REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département d'Electronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE LICENCE

FILIERE : **élecetrique**

Spécialité :électrique industrielle

Par

- **Bourourou mohammed**
- **Chihani bahmed**

Intitulé

• Conception d'un modulateur PWM configurable sur FPGA

Evalué le :15/09/2021

Par la commission d'évaluation composée de* :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M.	*MCB*	*Président*	*Univ-BBA*
M. S.MEGUELLATI	*…*	*Encadreur*	*Univ-BBA*
M.	*….*	*Examinateur*	*Univ-BBA*

Année Universitaire 2020/2021

REMERCIEMENTS

Nous tenons à remercier tout d'abord Allah le tout puissant qui nous a donné le courage et la patience pour arriver à ce jour et d'accomplir ce modeste travail.

Nous tenons à exprimer vivement notre profonde gratitude à notre encadreur Mm Meguellati pour sa confiance, ses encouragements, sa guidance et surtout ses judicieux conseils, qui ont contribué à alimenter nos réflexions.

Nous exprimons nos remerciements à l'ensemble des membres de jury qui nous ont fait l'honneur de juger et de participer à la soutenance de ce projet de fin d'étude.

Nous tenons aussi à exprimer nos remerciements à tous ceux qui nous ont aidés de près ou loin durant l'élaboration de notre mémoire.

Sommaire

Liste des figures

Introduction gènèral

La

modulation de largeur d'impulsion (pwm) est aujourd'hui devenue une partie intégrante de tout système électronique. Ces techniques ont été largement acceptées et font l'objet de nombreuses recherches de nos jours. Elle a trouvé son application dans un grand nombre d'applications en tant que régulateur de
tension. Son utilisation dans le contrôle de la tension de sortie de l'onduleur est l'application la plus fréquemment utilisée. Il existe essentiellement deux techniques principales de génération PWM : la technique analogique et la technique numérique.

En raison de ces inconvénients des techniques analogiques, les techniques numériques font l'objet de ce mémoire. Différentes topologies de générateurs PWM numériques ont été étudiées. Le code VHDL de chacune de ces topologies a été écrit et synthétisé à l'aide du logiciel Xilinx ISE 13.3. Une simulation comportementale a été effectuée sur l'architecture choisie.

Le présent mémoire est organisé comme suit :

Le chapitre 1 : Est consacré à quelques concepts théoriques de base sur la commande des moteurs à courant continu ainsi que la modulation PWM.

Le chapitre 2 :

Il s'agit d'une présentation consacrée au côté hardware et programmation du projet, en commençant par la structure des FPGA.

Le chapitre 3 :

Il Consacrée à la conception de carte fpga en utilisant le langage vhdl dans outil Xilinx ise.

Chapitre I : Modulation PWM et commande des moteurs à CC

I.1 Introduction

Dans cette première partie, nous présentons quelques notions théoriques, basiques, permettant de mieux comprendre le contexte et la réalisation de notre projet. La modulation PWM est maintenant une technique indispensable est quasi présente dans toutes les applications de contrôle de moteurs à courant continu. Une implémentation numérique de la PWM est alors nécessaire afin de suivre la tendance de numérisation dans le domaine d'électronique.

I.2 : La commande des moteurs à courant continu par modulation de largeur d'impulsion

Les moteurs à Courant Continu (moteur à CC) de petite ou grande puissance, sont utilisés partout dans notre vie quotidienne et ils sont intégrés dans des dispositifs divers avec des degrés de complexité différents. Les moteurs sont disponibles en différentes tailles et puissances.

I.2.1 Variation de vitesse :

La variation de vitesse d'un moteur à courant continu, s'opère par la variation de la tension d'alimentation. La solution analogique la plus simple consiste à un pont diviseur afin de varier la tension d'alimentation (pour des moteurs de petite puissance). L'inconvénient de la méthode, en plus de la nécessité d'une intervention mécanique, est les pertes thermiques aux bornes du pont diviseur. Une solution numérique se résume dans la génération d'un signal dont la valeur moyenne est modifiable. La valeur moyenne d'un signal rectangulaire S(t) de période T, tel que t_{on} est la durée de l'état haut du signal, est donnée par l'intégral :

$$Vmoy = \frac{1}{T}\int_0^T S(t)dt = \frac{1}{T}\left(\int_0^{t_{on}} 1.E.dt + \int_{t_{on}}^{T} 0.dt\right) =$$

Le rapport $\alpha = \frac{t_{on}}{T}$ est connu par le nom « rapport cyclique ».

On voit que la valeur moyenne Vmoy = αE, varie entre 0 (α=0) et E (α=1). La vitesse de rotation du moteur est proportionnelle à la valeur moyenne Vmoy, donc au paramètre α.

- α=0 : Moteur en arrêt (vitesse nulle)
- α=1 : Moteur en pleine puissance (vitesse maximale)

I.2.2 Variation de sens de rotation :

La variation du sens de rotation se fait en inversant la polarité de la tension de commande du moteur à CC. Dans une solution numérique, il suffit de générer un signal complémentaire. Pratiquement, les signaux de commande sont appliqués au moteur à travers un circuit à base de transistor appelé : pont en H (H bridge). Dans la figure suivante, lorsque les interrupteurs SW2 et SW3 sont fermés, le moteur tourne au sens des aiguilles. Lorsque SW1 et SW4 sont fermés le moteur tourne dans le sens contraire.

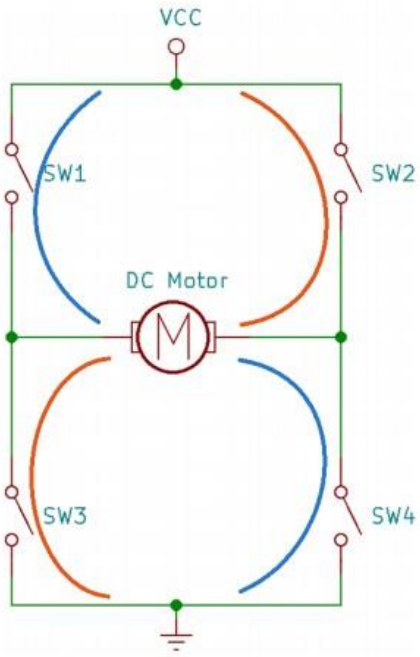


Figure I.1 : *un moteur à courant continu commandé à travers un pont en H*

I.3 La modulation de largeur d'impulsion (PWM pulse width modulation)

La modulation de largeur d'impulsion connue aussi par son acronyme anglais PWM, est une technique qui permet de générer un signal rectangulaire dont la largeur de l'impulsion (tension haute) est variable. Sa stratégie se résume donc sur le changement du paramètre α (rapport cyclique).

De nombreuses techniques numériques sont basées sur l'utilisation d'une conception à base de compteurs et des comparateurs. Ces techniques numériques sont plus faciles à mettre en œuvre que les techniques analogiques, de plus, ils sont immunisés contre le bruit ambiant et le

changement de température et ne souffrent pas de la dérive des composants et des pertes de commutation. Pour les schémas de contrôle sophistiqués, il est souhaitable d'utiliser Schéma de modulation PWM numérique.

Il existe de nombreuses techniques numériques disponibles selon la disposition et le type de compteur utilisé, mais dans ce chapitre, trois topologies principales de générateur PWM sont discutées. Ce sont (a) Générateur PWM basé sur un compteur de Haute fréquence (b) Générateur PWM basé sur un compteur (c) Générateur PWM à compteur en cascade.

I.3.1 Générateur PWM basé sur un compteur haute fréquence :

Cette architecture a été proposée par E.Koutroulis A.Dollas et K.Kalaitzakis dans [1]. Selon cette architecture, il existe un compteur à course libre à N-bits à grande vitesse dont la sortie est comparée à la sortie du registre, qui stocke le rapport cyclique souhaité (valeur N-bits), à l'aide d'un comparateur. La sortie du comparateur est mise à 1 lorsque ces deux valeurs sont égales. Cette sortie de comparateur est utilisée pour régler le verrou RS. Le signal de débordement du compteur est utilisé pour réinitialiser le verrou RS. La sortie du verrou RS donne la sortie PWM souhaitée. Ce signal de débordement est également utilisé pour charger un nouveau rapport cyclique N bits dans le registre.

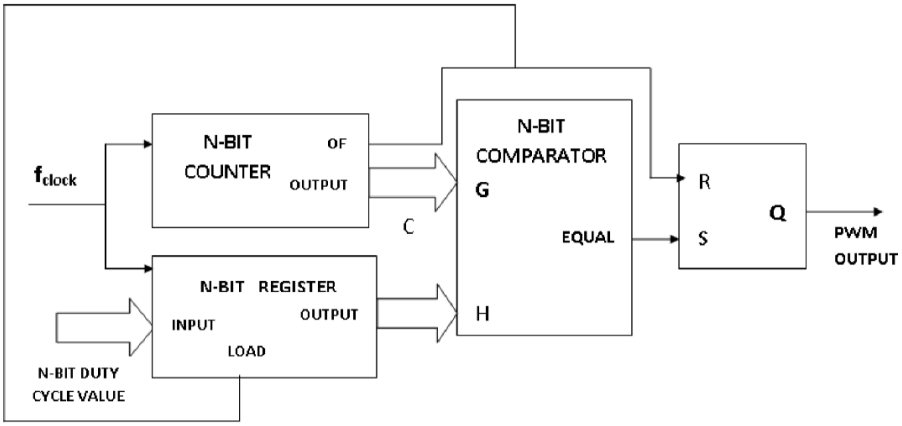


Figure I.2 *Architecture du générateur PWM proposée par E. Koutroulis et al. en [1]*

L'avantage de cette méthode est qu'elle est utilisée pour générer une sortie PWM haute fréquence, ce qui n'est pas possible dans une approche normale basée sur un compteur. La figure 2 montre le schéma fonctionnel correspondant de cette architecture.

I.3.2 Générateur PWM à base de compteur :

Cette architecture de générateur PWM est utilisée pour les alimentations à découpage de faible puissance. Cette architecture a été proposée par A.P.Dancy , R.Amirtharajah et A.P. Chandrakasan dans [8]. L'architecture représentée sur la figure 3 est basée sur le principe qu'en raison du déclenchement d'un compteur par un signal d'horloge, l'horloge est réglée égale à un certain multiple de la fréquence de commutation à l'aide d'un compteur. Le signal de sortie PWM est mis à l'état haut avant le signal d'horloge et il reste à l'état haut jusqu'à ce qu'il soit réinitialisé après que la valeur du compteur devienne égale à la valeur du rapport cyclique. La figure 3 montre l'architecture définie ci-dessus.

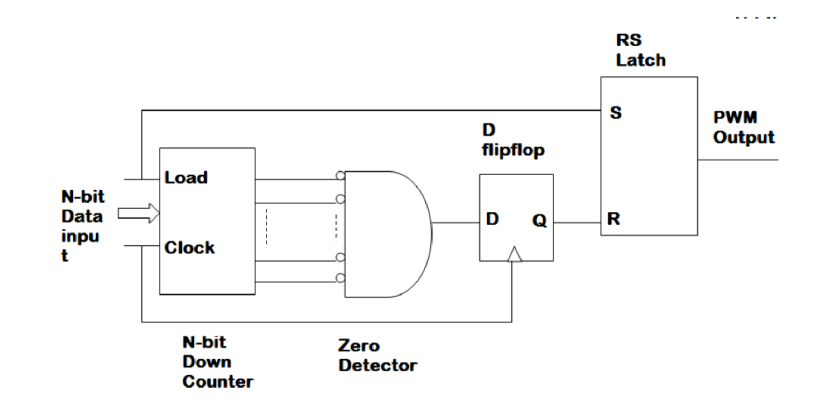


Figure I.3 *Architecture du générateur PWM à base de compteur proposé par A.P Dancy et al. dans [8].*

I.3.3 Architecture de générateur PWM à base de compteurs en cascade :

Dans cette architecture, deux compteurs de 4-bits sont mis en cascade pour former 8-bits. Un signal d'horloge est appliqué aux deux compteurs et la sortie de ces deux compteurs est connectée à la broche d'entrée A 8 bits du comparateur. La rapport cyclique, codé sur 8-bits, est appliqué à la deuxième entrée B du comparateur. La sortie du comparateur est « 1 » lorsque la valeur de la broche A et la valeur de la broche B coïncident. Cette sortie du comparateur est utilisée pour réinitialiser le verrou RS. Un signal de dépassement du compteur MSB supérieur est utilisé pour définir le verrouillage RS chaque fois qu'il est activé. La sortie du verrou RS est utilisée pour donner une sortie PWM.

I.4 Conclusion

Dans cette partie, nous avons appris comment changer la vitesse du moteur et comment le contrôler, et nous avons également amélioré les trois types de PWM, Modulation de largeur

d'impulsion fonction contrôle l'onduleur de tension ou de courant à partir du bloc de puissance du variateur. Chaque performance du système est influencée par le PWM qui devient donc un élément essentiel de la chaine.

Chapitre II : Circuits Configurables FPGA

II.1 Introduction

Ce chapitre présente la plateforme hardware visée par le projet ; il décrit les FPGAS ainsi que les principaux composants et éléments qui les caractérisent. On introduit le langage à description matérielle VHDL et le processus de conception d'application à l'aide de FPGA , le flot de conception. Pour terminer le chapitre, on décrit brièvement la plateforme matérielle spécifique adoptée dans le cadre de notre travail.

II .2 Architecture de base d'un FPGA

II.2.1 Définition :

Un circuit FPGAs, Un réseau de portes programmables sur champs (Field Programmable Gate Array), comme son nom l'indique, est un ensemble de cellules logiques (ou modules) et d'interconnexions, qui peuvent être reprogrammés en fonction des besoins de l'utilisateur. Nous pouvons le concevoir et y apporter des modifications chaque fois que nécessaire. Il offre un délai de fabrication instantané et des coûts de prototype négligeables, ce qui le rend adapté à la conception de systèmes embarqués.

II.2.2 Architecture interne des FPGAs :

Organisés sous forme de matrices, les FPGA sont composés d'éléments logiques de base, constitués de portes logiques, présentes physiquement sur le circuit. Ces portes sont reliées par un ensemble d'interconnexions modifiables : d'où l'aspect programmable du circuit.
L'architecture générale d'un FPGA est la suivante :

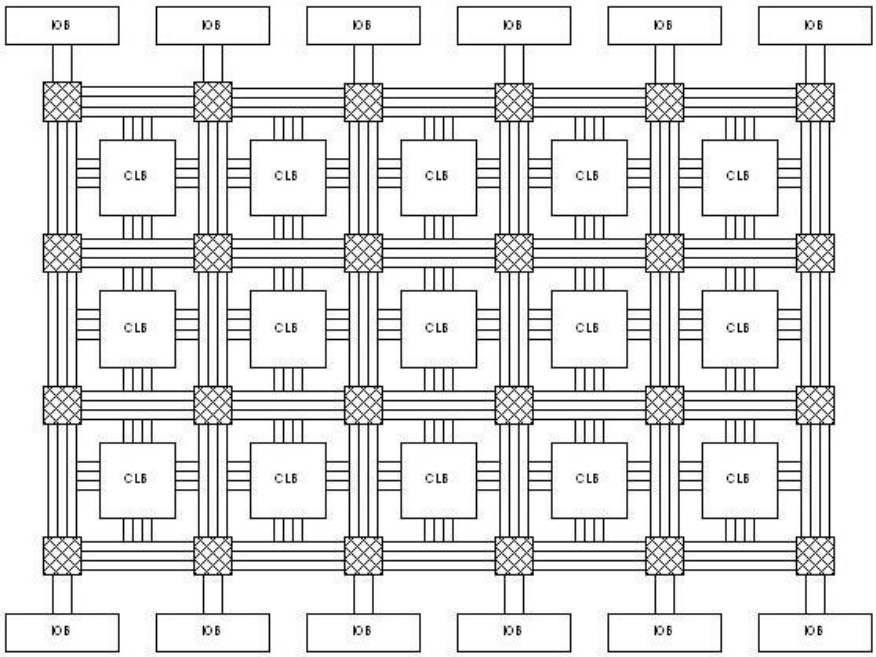


Figure II.1: *Architecture de base d'un FPGA.*

La structure du FPGA présentée dans la figure est composée :

• De blocs logiques ou éléments logiques contenant les fonctions logiques combinatoires et séquentielles.

La partie combinatoire permet de réaliser des fonctions de complexité moyenne avec des portes classiques ET, OU et NON de deux à une dizaine d'entrées.

La partie séquentielle comporte une ou deux bascules généralement de type D. Compte tenu du nombre d'éléments logiques et de leur structure, leur association permet de réaliser tous les types de bascule. L'intérêt est de créer des mémoires élémentaires à un bit.

• De réseaux d'interconnexions que l'on voit en Figure. Ces réseaux relient entre eux les blocs logiques et les blocs d'entrées/sorties. Ces connections peuvent directement relier :

• Des éléments internes dans un bloc grâce à un système de tables logiques appelées LUT. C'est une matrice de connections où les points de routage déterminent le niveau des entrées soit haut soit bas des portes logiques.

• Des éléments proches : on parle de liaisons directes entre les blocs.

• Plusieurs blocs présents sur toute la surface :on parle de liaisons à distance ou générales.

• De cellules d'entrées sorties modifiables qui servent d'interfaces entre les broches du circuit et le cœur du FPGA pour adapter les signaux suivants :

II.2.3 Applications du FPGA :

Les FPGA ont été rapidement acceptés au cours des dernières décennies. Voici quelques-unes des domaines d'applications des FPGA

• Les FPGA sont utilisés dans des machines informatiques personnalisées.

• Les FPGA fournissent une combinaison unique de calcul personnalisé hautement parallèle et de calcul à faible coût.

• Les FPGA sont utilisés dans plusieurs applications nécessitant l'électronique numérique telles que l'aéronautique, les télécommunications, le transport…

II.3 Langage de description matérielle VHDL

Un langage de description matériel (Hardware Description Language, HDL) est une instance d'une classe de langage informatique ayant pour but la description formelle d'un système électronique. C'est une représentation textuelle d'un comportement temporel ou d'une structure d'un circuit. En comparaison avec un langage de programmation classique, la syntaxe et la sémantique des HDL incluent des notations pour exprimer la concurrence et le temps, qui sont les principaux attributs du matériel.

II.4 Flot de conception FPGA

Le flot de conception est un terme qui désigne l'ensemble des étapes, standardisées, que doit suivre toutes conception numérique visant une FPGA comme cible d'implémentation.

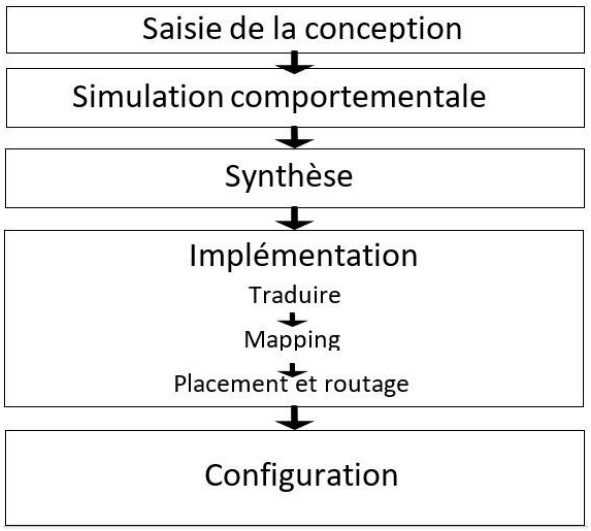


Figure II.2 Flot de conception FPGA

La figure II.2 illustre schématiquement le flot de conception. Ces étapes sont décrites la section suivante.

II.4.1 Saisie de conception :

C'est la première étape de l'implémentation d'une conception sur FPGA. Dans cette étape, la conception est passée comme entrée à l'outil CAO sous forme d'un code HDL (VHDL ou Verilog), sous forme schématique ou sous forme mixte (code HDL et schématique). Lors de cette étape, l'outil CAO réalise une première vérification de la conception (syntaxe du code et intégrité du schématique) qui doit être libre de toute erreur avant de passer à la simulation.

II.4.2 Simulation comportementale (avant synthèse):

Cette étape vérifie si la conception saisie est fonctionnellement correcte ou non. Cette simulation est appelée simulation RTL. Cette simulation est « logique » seulement ; c'est-àdire qu'elle ne tienne pas compte des temps de propagation et de retards.

II.4.3 Synthèse :

Le processus de synthèse est utilisé pour optimiser l'architecture de conception sélectionnée. Après la synthèse de conception, un rapport de synthèse est généré et fournit des informations sur le nombre de blocs logiques utilisés et sur l'utilisation du périphérique de l'architecture de conception synthétisée. La synthèse fait correspondre essentiellement la conception comportementale à la conception au niveau de la porte. Remarquons que les code VHDL ne sont pas toujours synthétisables, c'est pourquoi il faut faire attention au style du codage pour assurer l'implémentation finale.

II.4.4 Implémentation :

Après la synthèse de la conception, la mise en œuvre de la conception est effectuée et comprend les trois étapes suivantes :

(a) Traduire (Translate)

(b) Mappage (Map)

(c) Placement et routage (Place and route)

Avant de traduire la conception, un fichier contraint par l'utilisateur (UCF) est écrit pour affecter la configuration des broches du FPGA aux E/S de la conception. Une fois cela fait, Translate

fusionne ce fichier UCF et la netlist générée après la synthèse dans le fichier de conception Xilinx.

Le mappage est effectué pour adapter la conception aux ressources disponibles du périphérique cible, c'est-à-dire le FPGA. C'est aussi une étape importante de la conception.

La dernière étape de la mise en œuvre de la conception est le placement et le routage, qui place les blocs logiques de la conception dans le FPGA et les achemine ensemble afin qu'ils occupent une surface minimale et répondent aux exigences de synchronisation. Cette opération produit un fichier de sortie NCD.

II.4.5 Simulation temporelle (post synthèse):

Etant réalisé après l'implémentation, toutes les données concernant les temps propagation et retards induits par les différents éléments logiques ainsi que les retards dus aux interconnections, cette simulation permet une meilleure vérification de fonctionnement et est la plus proche de la réalité du circuit.

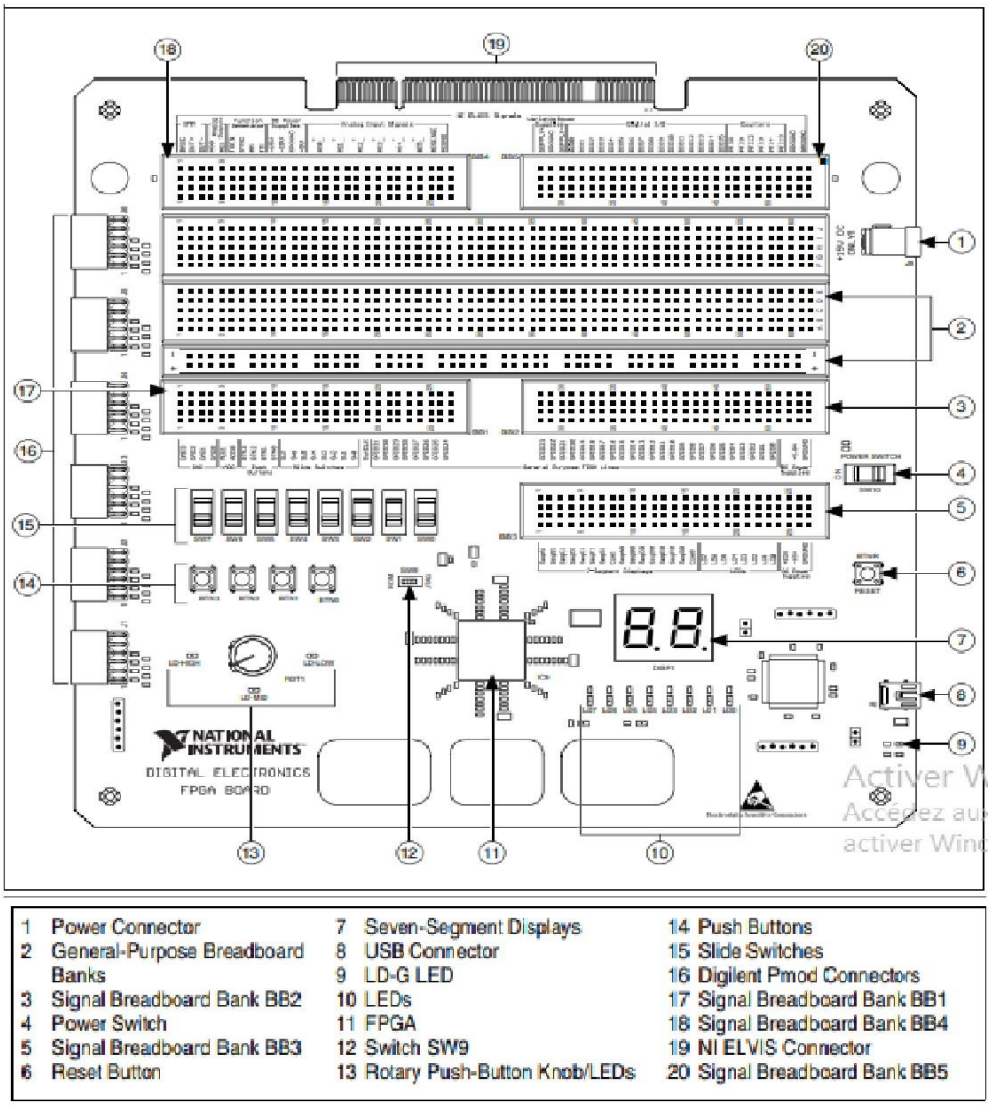
II.4.6 Configuration du dispositif Xilinx (FPGA) :

En utilisant les fichiers issus des trois étapes d'implémentation, l'outil CAO génère un fichier binaire (fichier.bit) permettant la configuration de l'FPGA.

II.5 Plateforme matérielle NI Digital Electronics FPGA Board

La plateforme matérielle utilisée dans le cadre de ce projet est NI Digital Electronics FPGA Board. C'est une plateforme de développement de circuit basée sur la FPGA XC3S500E Xilinx Spartan-3E. Outre la FPGA, la carte contient des commutateurs à glissière, des voyants, un afficheur à sept segments à deux chiffres, des boutons poussoirs, un bouton-poussoir rotatif et des voyants pour une horloge externe, une interface de téléchargement USB et une grande surface de travail expérimentation de circuits électroniques.

La figure suivante illustre le diagramme de référence de la vue de dessus de la carte NI Digital Electronics FPGA.



1	Power Connector	7	Seven-Segment Displays	14	Push Buttons
2	General-Purpose Breadboard Banks	8	USB Connector	15	Slide Switches
		9	LD-G LED	16	Digilent Pmod Connectors
3	Signal Breadboard Bank BB2	10	LEDs	17	Signal Breadboard Bank BB1
4	Power Switch	11	FPGA	18	Signal Breadboard Bank BB4
5	Signal Breadboard Bank BB3	12	Switch SW9	19	NI ELVIS Connector
6	Reset Button	13	Rotary Push-Button Knob/LEDs	20	Signal Breadboard Bank BB5

Figure II.3: Plateforme matérielle NI Digital Electronics FPGA.

II.6 Conclusion

Ce second chapitre englobe le côté hardware et programmation du projet, en commençant par la structure des FPGAs, ensuite, on a introduit le processus de conception d'applications FPGA Conception utilisant Simulink (Xilinx System Generator) et enfin, on a présenté la plateforme matérielle NI Digital Electronics FPGA Board ainsi que la FPGA inclus dedans.

Chapitre III : Conception, simulation et implémentation sur FPGA

Ⅲ.1 Introduction

Dans cette partie nous allons présenter les différentes étapes suivies afin de concevoir un modulateur PWM. Nous le faisons tout en précisant les outils spécifiques utilisés. Avant d'entamer l'implémentation, rappelons les objectifs de notre projet : conception et implémentation sur FPGA d'un modulateur PWM visant comme application la commande d'un moteur à courant continu. La suite du projet, sera consacrée à la conception d'un composant numérique qui permet de varier le rapport cyclique α du signal, donc la vitesse de rotation du moteur, ainsi que le sens de rotation de manière reconfigurable.

Ⅲ.2 Conception du modulateur (générateur de signaux) PWM

Ⅲ.2.1 Choix de conception :

Outil de conception : comme l'FPGA disponible dans notre laboratoire est un composant Xilinx, on utilisera naturellement l'outil Xilinx ISE (version 13) à travers toutes les étapes de conception. Pour la simulation, c'est le simulateur intégré ISim.

Méthode de saisie : Pour commencer, nous avons choisi de concevoir en utilisant le langage VHDL, qui présente un certain nombre d'avantages par rapport au schématique. Notamment, une description sous forme de code est plus facilement rendue générique est configurable (il suffit de modifier un paramètre generic pour modifier par exemple la taille des mots binaires.).

L'approche de conception : est de type **Buttom-Up**, c'est-à-dire partant des éléments de base et montant la hiérarchie jusqu'au niveau haut (top level).

Synchronisation : La conception est synchrone, c'est-à-dire toutes les interactions (remise à zéro, activation, fonctionnement normal) ont lieu avec un front d'horloge (positif dans notre cas).

Codage des données : Pour notre conception sur FPGA nous avons choisi de coder les signaux d'entrée et de sortie sur 8-bits. Ceci réalise un bon compromis entre la précision du signal généré (son rapport cyclique) et les ressources de l'FPGA, limitées surtout du côté broches I/O.

Architecture : Pour notre modulateur PWM, nous avons choisi l'architecture à base de compteur à haute fréquence, qui utilise un seul compteur. Le schéma de base est le suivant :

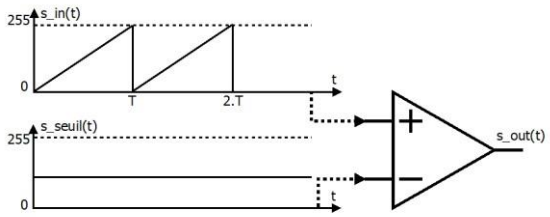


Figure Ⅲ.1 : schématique synoptique du modulateur PWM à base de compteur HF

Ⅲ.2.2 Principe de fonctionnement :

On considère un signal triangulaire s_in(t) code sur 8 bits non signé (donc compris entre 0 - 255). On définit une variable fixe qui représente un seuil s_seuil(t), codée sur 8 bits et qui peut prendre une valeur parmi les 256 valeurs. Les figures ci-dessous, montrent la sortie du comparateur s_out(t) pour les différentes valeurs de seuil.

- Graphe 1 : Seuil = 255
- Graphe 2 : Seuil = 200
- Graphe 3 : Seuil = 64
- Graphe 4 : Seuil = 0

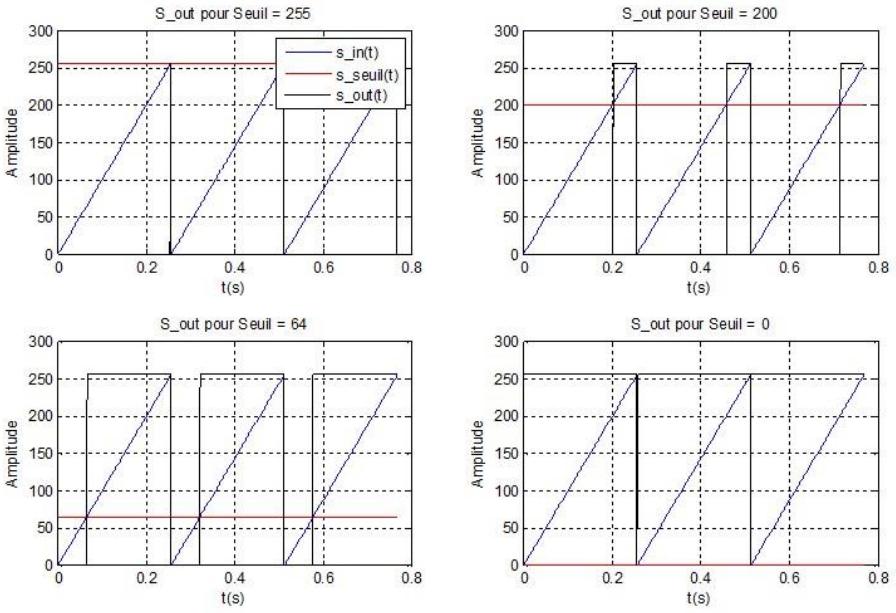


Figure Ⅲ. 2 : *Sorties du comparateur pour différentes valeurs du seuil et une entrée triangulaire*

On constate que la largeur d'impulsion est minimale pour un seuil de 255 (les conditions du comparateur : « ≥ seuil »). Un seuil décroissant a par conséquence une augmentation linéaire de la largeur d'impulsion. Pour augmenter la vitesse du moteur, il faut donc baisser le seuil de déclenchement.

Ⅲ.3 Conception VHDL et simulation des blocs de base :

Le synoptique représenté par la figure II.1 peut être partitionné en trois blocs : génération du signal triangulaire, comparateur numérique et enfin un bloc pour générer la valeur numérique représentant le rapport cyclique.

Ⅲ.3.1 Bloc génération de signal triangulaire :

La version numérique d'un signal triangulaire continu est un signal en escalier. Plus il y a de marches, plus proche du cas contenu et donc meilleur est le signal triangulaire. Le moyen numérique permettant la génération d'un tel signal est un compteur numérique. Un compteur de 8-bit permet donc de générer un signal triangulaire ayant $2^8 = 256$ marches.

Un compteur numérique est un système séquentiel et nécessite donc des signaux de synchronisation, à savoir : l'horloge Clk, la remise à zéro Rst, l'activation En. La description

VHDL peut être de type structural à base de bascules, qui traduirait mieux l'architecture réelle d'un compteur, mais le VHDL a l'avantage d'offrir des instructions séquentielles synthétisables, ce qui permet d'opter pour une description comportementale.

L'entité du bloc générateur de signal triangulaire est la suivante :

```
entity GenTriang is

   Port ( Rst : in  STD_LOGIC;
          En : in  STD_LOGIC;
          Clk : in  STD_LOGIC;
          Triag_out : out  STD_LOGIC_VECTOR (7 downto 0));
end GenTriang;
```

Le process au cœur du compteur est le suivant :

```
P0 : process(Rst, Clk, En )
begin
            if Clk = '1' and Clk'event then
                  if Rst ='1' then
                        TriagG_tmp <= (others =>'0');
else
                        if En = '1' then
                              TriagG_tmp <= TriagG_tmp+1;
else
                              TriagG_tmp <= TriagG_tmp;
end if;                     end if;              end if;
end process P0;
```

La vérification par simulation nécessite l'écriture d'un code VHDL de test, dit « testbench », permettant la génération des signaux de stimuli. La simulation fonctionnelle réalisée dans ISim a donné les chronographes de fonctionnement suivant.

On voit que la sortie atteint 255 ensuite reboucle le comptage à partir de 0.

Figure Ⅲ.3: chronogrammes de simulation du générateur de signaux triangulaires

Ⅲ.3.2 Bloc Comparateur numérique :

La comparaison numérique entre deux mots 8-bits est modélisée à travers une description VHDL comportementale synthétisable. L'entité est la suivante :

```
entity Compar is

   Port ( Rst : in  STD_LOGIC;
          En : in  STD_LOGIC;
          Clk : in  STD_LOGIC;
          D_in_P : in  STD_LOGIC_VECTOR (7 downto 0);
          D_in_N : in  STD_LOGIC_VECTOR (7 downto 0);
          D_out_CMP : out  STD_LOGIC); end
Compar;
```

Le process correspondant est le suivant :

```
       P1 : process(Rst, Clk, En )
begin
            if Clk = '1' and Clk'event then
                  if Rst ='1' then
                        D_out_CMP_tmp <= '0';
                  else
                        if En = '1' then
if D_in_P > D_in_N then
D_out_CMP_tmp <= '1';                                  else
                                    D_out_CMP_tmp <= '0';
end if ;
                        else
                              D_out_CMP_tmp <= D_out_CMP_tmp;
end if;                    end if;              end if;        end process
P1;
```

La simulation fonctionnelle a donné les résultats suivants :

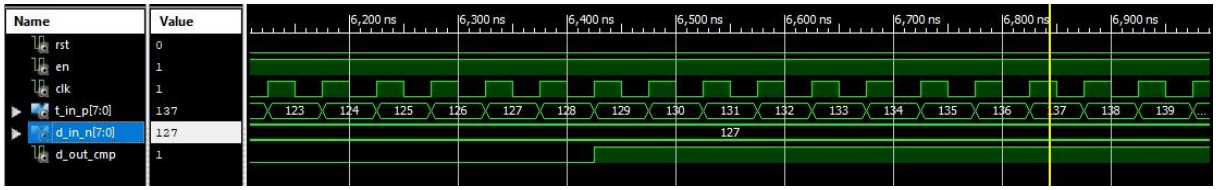


Figure III.4 **:** Chronogrammes de simulation du Comparateur numérique 8-bits

On a utilisé le bloc générateur triangulaire précédent pour générer le signal appliqué à l'entrée positive du comparateur. On voit que la sortie passe à '1' lorsque le signal triangulaire dépasse le seuil, ici égale à « 01111111 » ou 127 en décimal. Le retard d'un cycle d'horloge est dû au fait que les deux composants partagent le même signal d'horloge.

Ⅲ.3.3 Bloc générateur de rapport cyclique configurable :

La conception étant numérique, et afin de la rendre plus versatile, on a choisi de permettre la modification du rapport cyclique en proposant un ensemble de valeurs prédéterminées permettant à l'utilisateur de modifier la vitesse selon les valeurs de rapport cycliques correspondants. Dans notre conception on a choisi quatre valeurs qui permettent de faire graduer la vitesse du moteur entre faible et forte vitesse.

Afin de générer ces valeurs prédéterminées, on a écrit un code VHDL qui, à chaque impulsion sur son entrée horloge, génère un mots binaires ajouté de 63 à sa valeur précédente. Entre les 256 valeurs possibles, se bloc génère donc : 63, 126, 189 et 252.

L'entité VHDL est la suivante :

```
entity RapportC is

Port ( Rst : in  STD_LOGIC;
En : in  STD_LOGIC;
alphac : in  STD_LOGIC;
        alpha_out : out  STD_LOGIC_VECTOR (7 downto 0)); end
RapportC ;
```

Le process correspondant est le suivant :

```
        P_alpha : process(Rst, alphac, En )
begin
            if alphac = '1' and alphac'event then
                  if Rst ='1' then
                        tmp <= "00011111";
                                                                  if  tmp =
                                                          "11111100" then
                                                                  tmp <=
"00011111";
                                                                  end if;
                       else
                        if En = '1' then
tmp <= tmp+63;                              else
                                  tmp <= tmp;
                        end
if;                    end if;
end if;
      end process P_alpha;
```

La simulation a donné comme résultat :

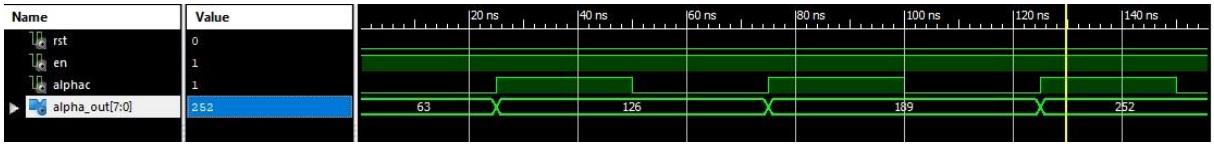


Figure Ⅲ.5: Chronogrammes de simulation du générateur de rapport cyclique configurable

Ⅲ.3.4 Bloc générateur d'horloge (diviseur programmable d'horloge) :

Le plus souvent, dans les cartes de développement FPGA, il existe un quartz qui assure un signal d'horloge très stable et de fréquence plus ou moins grande selon le modèle. Pour accommoder notre conception à toute situation pratique concernant la fréquence, on a conçu un générateur d'horloge configurable qui permet de diviser la fréquence d'entrée (quelle que soit sa valeur) par un facteur que l'on définit dans le code. Dans notre code ce facteur diviseur est défini par un signal temporaire Clk_out_tmp qui peut prendre des valeurs entre 1 et 16. Sachant que la carte utilisée possède un quartz de 50MHz, nous avons choisi un facteur de division égale à 16, ce qui donne une fréquence de fonctionnement de 3.12 MHz.

L'entité du générateur d'horloge est la suivante :

```
entity GenClk is

Port ( Clk_in : in  STD_LOGIC;
           Rst : in  STD_LOGIC;
       En : in  STD_LOGIC;
           ClkSel : in  STD_LOGIC_VECTOR (3 downto 0);
       Clk_out : out  STD_LOGIC); end
GenClk ;
```

le process correspondant est :

```
      P1 : process(Rst, Clk_in, En, ClkSel_tmp )
begin
            if Clk_in = '1' and Clk_in'event then
                  if Rst ='1' then
                        Clk_out_tmp <= '0';
else
                        if En = '1' then
       --- Sélection d'une sortie du compteur (division par
2^N)  case ClkSel_tmp is when x"0"        => Clk_out_tmp
<= Count_tmp(1); when x"1"        => Clk_out_tmp <=
Count_tmp(2); when x"2"        => Clk_out_tmp <=
Count_tmp(3); when x"3"        => Clk_out_tmp <=
Count_tmp(4); when x"4"        => Clk_out_tmp <=
Count_tmp(5); when x"5"        => Clk_out_tmp <=
Count_tmp(6); when x"6"        => Clk_out_tmp <=
Count_tmp(7); when x"7"        => Clk_out_tmp <=
Count_tmp(8); when x"8"        => Clk_out_tmp <=
Count_tmp(9); when x"9"        => Clk_out_tmp <=
Count_tmp(10); when x"A"       => Clk_out_tmp <=
Count_tmp(11); when x"B"       => Clk_out_tmp <=
Count_tmp(12); when x"C"       => Clk_out_tmp <=
Count_tmp(13); when x"D"       => Clk_out_tmp <=
Count_tmp(14); when x"E"       => Clk_out_tmp <=
Count_tmp(15); when x"F"       => Clk_out_tmp <=
Count_tmp(16); when others => Clk_out_tmp <=
Count_tmp(0); end case ; else
Clk_out_tmp <= Clk_out_tmp;
end if;end if; end if; end
process P1;
```

Ⅲ.3.5 Bloc générateur (modulateur PWM) :

On rassemble les trois blocs précédents afin de réaliser un générateur de signaux PWM qui permet la variation de la vitesse à travers la variation du rapport cyclique (4 valeurs possibles dans notre cas). Afin de pouvoir commander le sens de rotation du moteur, il suffit de pouvoir générer un signal complémentaire au signal pwm précédent. L'idée est d'appliquer le signal triangulaire et le seuil aux entrées positive et négative d'un premier comparateur et de les appliquer simultanément aux entrées négative et positive d'un deuxième comparateur. A tout instant, on aura deux sortie PWM complémentaires.

L'entité du code VHDL est la suivante :

```
entity CmdPWM is
```

```
Port ( Rst : in  STD_LOGIC;          alphac,  Clk : in  STD_LOGIC;
         En , LR: in  STD_LOGIC;
        PWM_out1,  PWM_out2 : out  STD_LOGIC); end CmdPWM ;
```

L'instanciation des différents blocs est faite comme suit :

```
U0: RapportC Port map( Rst =>Rst ,En => En, alphac =>alphac , alpha_out => alpha_out);
U1: ClkGen PORT MAP(  Clk_in => Clk, Rst => Rst, En => En, ClkSel => ClkSel, Clk_out
=> Clk_out);
U2: GenTriang PORT MAP(Rst =>Rst,En => En,Clk => Clk_out ,Triag_out => Triag_out);
U3: Compar PORT MAP(Rst => Rst,En => En,Clk => Clk_out,D_in_P => Triag_out,D_in_
N => alpha_out,D_out_CMP =>PWM_out1);
U4: ComparN PORT MAP(Rst => Rst,En => En,Clk => Clk_out,D_in_P => alpha_out,D_in
_N => Triag_out,D_out_CMP =>PWM_out2);
```

La simulation temporelle a donné :

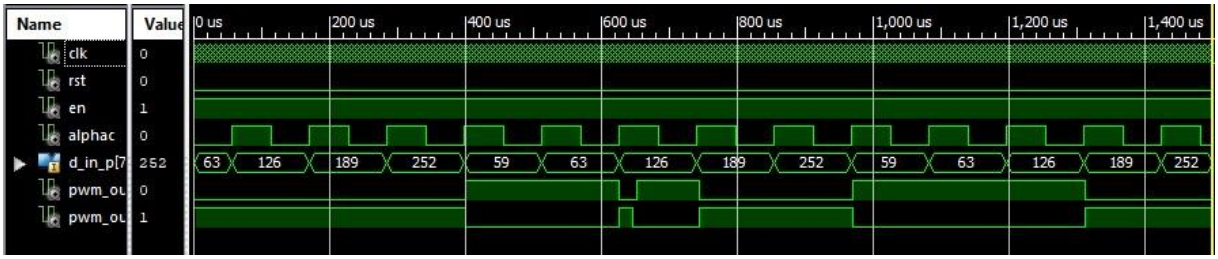


***Figure Ⅲ.6*:** Chronogrammes de simulation du générateur (modulateur PWM)

On voit que les deux sorties PWM_out1 et PWM_out2 sont complémentaires et qu'à chaque impulsion sur alphac une nouvelle valeur du rapport cyclique est chargée.

Afin de pouvoir choisir le sens de rotation, en ajoute une entrée LR qui permet de faire passer l'une ou l'autre des sortie PWM vers la sortie finale. Pour LR=0 c'est la sortie du premier comparateur qui passe, pour LR=1 c'est la deuxième. La simulation finale est la suivante :

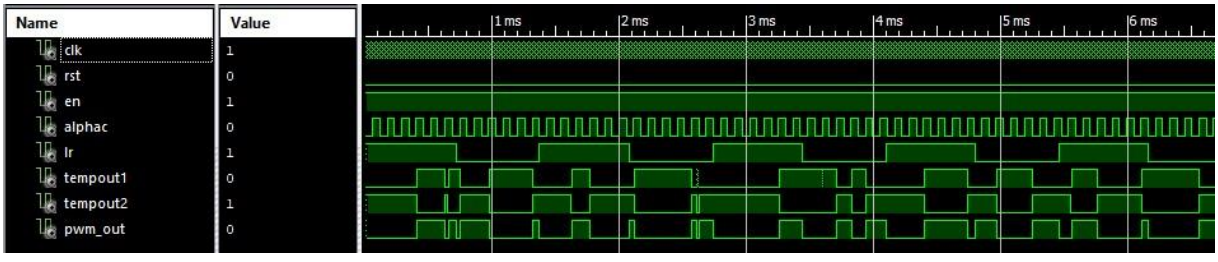


***Figure Ⅲ.7*:** Chronogrammes de simulation du générateur complet (modulateur PWM)

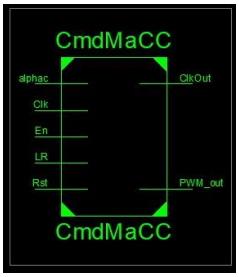
Ⅲ.4 Synthèse et Implémentation

Une fois satisfait des résultats de simulation, on passe à l'implémentation en utilisant les outils de Xilinx ISE.

Avant de lancer l'outil « implement » on doit d'abord spécifier quelle broche sur FPGA correspond à quel pin dans la conception VHDL. Pour ce faire, en ajoute un fichier dit de « contraintes » dans lequel on fait cette association.

```
Net "Clk" LOC="B8";
Net "Clk" IOSTANDARD = LVCMOS33;
## System level constraints
Net "Clk" TNM_NET = Clk;
TIMESPEC TS_Clk = PERIOD "Clk" 20000 ps;
Net "Rst" LOC="C13" | IOSTANDARD = LVCMOS33;# bouton poussoir 1
NET "alphac" LOC="D12" | IOSTANDARD = LVCMOS33;# bouton poussoir 2
NET "alphac" CLOCK_DEDICATED_ROUTE = FALSE;
Net "En" LOC="J11" | IOSTANDARD = LVCMOS33;# interrupteur 1
Net "LR" LOC="J12" | IOSTANDARD = LVCMOS33;# interrupteur 1 Net "PWM_out"
LOC="C11" | IOSTANDARD = LVCMOS33 ;# LED
```

La synthèse a généré les schématiques du composant entier ainsi que ses sous blocs :



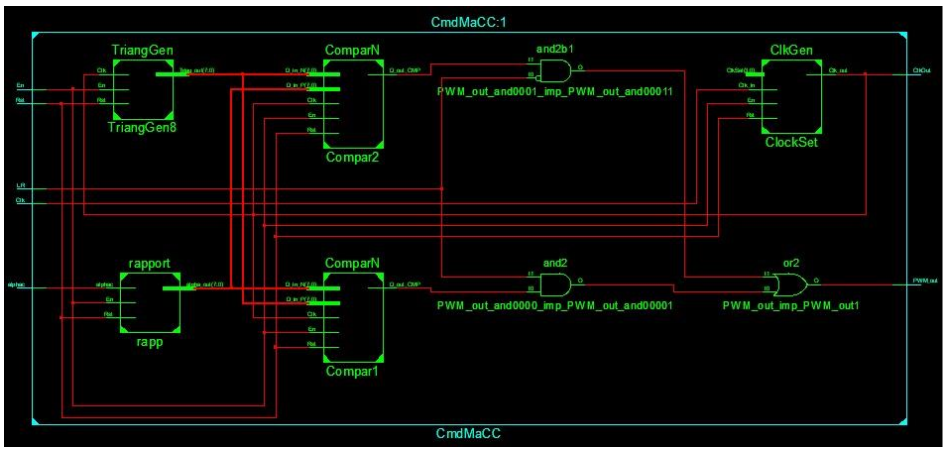


***Figure Ⅲ.8*:** Entité synthétisée du contrôleur PWM (CmdMaCC)

Le rapport final d'utilisation des ressources est le suivant :

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	26	9,312	1%	
Number of 4 input LUTs	31	9,312	1%	
Number of occupied Slices	27	4,656	1%	
Number of Slices containing only related logic	27	27	100%	
Number of Slices containing unrelated logic	0	27	0%	
Total Number of 4 input LUTs	44	9,312	1%	
Number used as logic	31			
Number used as a route-thru	13			
Number of bonded IOBs	7	190	3%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	3.29			

Table.1: Le rapport finale d'utilisation des ressources

Ⅲ.5 Configuration

La configuration se fait après avoir généré le fichier de programmation « .bit » . Il existe une option de génération de PROM/ACE cible sur l'onglet processus de Xilinx ISE qui convertit le fichier BIT en fichier PROM ou ACE. Ce fichier PROM ou ACE peut être téléchargé directement dans les cellules mémoire du FPGA. Nous devons nous assurer que le FPGA est connecté au PC sur lequel nous développons cette conception.

Après avoir téléchargé le fichier PROM ou ACE dans le FPGA, le FPGA est prêt à être utilisé comme générateur PWM. Nous pouvons donner différentes combinaisons d'entrées pour voir comment la sortie du FPGA varie. La sortie peut être vue par la LED dont le numéro de broche a été attribué à la sortie du générateur PWM.

Ⅲ.6 Conclusion

Dans cette partie, nous avons conçu et implémenté le modulateur PWM configurable sur la carte FPGA à l'aide de l'outil Xilinx ise, le langage utilisé dans la conception est VHDL. Les simulations ainsi que l'implémentation réelle sure FPGA ont été réussies.

Conclusion générale

L'objectif de ce mémoire était la conception et l'implémentation sur FPGA d'un modulateur PWM configurable, pour la commande d'un moteur à CC. Pour arriver à cette fin, nous avons commencé par aborder quelques topologies de générateurs PWM numériques les plus répandues. Nous avons choisi l'FPGA comme plateforme matérielle et avons présenté brièvement leur architecture.

La conception a suivi l'approche buttom-up, utile pour les petites conceptions comme la présente. Chaque bloc est alors décrit en langage VHDL et est simulé. On a ensuite rassemblé les différents blocs pour réaliser le modulateur complet final et l'on aussi vérifié par simulation. Toutes les simulations étant satisfaisantes, nous avons passé à l'implémentation sur la carte XilinX Spartant 3^E . Nous avons donc résumé le processus de l'implémentation présenté la plateforme matérielle NI Digital Electronics FPGA Board. L'application est fonctionnelle et répond bien à notre objectif initial.

Références:

[1] koutroulis E ,Dollas A. and Kalaitzakis k,"high –frequency pulse width modulation implementation using FPGA and CPLD ICs" ,journal of systems architecture ,vol.52(2006):pp.332-334.

[2] Dancy A.P , Amirtharajah R.and Chandrakasan A.P, "high-Efficiency Multiple-output DC-DC Conversion for Low –Voltage Systems ", IEEE Trans. on very large scale Integration (VLSI) Systemes , vol . 8, No .3 ,june 2000 : pp.252-263.

[3] Rahim M.A. and Islam Z, "Field programmable Gate Array-Based Pulse Width Modulation for signale Phase Active Power Filter "American Journal of applied sciences , vol.6 (2009) :pp.1742-1747.

[4] Retif J.M ,Allard B, Journal X and Peres A, "Use of ASIC's in PWM techniques for power converters ", Proceedings of the international Conference on Industrial Electronics ,control and control and instrumentation , IEEE Xplore press, Taipei,Taiwan,(1996):pp:138148.DOI:10.1049/ip-b:19850019

[5] Instrument, national.NI Digital Electronics FPGA Boord user manuel. Mai 2009.

[6] Xilinx.spartan-3E FPGA Family data sheet .14 décembre 2018 .

[7] Tzou, Y.Y., HJ. Hsu, T.S .Kou ,"FPGA based SVPWM control IC for 3-phase PWM inverters ", proceedings of the IEEE 22nd international Conference on Industrial Electronics, Control and Instrumentation, IEEE Xplore Preess, Taipei,Taiwan,(1996):pp:138148.DOI:10.11.09/IECON.1996.570921.