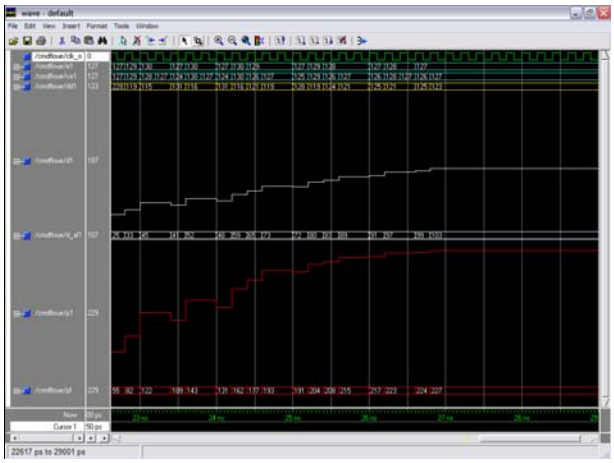


Figure III.20 : Présentation du simulateur « ModelSim Simulator »

III.5. Optimisation, placement et routage

Pendant l'étape d'optimisation, l'outil cherche à minimiser les temps de propagation et à occuper le moins d'espace possible sur le circuit FPGA cible. Le placement et routage permet de tracer les routes à suivre sur le circuit afin de réaliser le fonctionnement attendu. La figure III.21 donne un aperçu de l'outil de placement et routage « FPGA Editor » qui permet de visualiser et d'éditer le circuit routé.

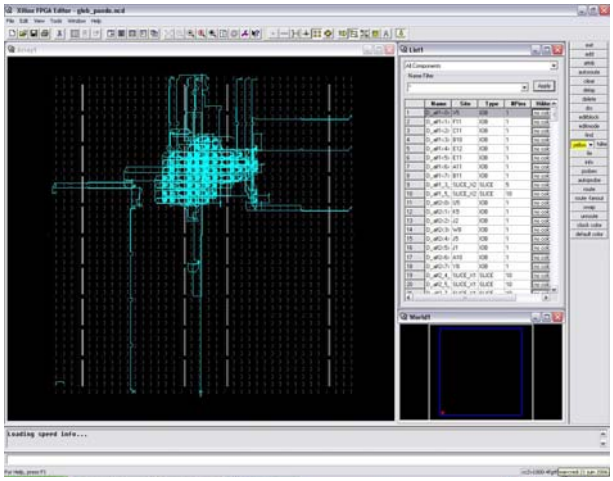


Figure III.21 : Aperçu de l'outil « FPGA Editor »

III.6. Programmation du composant et test

Dans cette dernière étape (Generate Programming Files), on génère le fichier à charger sur le circuit FPGA à travers l'interface JTAG. Une fois le programme chargé sur le circuit, on peut tester et visualiser les résultats directement sur la carte de développement Virtex-II à travers les nombreuses interfaces qu'elle offre, soit directement sur les deux afficheurs 7 segments, soit à travers l'interface RS232 ou bien à travers l'interface LVDS 16 bits.

Chapitre IV

Application, résultats et discussion

Chapitre IV

Application, résultats et discussion

L'objet de notre étude est l'implémentation sur circuit FPGA des deux méthodes MPPT déjà citées dans le chapitre II (P&O et floue optimisée par les Algorithmes Génétiques). Dans ce dernier chapitre, on présente la structure générale des programmes écrits en VHDL ainsi que les résultats de simulation obtenus pour des conditions différentes de température et d'ensoleillement. On interprète les résultats obtenus et on fait une comparaison entre les deux méthodes MPPT. Enfin, on termine par donner les observations et les conclusions qu'on peut tirer à partir de ces résultats.

La simulation nous permet de valider théoriquement les programmes écrits en VHDL avant de les implémenter sur la carte de développement. Une fois les programmes chargés sur la carte, il faut effectuer des tests réels pour les valider pratiquement.

I. Description des programmes écrits en VHDL

Dans cette partie, on fait une description générale des programmes qu'on a écrit en VHDL et on les illustre par des schémas simplifiés pour chacune des deux méthodes. On explique le rôle de chaque bloc dans le programme et la relation existante entre les différents blocs. Notons juste que pour les deux méthodes, on a utilisé un codage binaire des variables sur 8 bits.

I.1. Méthode P&O

On a déjà présenté l'organigramme de la méthode Perturbation et Observation (P&O) dans le chapitre II, on a traduit cet organigramme et on l'a écrit en langage VHDL. Le programme peut être représenté par une boite noire commandée par un signal d'horloge (*CLK*), ayant comme entrée la valeur de la puissance instantanée (*P*), et comme sortie la valeur du rapport cyclique (*D*) qui sera appliqué au convertisseur continu-continu (figure IV.1). Les valeurs de *P* et *D* sont mémorisés à chaque fois à l'intérieur du bloc, et le calcul de *dP=P(k)-P(k-1)* et de *dD=D(k)-D(k-1)* est effectué à l'intérieur du bloc.

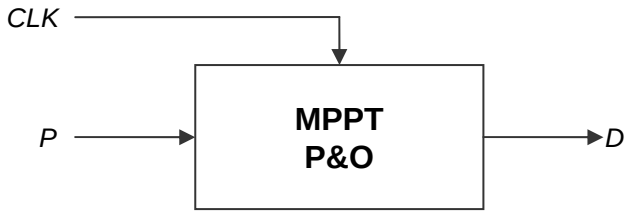


Figure IV.1 : Schéma synoptique simplifié du bloc "MPPT P&O"

Le programme est écrit sous forme d'un process qui se déclenche à chaque front montant de l'horloge (*CLK*). Le choix de la valeur de *D(k+1)* se fait par des instructions d'assignation conditionnelle (instruction if) et les résultats sont fournies en sortie instantanément.

I.2. Méthode du contrôleur flou optimisé par les AGs

La structure de base du contrôleur flou a été expliquée dans le chapitre II. On a écrit le programme en VHDL conformémen6t à cette structure en le divisant en trois blocs principaux comme suit :

- Le premier bloc (Fuzzification) contient les formes des fonctions d'appartenance qui sont mémorisées sous forme de tableaux ; il permet de calculer les valeurs des degrés de vérité des différentes fonctions d'appartenance (*NGE*, *NPE*, *ZEE*, *PPE*, *PGE* pour l'entrée *E*, et *NGC*, *NPC*, *ZEC*, *PPC*, *PGC* pour l'entrée *CE*) à partir des valeurs des entrées *E* et *CE*.
- Le second bloc (Inférences) contient les règles d'inférences qui permettent de calculer, à partir des degrés de vérité de *E* et *CE*, le degré d'activation de chacune des fonctions d'appartenance (*NGD*, *NPD*, *ZED*, *PPD*, *PGD*) de la sortie *dD*.
- Enfin, le troisième bloc (Défuzzification) construit par agrégation l'ensemble flou de sortie et calcule la valeur réelle de la sortie *dD* par la méthode du centre de gravité.

Le contrôleur flou se compose en réalité des trois blocs suscités. Cependant pour avoir la même structure que celle du programme P&O, on a ajouté un bloc en entrée et un autre en sortie. Le bloc d'entrée permet de calculer les valeurs de *E* et *CE* à partir de l'entrée *P*, et le bloc de sortie calcule la valeur du rapport cyclique *D* à partir de la sortie *dD*.

Chacun de ces blocs est écrit sous forme de plusieurs process qui se déclenchent à chaque changement d'état de leurs entrées, et fournissent les résultats en sortie instantanément. Seul le premier bloc (entrée) est commandé par un signal d'horloge (*CLK*), donc toute la chaîne ne commence à travailler qu'à partir d'un front montant du signal d'horloge.

La figure ci-dessous donne le schéma détaillé du contrôleur flou optimisé.

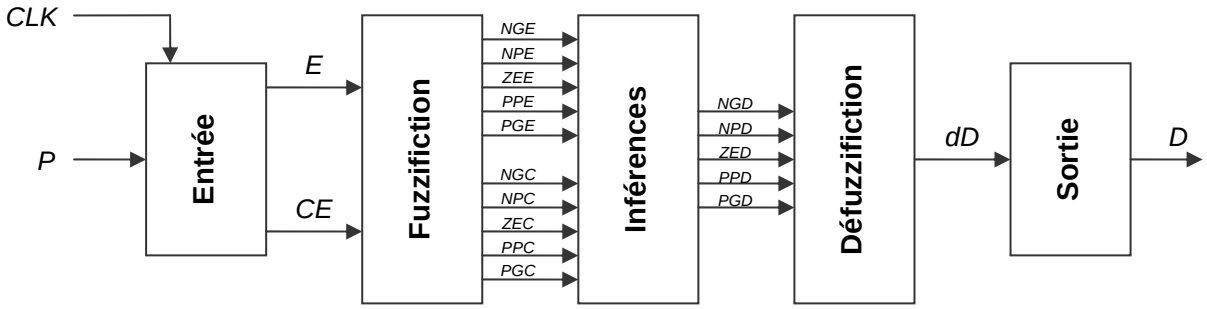


Figure IV.2 : Schéma synoptique détaillé du bloc "MPPT flou optimisé"

Ainsi, le programme peut être représenté par une boite noire ayant les mêmes entrées/sorties que celle de la méthode P&O (figure IV.3).

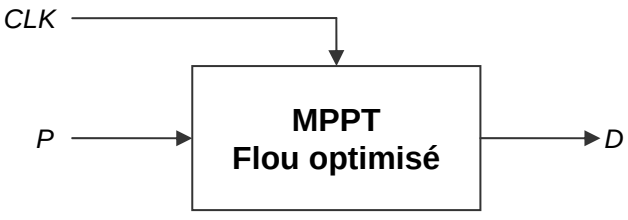


Figure IV.3 : Schéma synoptique simplifié du bloc "MPPT flou optimisé"

I.3. Environnement de test

Afin de tester le bon fonctionnement des programmes écrits en VHDL sur la carte de développement Virtex-II, on a conçu un environnement de test 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 comme le montre la figure III.4. La figure est suivie par une brève explication du rôle de chaque bloc dans l'environnement de test.

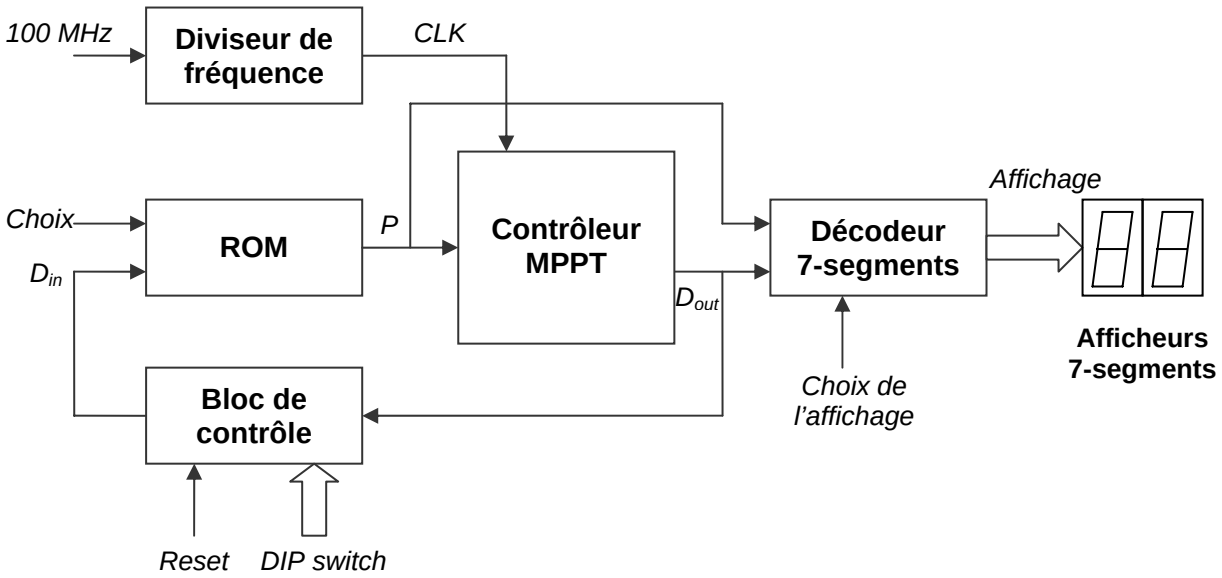


Figure IV.4 : Schéma synoptique détaillé de l'environnement de test

● Le "Diviseur de fréquence"

Ce bloc a pour rôle de générer un signal d'horloge à 1Hz à partir du signal d'horloge interne de 100 MHz. En réalité le système travaille habituellement avec une fréquence de 100 Hz, mais pour les besoins du test et pour pouvoir visualiser les résultats à l'œil nu, on travaille avec une période d'horloge de 1 seconde.

● Le "Décodeur 7-segments"

Dans ce bloc, la valeur de P ou D_{out} (selon le choix de l'utilisateur) est traduite en format 7-segments et adaptée pour pouvoir être affichée sur les deux afficheurs 7-segments disponibles sur la carte de développement.

● Les "Deux afficheurs 7-segments"

Ces deux afficheurs permettent de visualiser la valeur de la puissance P ou du rapport cyclique D en format hexadécimal. On peut ainsi suivre la variation de l'une de ces deux grandeurs (selon le choix de l'utilisateur) en temps réel et ainsi vérifier le bon fonctionnement du dispositif.

● Le bloc "ROM"

Ce bloc contient la caractéristique puissance-tension d'un système photovoltaïque sous forme de tableau, il remplace donc dans notre test un système photovoltaïque complet. Il contient deux courbes différentes de la caractéristique puissance-tension correspondant à des conditions atmosphériques différentes d'ensoleillement et de température. Ces deux courbes sont illustrées à la figure IV.5, et le choix de la courbe désirée est effectué en appuyant sur le bouton "choix".

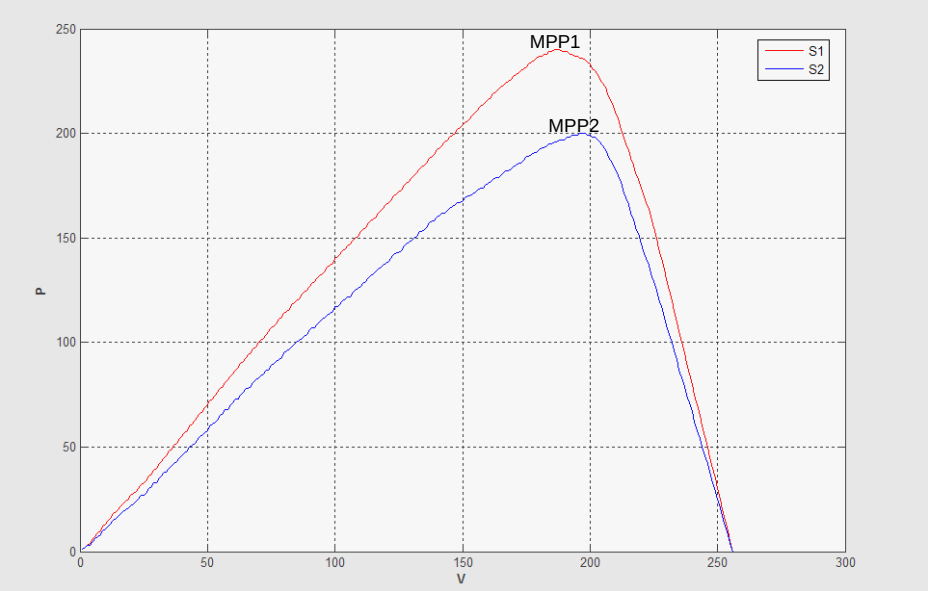


Figure IV.5 : Les deux courbes de la caractéristique puissance-tension contenus dans le bloc "ROM"

On a introduit deux courbes différentes de la caractéristique puissance-tension dans le bloc ROM pour disposer de deux points MPP (Maximum Power Point) différents. On peut ainsi estimer et comparer la rapidité de la réponse des deux méthodes de poursuite MPPT en cas de changement brusque des conditions météorologiques. Notons que pour la courbe *S1* l'ensoleillement est plus important que pour la courbe *S2*.

● Le "Bloc de contrôle"

Ce bloc permet de réinitialiser le système avec une nouvelle valeur du rapport cyclique en appuyant sur le bouton poussoir "Reset". L'utilisateur peut introduire la valeur initiale désirée du rapport cyclique *D* en agissant sur l'entrée à 8 bits "DIP switch". Ces switchs peuvent être mis statiquement à 1 ou à 0 par l'utilisateur pour former n'importe quel mot binaire de 8 bits et ainsi balayer tout l'intervalle de variation du rapport cyclique *D* (*D=0* correspond au mot binaire "00000000" *et D=1* correspond à "11111111").

● Le "Contrôleur MPPT"

Ce bloc représente le bloc VHDL à tester, c'est-à-dire le cœur du programme. Pour la première méthode, on l'a remplacé par le bloc "MPPT P&O", et pour la seconde méthode par le bloc "MPPT flou optimisé". Ces deux blocs ont été détaillés avec des schémas simplifiés et leur principe de fonctionnement a été expliqué plus haut.

II. Synthèse

Pendant l'étape de synthèse, le synthétiseur convertit le programme VHDL en portes logiques et bascules de base, c'est-à-dire en structure électronique. 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.

II.1. Synthèse du programme de la méthode P&O

La figure IV.6 montre le schéma équivalent du bloc VHDL du contrôleur P&O généré par le synthétiseur.

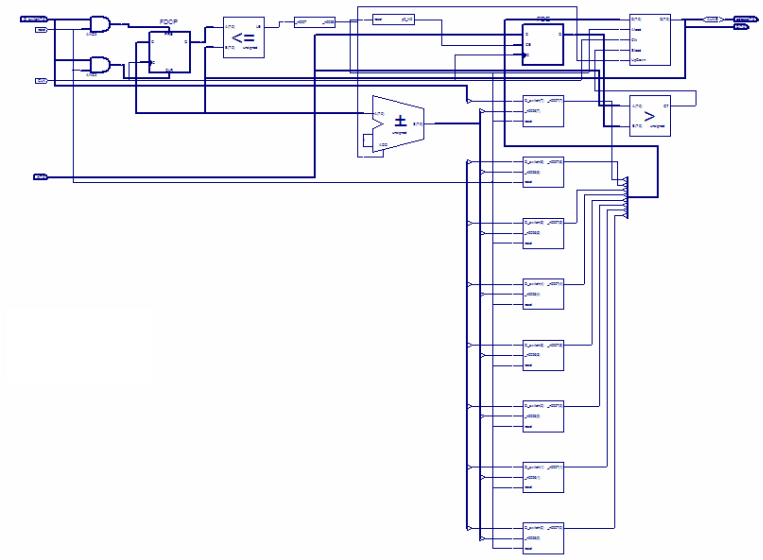
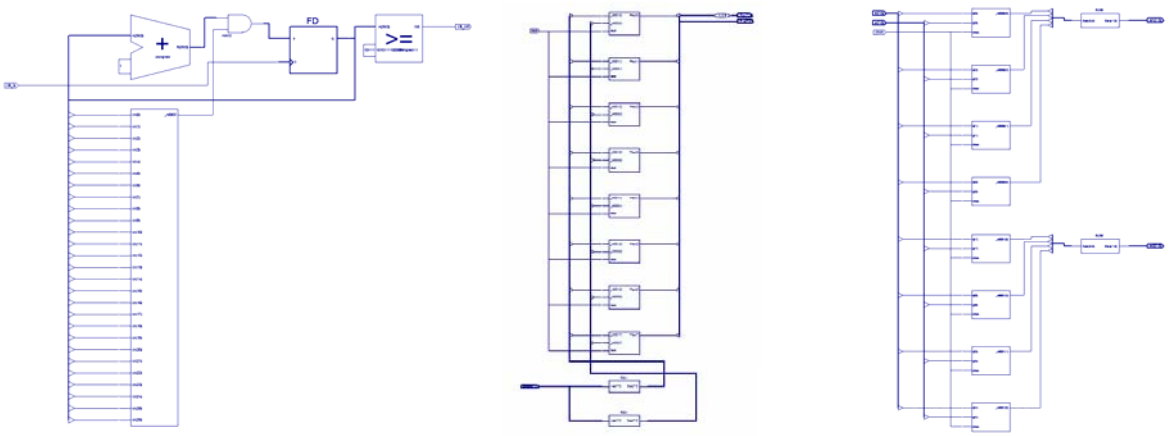


Figure IV.6 : Schéma équivalent du bloc "MPPT P&O" généré par le synthétiseur

La figure IV.7 montre les schémas électroniques équivalents que le synthétiseur a générés pour chacun des blocs de l'environnement de test.



a) Bloc "Diviseur de fréquence" *c) Bloc "ROM"* *b) Bloc "Décodeur 7- segments"*

Figure IV.7 : Schémas équivalents des différents blocs de l'environnement de test

La figure IV.8 présente le schéma équivalent du programme principal de la méthode P&O, c'est-à-dire du bloc P&O dans son environnement de test. On peut voir les différentes entrées/sorties ainsi que la relation entre les différents blocs. On remarque bien que ce schéma correspond au schéma synoptique qu'on a décrit plus haut.

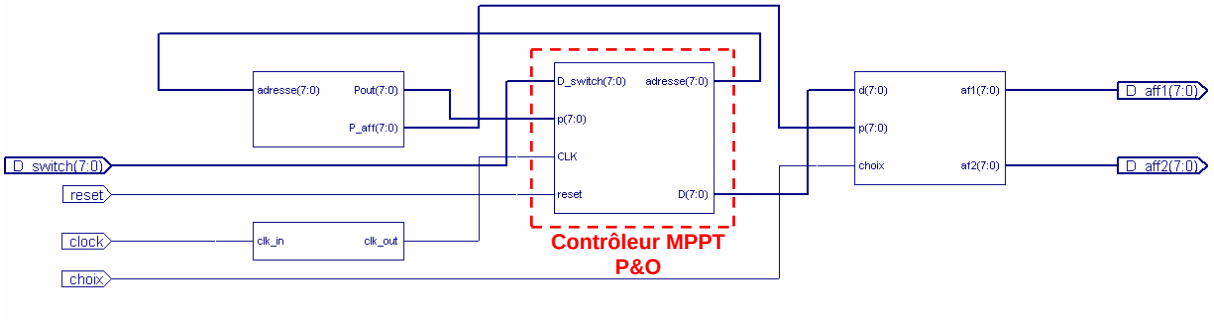
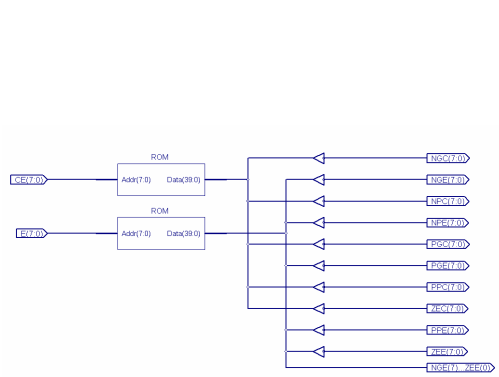


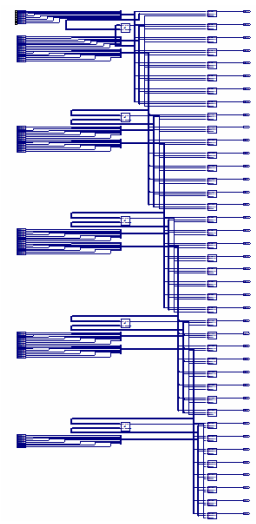
Figure IV.8 : Schéma équivalent du programme "P&O global"

II.2. Synthèse du programme du contrôleur flou optimisé par les AGs

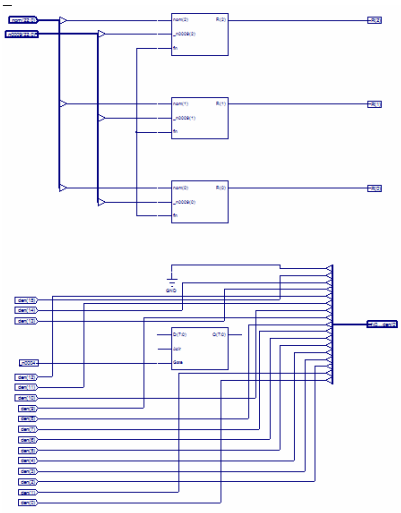
Dans la figure IV.9, on peut voir les schémas équivalents des différents blocs constituant le contrôleur flou optimisé.



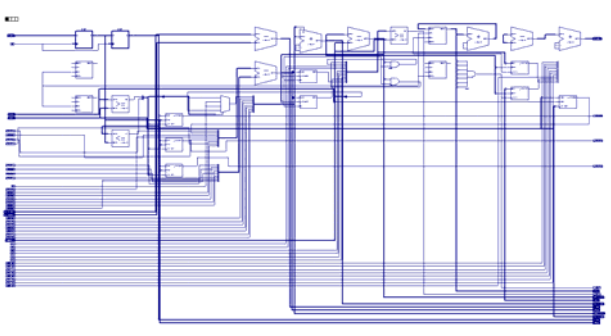
a) Bloc "Fuzzification"



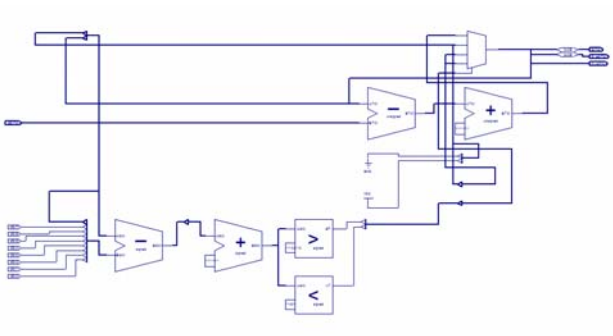
b) Bloc "Inférence"



c) Bloc "Défuzzification"



d) Bloc "Entrée"



e) Bloc "Sortie"

Figure IV.9 : Schémas équivalents des blocs constituant le "contrôleur MPPT flou optimisé"

La figure IV.10 montre le schéma équivalent du bloc VHDL "MPPT flou optimisé" dans son environnement de test. La structure des blocs de l'environnement de test reste la même et donc les schémas équivalents de ces blocs sont identiques à ceux de la méthode P&O. on voit au milieu de la figure les trois blocs principaux du contrôleur flou ("Fuzzification", "Inférences" et "Défuzzification") ainsi que les blocs secondaires ("Entrée" et "Sortie").

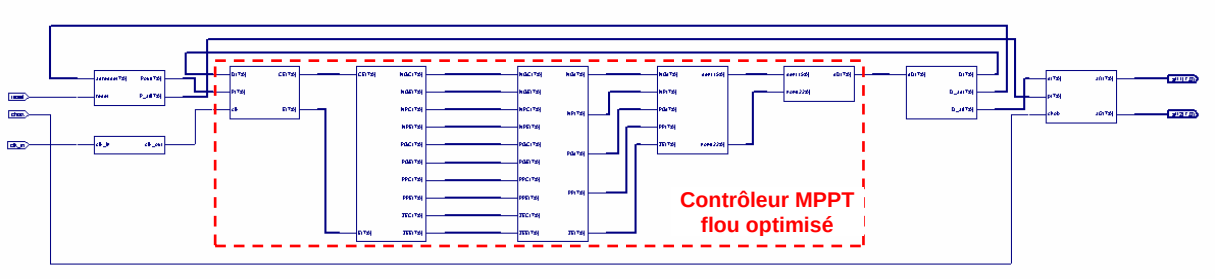


Figure IV.10 : Schéma équivalent du programme "MPPT flou optimisé"

III. Simulation

Comme on l'a déjà cité dans le chapitre III, cette opération nous permet de valider le projet à chaque étape du développement. C'est donc dans cette partie qu'on va pouvoir visualiser les résultats obtenus par les deux méthodes et comparer leurs performances. Dans la première étape, on compare les performances des deux contrôleurs dans la recherche du point MPP ; On les force à commencer la recherche à partir du même point et on calcule le temps nécessaire pour qu'ils atteignent le point de puissance maximale MPP et se stabilisent autour de lui. Dans la seconde étape, on agit sur la caractéristique puissance-tension en basculant d'une courbe à l'autre (changement brusque des conditions atmosphériques) alors que les deux contrôleurs sont stabilisés autour du point MPP, on peut alors comparer la vitesse de poursuite du point MPP pour les deux contrôleurs.

III.1. Recherche du point de puissance maximale

Dans cette étape, on initialise le contrôleur pour qu'il commence à travailler avec la valeur minimale du rapport cyclique (D=10%), on observe le comportement du contrôleur et on calcule le temps nécessaire pour qu'il atteigne le point MPP. On effectue cette opération pour chacune des deux méthodes et on peut ensuite comparer leurs résultats. La figure IV.11 montre la réponse du contrôleur "MPPT P&O" et la figure IV.12 celle du contrôleur "MPPT flou optimisé" pour des conditions constantes d'ensoleillement et de température.

On remarque sur la figure IV.11 que le pas d'incrémentation du rapport cyclique est constant (on l'a fixé à 1% de la valeur de la puissance max), le contrôleur converge donc lentement vers le point MPP (57 cycles d'horloge, ce qui est équivalent à 0,57s à 100 Hz). Une fois le point MPP atteint, la valeur de D continuera à osciller autour de la valeur idéale V_{mp} comme cela a été expliqué dans le chapitre II. Ces ondulations causent une perte d'énergie et représentent un des plus grands inconvénients de cette méthode, mais elles restent acceptables puisqu'elles ne dépassent pas les 5% de P_{max}.

Sur la figure IV.12, on voit que la réponse du contrôleur flou optimisé est beaucoup plus rapide que celle du contrôleur P&O : 14 cycles d'horloge seulement (ce qui est équivalent à 0,14s à 100 HZ). On peut remarquer que le pas d'incrémentation n'est pas constant, et que la recherche du point MPP se divise en deux phases, une phase rude où le pas dD est grand pour atteindre rapidement la zone du point MPP, et une phase fine pour affiner la recherche et se stabiliser autour du point MPP en minimisant les ondulations.

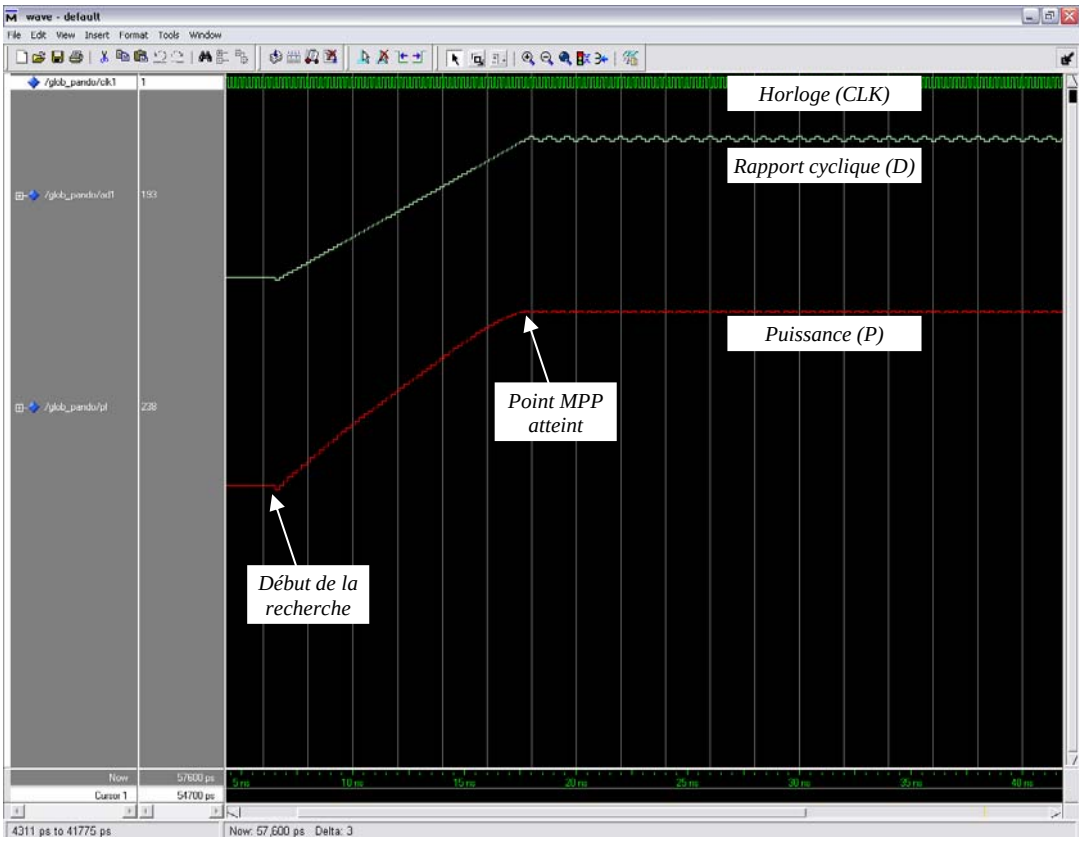


Figure IV.11 : Simulation de la recherche du point MPP par le contrôleur "MPPT P&O"

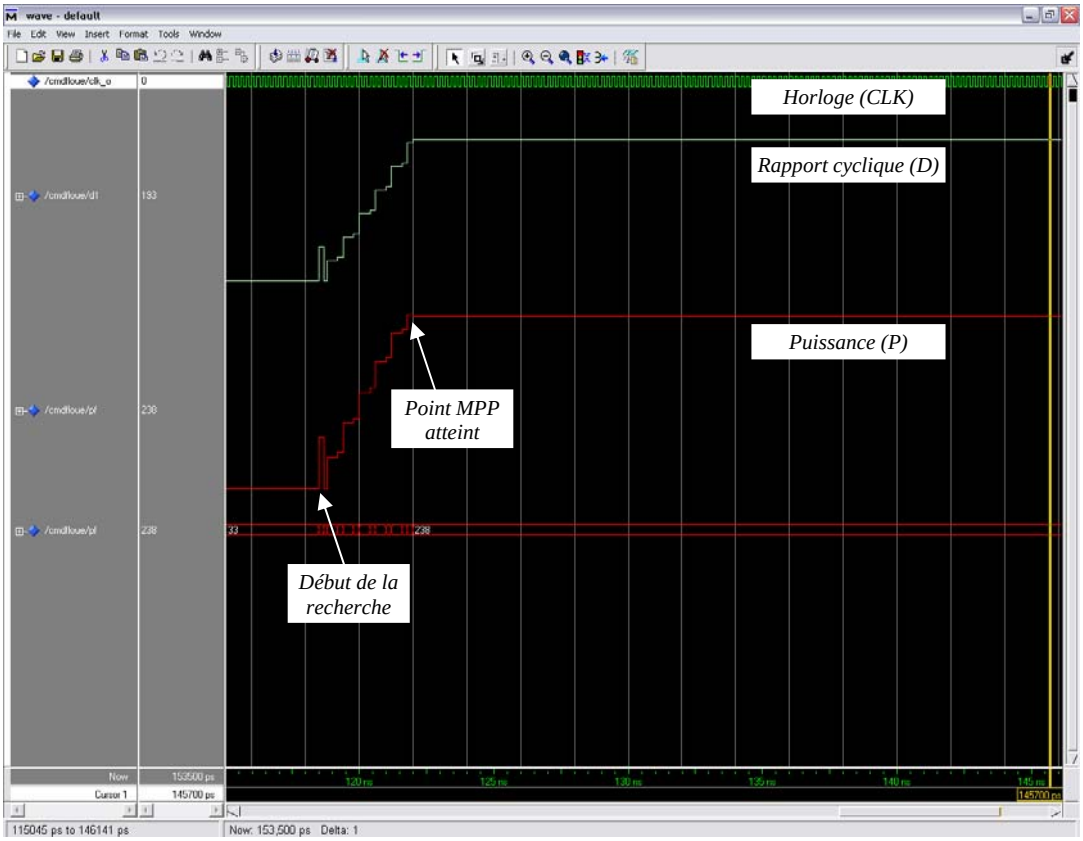


Figure IV.12 : Simulation de la recherche du point MPP par le contrôleur "MPPT flou optimisé"

III.2. Poursuite du point de puissance maximale

Dans cette seconde étape, on observe la réponse de chacun des deux contrôleurs lors de changements brusques des conditions météorologiques. On agit sur la caractéristique puissance-tension en basculant d'une courbe à l'autre (*S1* à *S2* ou *S2* à *S1* de la figure IV.5) alors que le contrôleur est stabilisé autour du point MPP, et on calcule le temps nécessaire pour qu'il atteigne le nouveau point MPP ; on peut alors comparer la vitesse de poursuite du point MPP pour les deux contrôleurs.

La figure IV.13 montre la réponse du contrôleur "MPPT P&O" pour des changements brusques de la caractéristique puissance-tension et la figure IV.14 celle du contrôleur "MPPT flou optimisé" pour les mêmes conditions changeantes.

Sur la figure IV.13, on peut voir que le contrôleur "MPPT P&O" met 8 cycles d'horloges pour trouver le nouveau point MPP lors d'une diminution brusque de l'ensoleillement (passage de la courbe *S1* à la courbe *S2*) ; et il met 10 cycles d'horloge pour trouver le nouveau MPP lors d'une augmentation brusque de l'ensoleillement (passage de la courbe *S2* à *S1*).

Pour le contrôleur MPPT flou optimisé (figure IV.14), on remarque que la poursuite du point MPP est plus rapide. En effet, lors d'une diminution brusque de l'ensoleillement, il trouve le nouveau point MPP après 3 cycles d'horloges seulement (0,03s à 100 Hz), et lors d'une augmentation brutale de l'ensoleillement, il ne met que 5 cycles d'horloge avant de se stabiliser à la valeur de P_{max}.

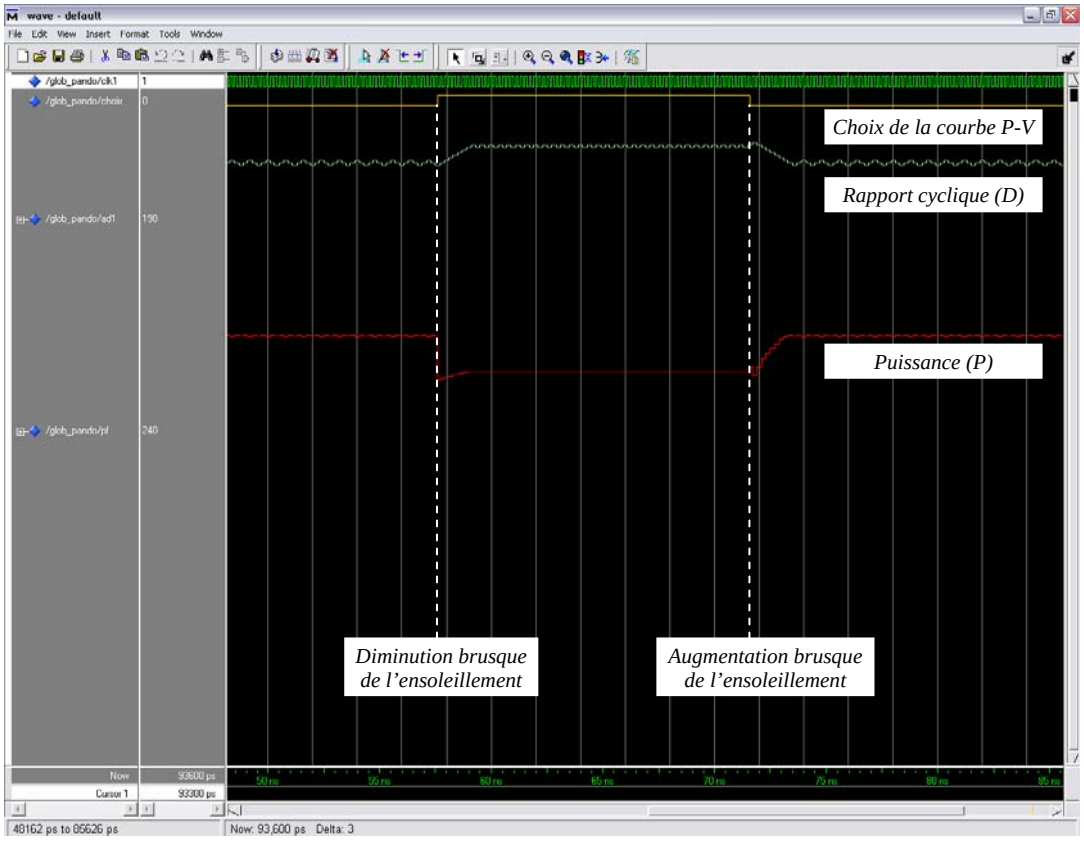


Figure IV.13 : Simulation de la poursuite du point MPP par le contrôleur "MPPT P&O" pour des changements brusques des conditions météorologiques

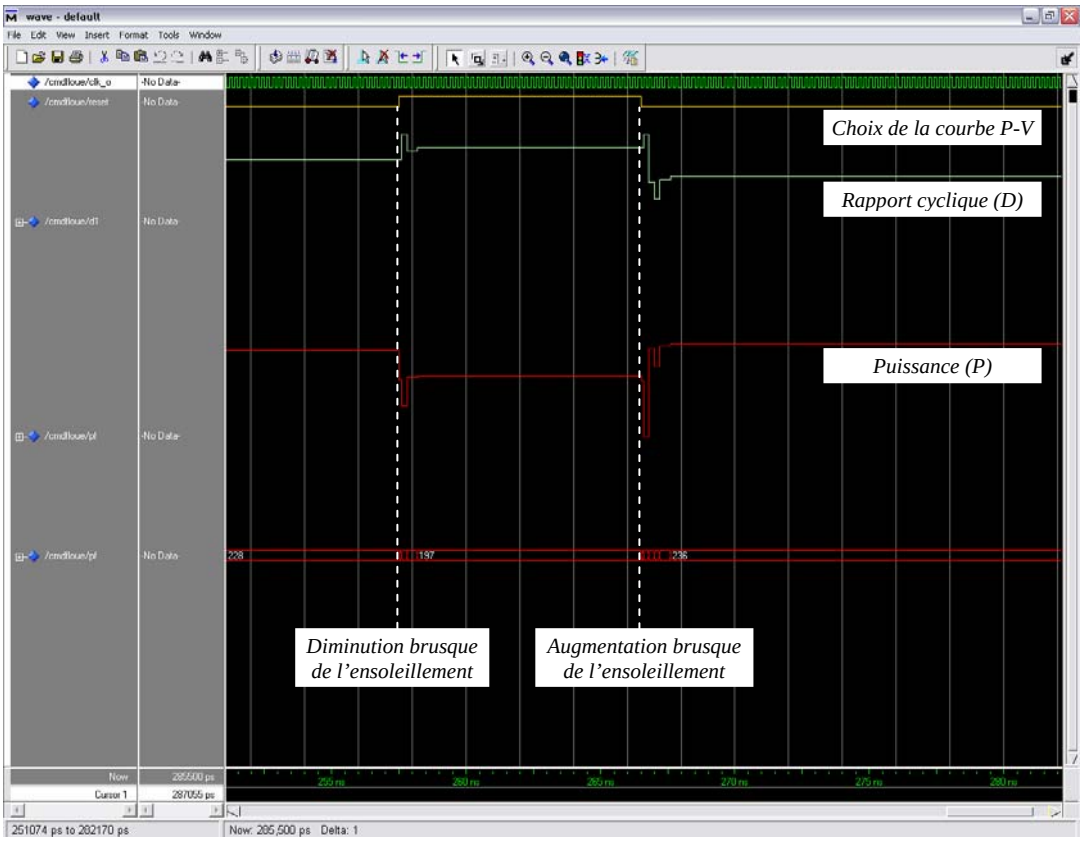


Figure IV.14 : Simulation de la poursuite du point MPP par le contrôleur "MPPT flou optimisé" pour des changements brusques des conditions météorologiques

IV. Placement et routage des programmes sur le circuit FPGA

Dans cette étape, l'outil de placement et routage trace les routes sur le circuit FPGA afin qu'il puisse réaliser le fonctionnement attendu. L'outil « FPGA Editor » nous permet de visualiser le routage réalisé sur le circuit FPGA pour chacune des deux méthodes. La figure IV.15 montre le routage du contrôleur "MPPT P&O" sur le circuit FPGA, et la figure IV.16 montre celui du contrôleur "MPPT flou optimisé". On voit que le programme du contrôleur "MPPT P&O" ne prend pas beaucoup de place sur le circuit FPGA et n'utilise pas un grand nombre de ses ressources ; par contre, le programme du contrôleur "MPPT flou optimisé" occupe pratiquement tout l'espace disponible sur le circuit FPGA car il utilise un grand nombre des ressources offertes.

Le tableau IV.1 contient les informations sur les ressources utilisées par chacun des deux programmes. Ces informations sont fournies à la fin de l'étape de synthèse.

Ressources utilisées	MPPT P&O		MPPT flou optimisé		Nombre total disponible
Bascules	179	(3%)	3597	(70%)	5120
Bascules "Flip-Flop"	91	(0%)	166	(1%)	10240
LUT à 4 entrées	274	(2%)	6809	(66%)	10240
IOB	27	(8%)	19	(5%)	324
Multiplieurs 18X18	0	(0%)	1	(2%)	40
Horloges	2	(12%)	8	(50%)	16

Tableau IV.1 : Ressources du circuit FPGA utilisées par chacune des deux méthodes

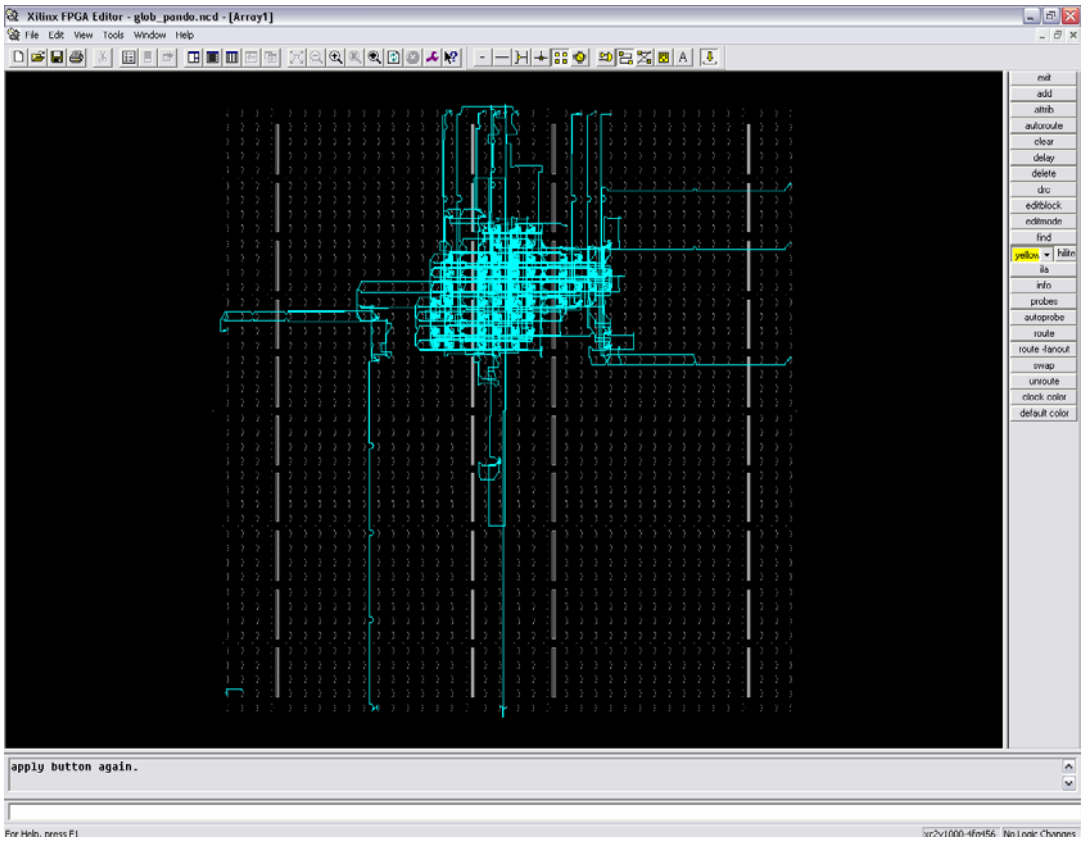


Figure IV.15 : Routage du circuit FPGA pour le programme "MPPT P&O"

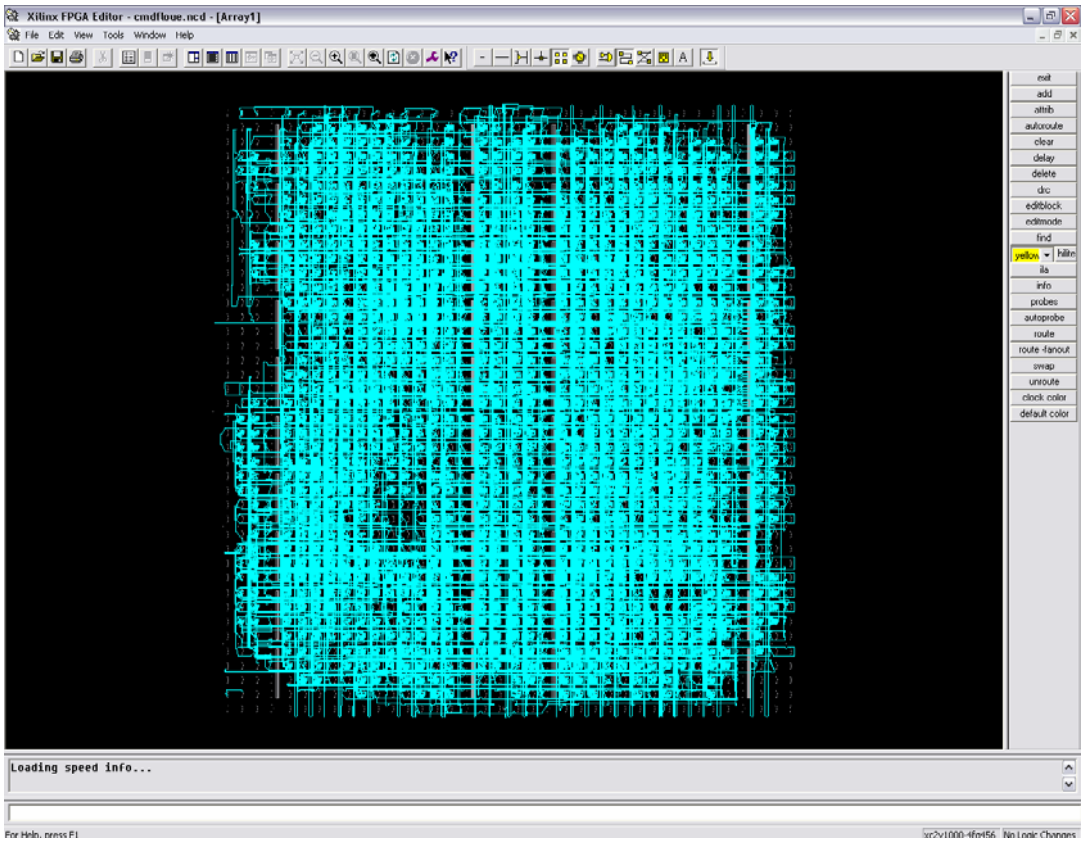


Figure IV.16 : Routage du circuit FPGA pour le programme "MPPT flou optimisé"

V. Interprétation des résultats et discussion

A partir des résultats que l'on a obtenus, on peut dire que chacune des deux méthodes MPPT implémentées sur circuit FPGA a ses avantages et ses inconvénients.

Le plus grand avantage de la méthode Perturbation et Observation est la facilité avec laquelle on peut l'implémenter, elle ne prend pas beaucoup de place sur le circuit FPGA et n'utilise qu'une petite partie des ressources offertes. Son inconvénient est la lenteur de sa réponse car elle utilise un pas constant d'incrémentation (ou de décrémentation) du rapport cyclique *D*, et une fois le point de puissance maximale atteint, l'algorithme continue à osciller autour de ce point provoquant des ondulations indésirables et des pertes de puissance. Le choix du pas d'incrémentation ΔD est un compromis à faire entre la rapidité de la réponse de l'algorithme et l'amplitude des oscillations autour de l'état stable, et le choix de la bonne valeur de ΔD n'est pas toujours évident.

L'avantage de la méthode floue optimisée par les Algorithmes Génétiques est la rapidité de sa réponse, surtout pour des variations brusques des conditions de fonctionnement. Elle converge rapidement vers le point de puissance maximale et se stabilise autour de lui en annulant quasiment les ondulations autours de l'état stable. Cependant, le plus grand inconvénient de cette méthode réside dans la complexité de son implémentation ; en effet, elle demande une programmation lourde en VHDL et occupe un espace important dans le circuit FPGA utilisé.

L'utilisation de l'une ou l'autre méthode est déterminée par l'application dans laquelle elle va être utilisée. La méthode P&O satisfait la majorité des applications communes de l'énergie photovoltaïque (consommation domestique, éclairage, etc.). Les applications de pointe (domaine spatial) font appel aux méthodes complexes ayant une meilleure efficacité telle que la méthode du contrôleur MPPT flou optimisé.

Conclusion

Conclusion

Dans ce travail, on a tenté d'explorer et d'exploiter au maximum les ressources offertes par les circuits FPGA en implémentant deux méthodes MPPT différentes utilisées dans les systèmes photovoltaïques : une méthode simple et classique (P&O), et une autre plus complexe (contrôleur flou optimisé par les Algorithmes Génétiques).

On a commencé par introduire les notions théoriques sur l'énergie photovoltaïque, la logique floue et les Algorithmes génétiques, qui sont des domaines de pointe de l'électronique moderne et qui sont nécessaires pour la compréhension du but de ce travail.

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 routage et enfin la programmation du composant.

La simulation nous a permis de valider les programmes des deux méthodes avant de les charger sur le circuit FPGA. Elle nous a également aidé à comparer les performances des deux méthodes pour différentes conditions de travail. Grâce à elle, on a pu déterminer les avantages et les inconvénients de chacune de ces deux méthodes.

Si nous devions faire un choix entre ces deux méthodes, les deux paramètres à prendre en considération sont la complexité de l'implémentation et les performances de la méthode. Si on privilégie la simplicité de l'implémentation et que l'on juge que des performances moyennes sont suffisantes, le choix sera porté sur la méthode P&O. Si on estime par contre que les performances de la méthode MPPT doivent être optimales, même pour une implémentation plus complexe, le choix sera naturellement porté sur la méthode du contrôleur flou optimisé.

Ce travail se veut une initiation au développement de projets sur circuits FPGA. On espère qu'il pourra contribuer à aider les étudiants qui s'intéressent à ce domaine de la technologie dans l'élaboration de leur travail.

Bibliographie

Bibliographie

[1] K. SOBAIHI, *Etude et réalisation d'un hacheur de tracking (MPPT) à contre réaction de tension*, Mémoire de Magister, Ecole Nationale Polytechnique, Algérie, 2003.

[2] G.F. TCHOKETCH KEBIR, *Commande des hacheurs MPPT par la logique floue*, Mémoire de Magister, Ecole Nationale Polytechnique, Algérie, 2005.

[3] J. ROYER, T. DJIAKO, E. SCHILLER et B. SADA SY, *Le pompage photovoltaïque*, Manuel de cours, Université d'Ottawa, Canada, 1998.

[4] F. CHEVRIE et F. GUELY, *La logique floue*, Cahier technique Schneider N° 191, 1998.

[5] T. OBEIDI, *Application des Algorithmes Génétiques dans la commande des hacheurs MPPT*, Mémoire de Magister, Ecole Nationale Polytechnique, Algérie, 2006.

[6] P.G. CIARLET, *Optimisation et Algorithmes Génétiques*, Manuel de cours, Université de Pierre et Marie Curie, France, 2000.

[7] J-P. DAVID, *Architecture synchronisée par les données pour système reconfigurable*, Thèse de doctorat, Université Catholique de Louvain - Faculté des Sciences Appliquées, Belgique, Juin 2002.

[8] MEMEC Design, *Virtex-II V2MB1000 Development Bord user's Guide*, http://legacy.memec.com/solutions/refernce/xilinx, Décembre 2002.

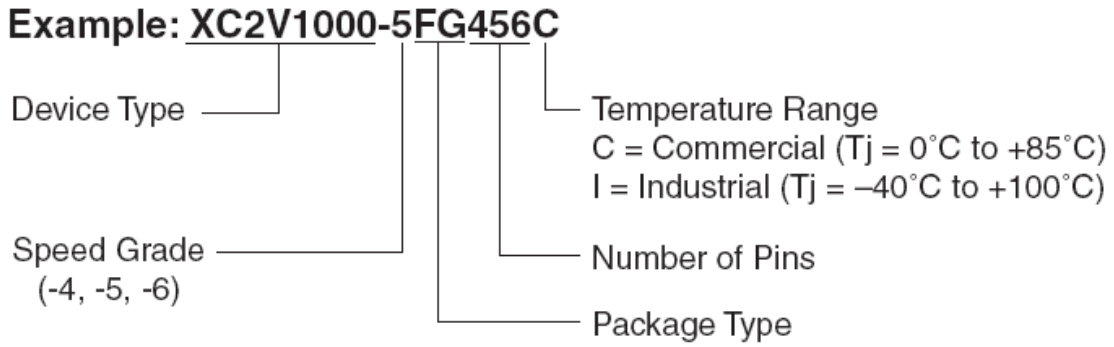
[9] XILINX, *Virtex-II Platform FPGAs : Complete Data Sheet*, http://www.xilinx.com, Mars 2005.

[10] D.GAUTHEY et E.MESSERLI, *Conception numérique : description VHDL et synthèse*, Revue scientifique "Visions", Haute Ecole Spécialisée de Suisse Occidentale, Suisse, 2000.

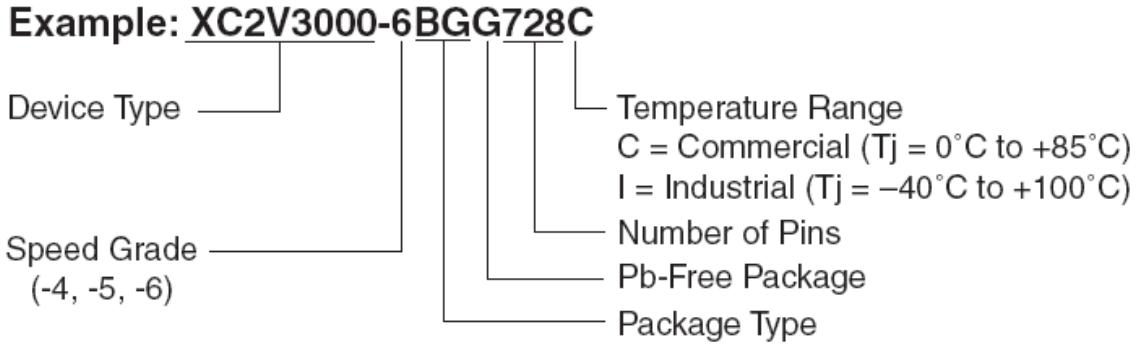
[11] P. LECARDONNEL et P. LETENNEEUR, *Introduction à la synthèse logique VHDL*, Lycée Julliot de la Morandière, Granville, Belgique, 2001.

Annexe

Annexe : Nomenclature des circuits FPGA de la famille Virtex-II



Virtex-II Ordering Example. Regular Package



Virtex-II Ordering Example. Pb-Free Package