الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Abdelhafid Boussouf-Mila
Institut des Sciences et Technologie
Département de Génie Mécanique et Électromécanique

N^0 Ref :……

Projet de Fin d'Etude préparé En vue de l'obtention du diplôme de

MASTER

Spécialité : Électromécanique

Conception et mise en œuvre d'un générateur d'impulsions modulées en largeurs avec un FPGA-XILINX

Réalisé par :
- Islam Bey
- Ayoub Aifour

Soutenu devant le jury :

Me	**Dalila Yessad**	**Président**
M	**Bassem Keghouche**	**Examinateur**
M	**Billel Smaani**	**Promoteur**

Année universitaire : 2021/2022

Remerciement

Grâce à dieu miséricordieux tout puissant, qui nous a éclairé le chemin de réussite, et de nous avoir donné la santé, la patience, la puissance et la volonté pour réaliser ce travail.

Au terme de ce travail, Nous tenons également à exprimer nos sincères remerciements à tous ceux qui ont contribués de loin au de près à l'élaboration de ce projet de fin d'étude surtout ***Mr SMAANI Billel****, notre promoteur de nous avoir conseillé judicieusement, orienté, encouragé et de nous avoir apporté son attention tout au long de la réalisation de ce travail.*

Nos vifs remerciements vont également à l'ensemble des membres du jury, ***Mr KEGHOUCHE Bassem*** *et* ***Me YESSAD Dalila.***

D'avoir accepté de juger notre travail et pour l'intérêt qu'ils y ont manifesté.

Nos vifs remerciements à toute personne ayant contribué, de près ou de loin, dans ce travail

Dédicace

Avant tout et avec l'aide et la protection d'ALLAH S'est réalisé se modeste travail, merci Allah pour m'avoir donné la santé, la force et le courage pour mener a réalisé ce travail.

Je dédie ce modeste travail à :

A mes parents qui m'ont inculqué un esprit de combativité et de persévérance et qui m'ont toujours poussé et motivé dans mes études.

A ma mère ***Noura*** *et mon père* ***Abd-elhak*** *qui m'a soutenu durant tous les années d'études et pour leurs aides, encouragement, sacrifices et leur patience, merci pour tout ce que vous fait pour moi, que j'espère les rendre fière par ce travail. Puisse Allah vous garder Toujours la source de bonheur de ma vie.*

A mon très chère frère ***WASSIM*** *et mes chères sœurs* ***Imane et Ranime*** *et ma petite princesse* ***NADA*** *je vous souhaite une vie pleine de bonheur et de réussite.*

A mon binôme et ami ***Ayoub*** *merci pour l'excellent travail. Qu'avec lesquelles nous passeront des beaux moments et des bons souvenirs.*

A toute personne ayant contribué, de près ou de loin, dans ce travail, Enfin je dédie toute la promotion de ***MASTER 2 Electromécanique*** *je vous souhaite de bon continuation.*

Islam

Dédicace

Avant tout et avec l'aide et la protection d'ALLAH S'est réalisé se modeste travail, merci Allah pour m'avoir donné la santé, la force et le courage pour mener a réalisé ce travail.

Je dédie ce modeste travail à :

A mes parents qui m'ont inculqué un esprit de combativité et de persévérance et qui m'ont toujours poussé et motivé dans mes études.

*A ma mère **SOUAD** et mon père **OMAR** qui m'a soutenu durant tous les années d'études et pour leurs aides, encouragement, sacrifices et leur patience, merci pour tout ce que vous fait pour moi, que j'espère les rendre fière par ce travail. Puisse Allah vous garder Toujours la source de bonheur de ma vie.*

*A mon très chère frère **CHOUAIB** et ma chère sœur **SARA** je vous souhaite une vie pleine de bonheur et de réussite.*

*A mon binôme et ami **ISLAM** merci pour l'excellent travail. Qu'avec lesquelles nous passeront des beaux moments et des bons souvenirs.*

*A toute personne ayant contribué, de près ou de loin, dans ce travail, Enfin je dédie toute la promotion de **MASTER 2 Electromécanique** je vous souhaite de bon continuation.*

Ayoub

Chapitre I

La technique de modulation de la largeur d'impulsion (MLI)

Chapitre II

Les circuits FPGAs et le VHDL

Chapitre III

La carte de développement et les outils de conception

AC : Alternative Courant.

ASIC: Application Specific Integrated Circuit.

CC : Courant Continu.

CDMA : Code Division Multiple Accès.

CNA : Convertisseur Numérique-Analogique.

CMOS : Complementary Metal Oxide Semi-conducteur.

CPLD : Complexe Programmable Logique Devise.

CLB: Configurable Logic Block.

CLK: Clock.

DAC: Digital Analog Converter.

DC: Direct Courant.

DSP: Digital Signal Processor.

E/S: Entrées/Sorties.

FPGA: Field Programmable Gate Array.

IEEE: Institute of Electrical and Electronic Engineers.

ISE: Integrated Software Environment.

IOB: Input Output Block.

I/O: Inputs and Outputs.

LOC: Location.

LUT: Look Up Table.

MLI : Modulation de Largeur d'Impulsions.

PWM: Pulse Width Modulation.

PC: Personal Computer.

RAM: Random Access Memory.

RTL: Register Transfer Level.

USB: Universal Serial Bus.

VHDL: VHSIC Hardware Description Language.

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

La modulation de largeur d'impulsion (MLI) est une technique standard largement utilisée dans le contrôle des convertisseurs de puissance DC/DC et DC/AC. Dans ce contexte, la plupart des convertisseurs de niveau de puissance élevé fonctionnent à des fréquences de commutations supérieures (à 1 MHz) et des niveaux de puissance élevés peuvent être obtenus en utilisant la technologie de transformation planaires [1]. Aussi, la MLI standard est une excellente technique qui offre une méthode simple pour le contrôle de tels systèmes. Cependant, l'utilisation de la modulation de largeur d'impulsion (MLI) dans le système de contrôle d'électronique de puissance n'est pas nouvelle, il existe différentes approches pour développer des signaux modulées en largueurs. Dans ce cas, de nombreux circuits numériques peuvent être utilisés, mais ce qui est intéressant, c'est de générer des signaux MLIs en utilisant des langages de description matériels (HDLs) et des circuits reconfigurables tels que les FPGAs.

Un FPGA, soit en Anglais *« Field Programmable GateArray »* est un circuit logique reprogrammable selon les exigences d'application et les fonctionnalités souhaitées. Il peut être utilisé pour implémenter des équations et des systèmes logiques simples ou complexes, ainsi que pour la conception des systèmes de contrôle de différentes machines. Le VHDL est un langage HDL utilisé pour décrire le comportement des systèmes, il s'agit donc d'un langage de description de matériel de type VHSIC (circuit intégré à très grande vitesse), il est largement utilisé par l'industrie pour le prototypage de circuits ASICS ainsi que dans le milieu universitaire afin simuler et de synthétiser des architectures nouvelles [1].

L'objectif de ce projet est de réaliser un générateur d'impulsions modulées en largeur (MLI) en utilisant la carte de développement Basys2. Ceci en implémentant la MLI standard sur le FPGA Spartan-3E à l'aide du langage VHDL et l'outil ISE Design de Xilinx.

Dans ce contexte, ce mémoire est organisé en quatre chapitres :

Le premier chapitre se focalisera sur l'introduction des techniques de modulation de largeur d'impulsion, ainsi que leurs avantages et leurs applications.

Le deuxièmes chapitre décrira le l'architecture des circuits FPGAs ainsi que le langage de description matériel VHDL.

Le troisième chapitre décrira. La carte de développement et les outils de conception.

Le quatrième chapitre Conception, implémentation et réalisation. Il se focalisera également sur les résultats obtenus la partie des simulation et aussi l'expérimentale.

Chapitre I

La technique de modulation de la largeur d'impulsion (MLI)

I.1. Introduction

La modulation de largeur d'impulsion (MLI) est une technique de grande importance dans les cas des convertisseurs DC/DC et DC/AC, permettant ainsi la création de signaux de contrôler de l'analogique avec des sorties numériques via des composants programmables. Un générateur MLI consiste donc en une succession rapide d'impulsion électrique « tout ou rien » [2].

Dans ce chapitre, nous allons introduire la technique de modulation de largeur d'impulsion. Nous présenterons ainsi les différents types de générateurs MLI et les diverses approches permettant la génération d'un signal modulé en larguer.

I.2. Définition

La modulation de largeur d'impulsion (PWM) est une technique permettant de fournir un "1" logique et un "0" logique pour un période de temps contrôlée. C'est une source de signal implique la modulation de son rapport cyclique pour contrôler la quantité de puissance envoyée à une charge. La MLI génère des impulsions sur sa sortie en de telle sorte que la valeur moyenne des hauts et des bas soit proportionnelle à l'entrée MLI. En filtrant les impulsions, on obtient une valeur analogique proportionnelle à l'entrée MLI. Une entrée MLI est donc, peut-être de n'importe quelle largeur. Les valeurs les plus courantes sont 8 bits et 16 bits. Le PWM développé peut être utilisé dans de nombreuses applications diverses et complexes telles que la robotique, le contrôle des moteurs et du mouvement [2].

I.3. Types de générateurs MLI

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 MLI sont discutées, qui sont : (a) générateur basé sur un compteur de haute fréquence (b) générateur basé sur un compteur (c) générateur à compteur en cascade.

I.3.1 Générateur à base de compteur haute fréquence

Cette architecture a été proposée par E. Koutoulik et al [3], 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.

Puis, 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 du débordement du compteur est utilisé pour réinitialiser le verrou RS.

Ainsi, la sortie de la bascule RS donne la sortie PWM souhaitée. Ce signal de débordement est également utilisé pour charger un nouveau rapport cyclique N bits dans le registre [3].

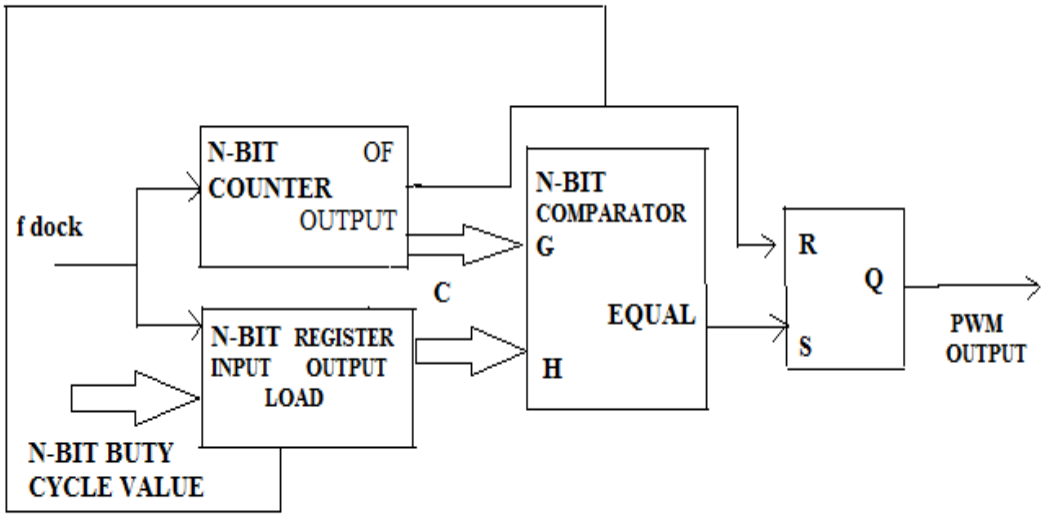


Figure I.1 : Architecture du générateur MLI développé par E. Koutroulis et al.

L'avantage de cette méthode est qu'elle est très utilisée pour générer une sortie MLI e haute fréquence, ce qui n'est pas faisable dans les approches normales qui se basent sur des compteurs.

I.3.2 Générateur à base de compteur classique

Cette architecture est très utilisée dans le cas des alimentations à découpage de faible puissance, elle a été proposée par A. P. Dancy et al [3]. Cette architecture est représentée par la figure I.2. Elle se base 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 MLI 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 [3].

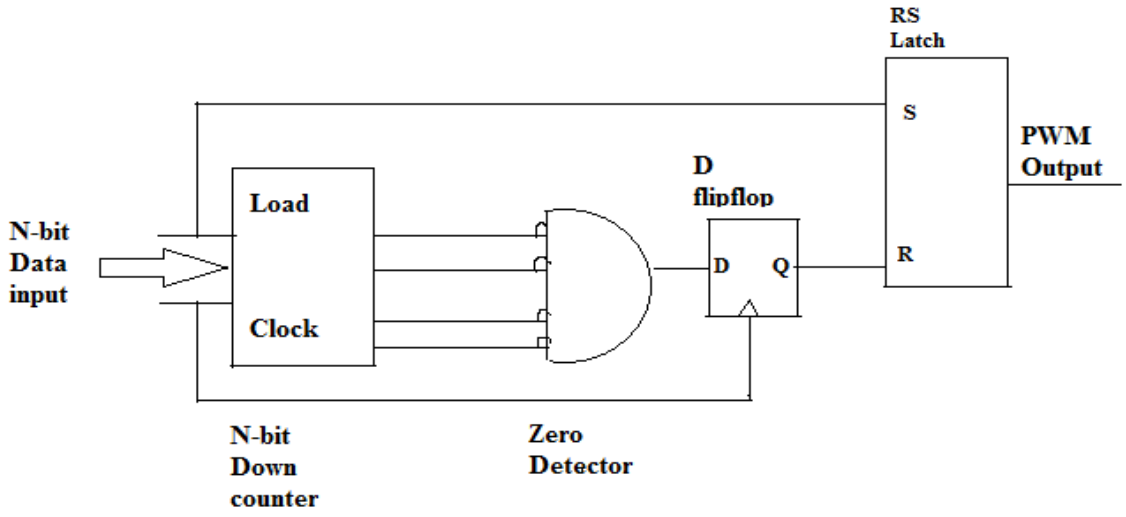


Figure I.2 : Architecture du générateur MLI à base de compteur proposé par A. P Dancy et al.

I.3.3 Architecture à base de compteurs en cascade

Dans le cas de cette architecture, deux compteurs de 4-bits sont mis en cascade afin de former 8-bits. Ensuite, 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. Le 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. En fin, la sortie du verrou RS est utilisée pour donner une sortie PWM [3].

I.4. Caractéristique de la commande MLI

Une commande MLI est souvent caractérisée par plusieurs paramètres qui sont **:**

- L'indice de modulation m qui est défini comme étant le rapport entre la fréquence de la porteuse (f') et la fréquence du signal de référence (f) : $m = f' / f$.
- La valeur de *m* peut être un entier comme elle peut être aussi un non entier.
- Dans le premier cas, la fréquence de la porteuse est un multiple de la fréquence de l'onde de référence f et cette modulation est dite synchrone. Par contre dans le 2ème cas, la modulation est dite asynchrone.

- Une valeur impaire de *m* permet d'obtenir pour la tension de sortie une alternance négative identique, au signe prés, à son alternance positive, ce qui élimine les harmoniques pairs dans la tension de sortie.
- Le coefficient de réglage de la tension *r* qui est défini comme étant le rapport entre l'amplitude de référence et la valeur crête de l'onde de modulation *r=A ref /Ap*.
- Le calage qui est dit optimal lorsque la position relative de la référence et de la modulante rend chaque alternance de la tension de sortie symétrique par rapport à son milieu [4].

I.4.1 Fréquence

La fréquence détermine la vitesse à laquelle le MLI effectue un cycle et par conséquent à quelle vitesse il passe de l'état haut et à l'état bas et le contraire. Elle se calcule à partir de la période du signal MLI.

La commande par PWM est très liée à la notion de fréquence. Pour que l'impression d'une valeur moyenne constante d'allumage apparaisse, il faut que l'alternance d'allumage/extinction soit suffisamment rapide pour qu'elle ne se remarque pas. Si par exemple le cycle complet de PWM durait une seconde (cette durée est nommée période), ce qui donne une fréquence de 1 Hertz (1 cycle par seconde), les durées d'allumage et d'extinction de l'actionneur seraient réparties proportionnellement sur cette seconde. Imaginons une lampe halogène allumée à 40% de sa puissance. Elle doit être alimentée 40% du temps et éteinte durant 60 % du temps. Avec une fréquence de 1 Hz, elle serait allumée pendant 0,4 seconde puis éteinte pendant 0,6 seconde. L'effet de clignotement résultant est parfaitement perceptible... La fréquence du PWM doit donc être beaucoup plus grande que 1 Hz, ou autrement dit la période doit être bien plus courte qu'une seconde. Selon les utilisations la fréquence de PWM va de 100 Hz (100 cycles par seconde) à 200 kHz. Nos cartes de commande permettent actuellement d'avoir deux fréquences de PWM : 100 Hz et 400 Hz [5].

I.4.2 Rapport cyclique

Le rapport cyclique MLI décrit la durée pendant laquelle le signal est à l'état haut (actif) en pourcentage de la durée d'un cycle complet. Il est définit par les registres de PDCx (PDCxL et PDCxH). Il y a au total 4 registres de rapport cyclique MLI pour 4 paires de canaux MLI.

On appelle rapport cyclique le rapport : *a= 100 x Th/ T*.

- Si th =0 alors a=0% et la tension moyenne de sortie est nulle.
- Si th=T alors a=100% et la tension moyenne de sortie est égale à Vcc [5].

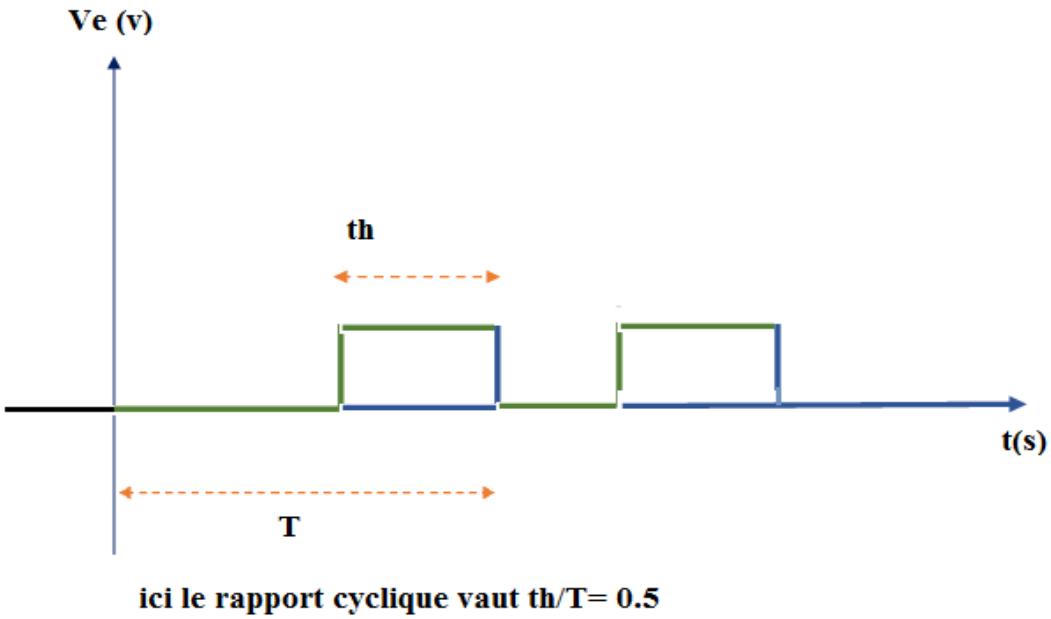


Figure I.3 : Signal décrivant le rapport cyclique.

Le principe utilisé pour réaliser un rapport cyclique variable est de comparer un signal triangulaire avec un autre signal de tension continu variable.

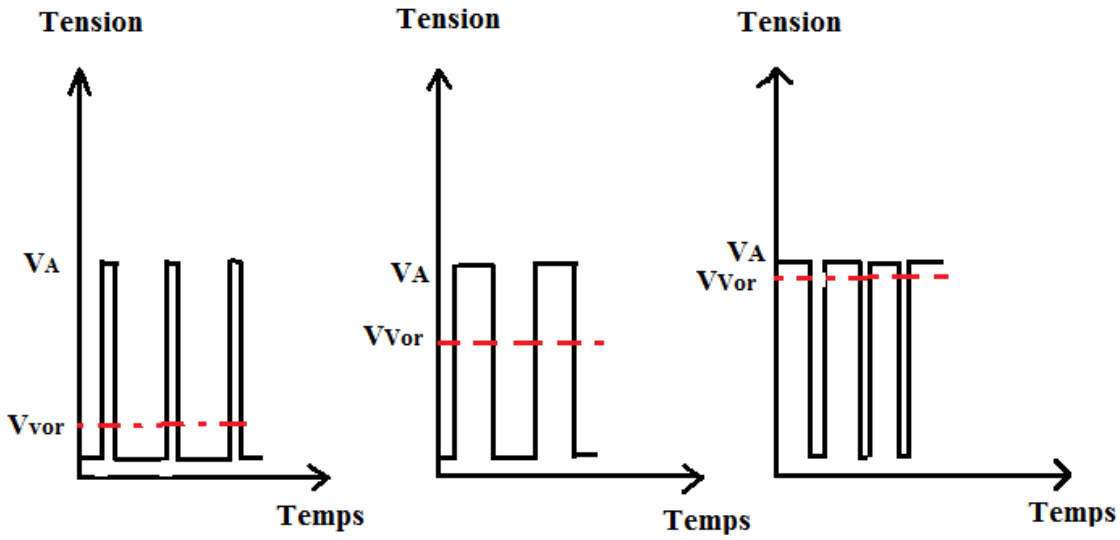


Figure I.4 : Signaux MLIs avec différent rapport cyclique : (a) rapport cyclique de 25%, rapport cyclique de 50%, (b) rapport cyclique de 75%

I.5. Différent approches

I.5.1. MLI "intersective"

C'est l'approche la plus classique. Elle consiste à comparer la modulante (le signal à synthétiser) à une porteuse généralement de forme triangulaire. Le signal de sortie vaut « 1 » si la modulante est plus grande que la porteuse, sinon il vaut « 0 » ; le signal de sortie change d'état pour chaque intersection de la modulante et sa porteuse [5].

Cette méthode est très rentable dans des réalisations analogiques. Dans ce cas on utilise un générateur triangle et d'un comparateur. Il existe de nombreux circuits intégrés dédiés.

On peut classifier les sous-types de plusieurs manières :

- Analogique ou numérique échantillonné, selon que la modulante et le comparateur sont en temps continu ou discret ;
- À porteuse triangulaire centrée ou en dent de scie (à gauche ou à droite) ;
- Asynchrone ou synchrone, selon que la modulante et la porteuse sont de fréquence exactement multiples ou non [6].

I.5.2. MLI " vecteur spatial "

La MLI dite « space vector » (vecteur spatial) est approprié aux applications des variateurs de vitesse triphasés sans neutre.

Dans ce cas, on considère généralement un système triphasé, et à lui appliquer une transformée de Concordia pour se ramener dans le plan (Vα, Vβ). Le système triphasé de tensions à générer pour la durée d'échantillonnage en cours peut alors être représenté comme un unique vecteur dans ce plan (voir aussi commande vectorielle).

Ce vecteur n'est pas directement réalisable par les interrupteurs du variateur, mais on peut chercher les trois configurations les plus proches (situées sur les sommets et au centre de l'hexagone), et les appliquer successivement pendant une fraction adéquate de la période d'échantillonnage, de façon à obtenir en moyenne le vecteur recherché.

En modulation sinusoïdale, elle donne des résultats similaires à la MLI intersective à porteuse triangulaire centrée. Néanmoins, elle peut être plus facile à implanter dans un microcontrôleur, ce qui justifie son usage [6].

I.5.3. MLI " pré calculée "

Elle est largement utilisé dans le cas où la fréquence de la porteuse est faible, on à donc besoin d'optimiser le spectre du signal généré. En effet, le motif du signal de sortie est prédéterminé (hors ligne) et stocké dans des tables qui sont ensuite relues en temps réel.

Puisque ce type est synchrones (la fréquence porteuse est exactement multiple de la fréquence de la modulante), la condition nécessaire pour avoir un spectre harmonique constant. Aussi, ce type de MLI ne peut être réalisé qu'en numérique [6].

I.5.4. Commande par hystérésis

Cette méthode consiste à créer un signal MLI directement à partir de la grandeur à contrôler, cela par des décisions de type « tout ou rien ».

Cette méthode est très simple et le temps de réponse minimal aux perturbations. L'inconvénient majeur est l'absence de contrôle de la fréquence de commutation des transistors, ce qui rend délicat leur dimensionnement [6].

I.6. Principe de base

Le principe de base de la modulation de largeur d'impulsion repose sur le découpage d'une onde rectangulaire. Ainsi, la tension de sortie de l'onduleur est formée par une succession de créneaux d'amplitude égale à la tension d'alimentation (continue) et de largeur variable. La technique la plus répandue pour la reproduction d'un signal MLI est de comparer un signal triangulaire appelé porteuse de haute fréquence à un signal de référence appelé modulatrice et qui constitue l'énergie du signal recueil à la sortie de l'onduleur [7].

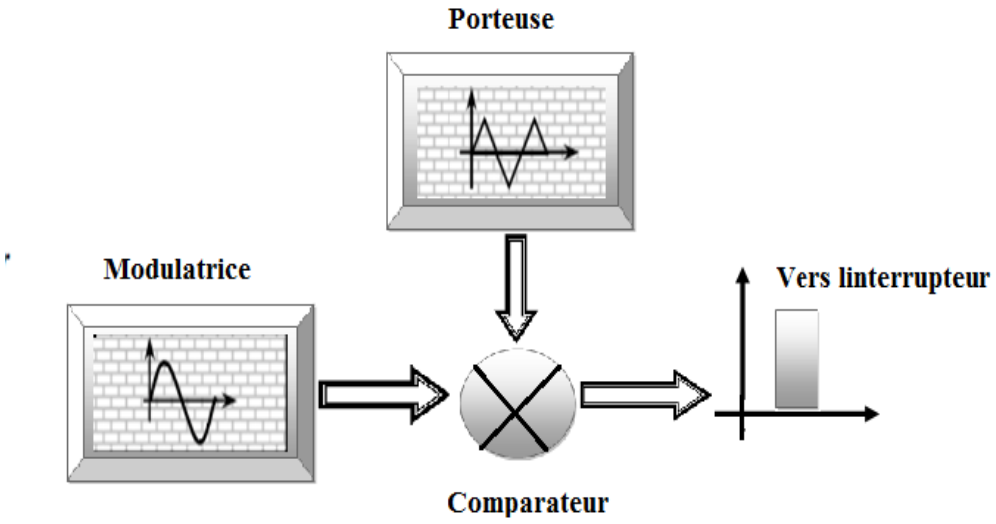


Figure I.5 : Schéma synoptique décrivant le principe de la MLI.

Il est donc formé à partir de la génération d'une tension carrée à un rapport cyclique variable et d'une fréquence constante, comme présente la figure I.6. La tension est périodique, de période **T** est égale à **E** pendant une durée **Ton**, à **0** pendant : [**Ton, T**]**.**

Un générateur MLI fonctionne avec un compteur 32 bits (8 ou 16 bits) et une horloge. À chaque Top d'horloge, le compteur est incrémenté d'une unité. Lorsqu'il atteint une valeur maximale (MAX), il revient à 0, puis le cycle recommence.

Le rapport cyclique est fixé par un registre dont la valeur R est inférieure à la valeur MAX du signal. À chaque incrémentation, la valeur de compteur est comparée à R. Si le compteur est supérieur à R, la sortie bascule au niveau bas. Lorsque le compteur est remis à zéro, la sortie bascule au niveau haut. Donc on obtient un signal carré [7].

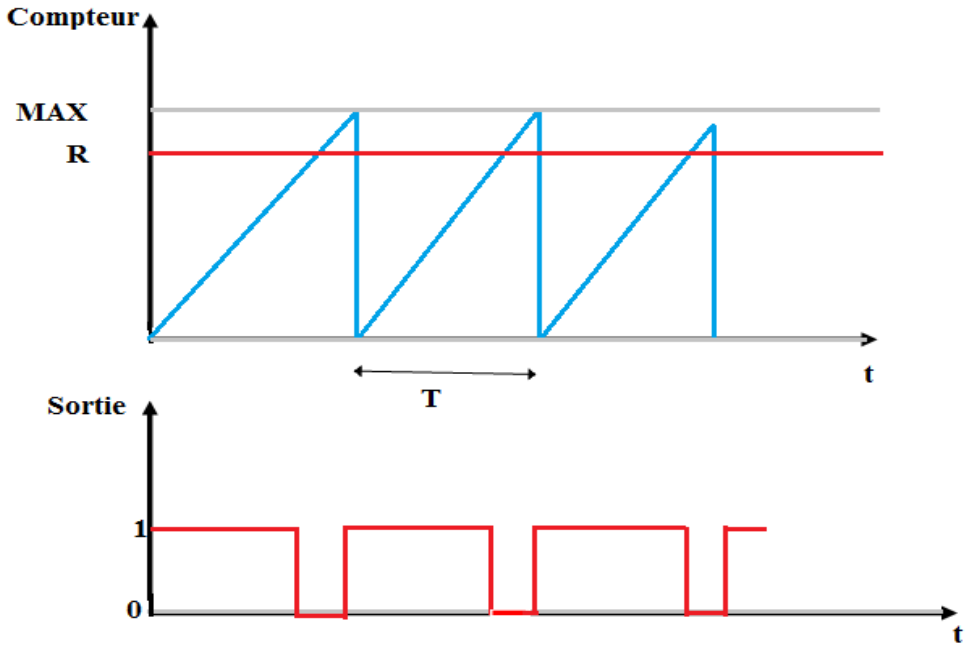


Figure I.6 : Principe de génération de la MLI.

I.7. Intérêt de la MLI

Elle permet d'obtenir un équivalent d'une variation de tension continue à l'aide d'un contrôle en « tout ou rien ». Le MLI (signal tout ou rien) permet également aux composants de puissance de beaucoup moins chauffer qu'en analogique (signal continue). D'autre part, les signaux numériques sont moins sensibles au parasitage que les signaux analogiques ils sont donc plus fiable.

Le principal intérêt de la technique PWM est de limiter la chauffe des composants électroniques. En effet, en commande analogique, pour obtenir une variation de puissance il faut dissiper le complément de la puissance maximale consommée.

Par exemple : une lampe de 20 Watts allumée au maximum consomme 20 Watt. Si par une commande de gradation elle est allumée au quart de sa puissance, elle consomme 10 Watt. Le composant analogique devrait alors dissiper 10 W, ce qui implique un énorme radiateur si l'on utilise un système analogique. En PWM, la puissance fournie est soit maximale, soit nulle. Lorsqu'elle est maximale, pendant la moitié du temps dans cet exemple, il n'y a pas besoin de dissiper de puissance résiduelle. Lorsqu'elle est nulle, il n'y a pas besoin de dissiper non plus de puissance, car aucune puissance n'est fournie [8].

I.8. Applications

La MLI est utilisé dans de nombreuses applications simples et complexes, les usages les plus fréquents sont [8] :

- La conversion numérique-analogique
- Les amplificateurs de classe D, en audio
- Les alimentations à découpage, variateurs de vitesse, et plus généralement tous les dispositifs d'électronique de puissance utilisant des composants de type MOSFET, IGBT, GTO.
- Il est aussi possible de faire de la transmission de données par cette méthode [8].

I.8.1 Variation de la vitesse

La variation de la 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 ton est la durée de l'état haut du signal [9].

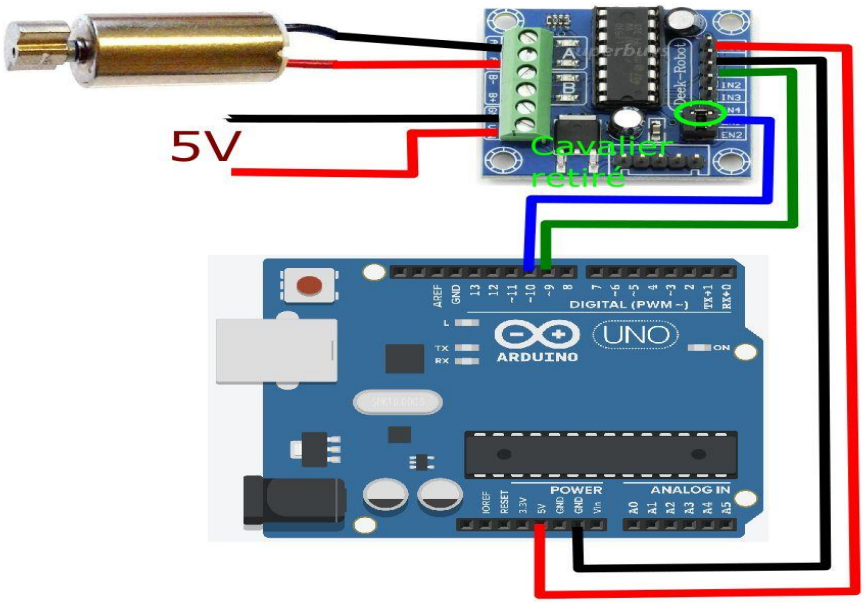


Figure I.7 : Variateur de vitesse d'un moteur à courant continu [9]

I.8.2 Convertisseurs

Ce sont des dispositifs à composants électroniques capables de modifier la tension et/ou la fréquence de l'onde électrique.

On différencie quatre types de convertisseurs (AC/DC, DC/AC, DC/DC, AC/AC) :

- Conversion AC-DC (les redresseurs) : Les redresseurs convertissent une tension alternative en une tension continue variable.
- Conversion DC-AC (les onduleurs) : Les onduleurs convertissent une tension continue fixe en une tension alternative de fréquence variable.
- Conversion DC -DC (les hacheurs) : Les hacheurs convertissent une tension continue fixe en tension continue variable.
- Conversion AC-AC (les gradateurs) : Les gradateurs permettent d'obtenir une tension alternative variable à partir d'une tension alternative fixe [10].

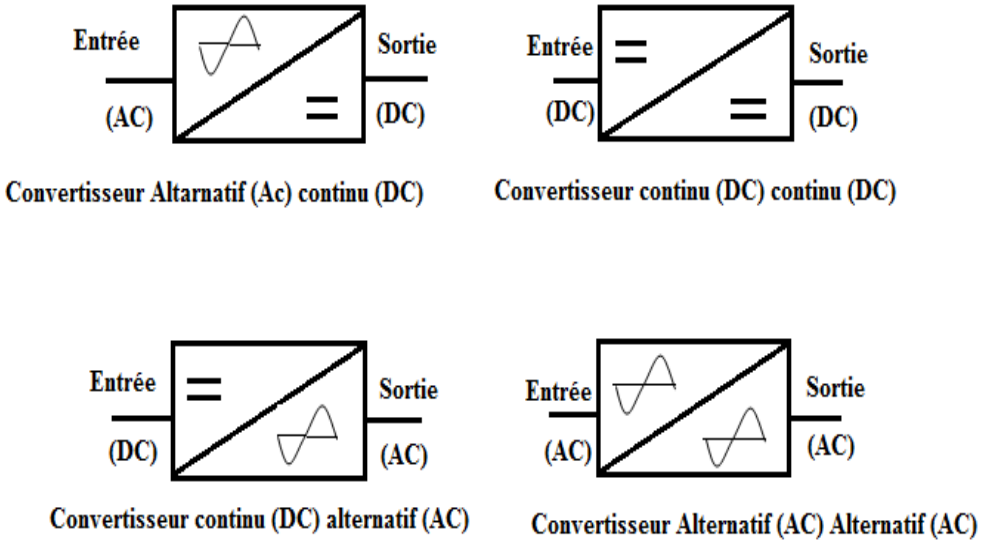


Figure I.8 : Types de convertisseurs.

I.8.3 Synthèse vocale

La synthèse vocale est une technique informatique qui permet de créer de la parole artificielle à partir de n'importe quel texte. Il associée à un filtre passe bas permet la synthèse des signaux audio [10].

I.9. Avantages et inconvénients

Nous citons les avantages suivants **:**

- Variation de la tension "en direct" très simple à réaliser.
- Perte de puissance dans les dispositifs de commutation est très faible.
- Fonctionnalité à faible consommation, sans bruit et à faible coût.
- Haute efficacité.
- Flexibilité dans le contrôle.
- Poids léger.
- Réponse rapide.

Le signal alternatif est de forme carrée et non pas sinusoïdale, ce qui génère une quantité importante d'harmoniques parasites. [11]

I.9. Conclusion

Dans ce chapitre, nous avons décrit la technique de modulation du largueur d'impulsion (MLI). Dans ce cadre, nous avons présenté le principe de génération d'une MLI, ses caractéristiques, ses différents types et approches, ainsi que ses applications.

En effet, une MLI est une technique de bonne fiabilité et d'excellents résultats analogiques avec des moyens numériques. Cependant, les avantages et les performances de cette technique nous ont conduits à considérer cette technique (MLI) dans le cadre de ce projet.

Chapitre II

Les circuits FPGAs et le VHDL

II.1. Introduction

Les FPGAs, soit en Anglais ‘’Field Programmable Gate Array’’ sont des circuits numériques reconfigurables (ou programmables). Ils permettent le prototypage de circuits ASICs ainsi que la conception et la validation des circuits et des systèmes numériques simples ou complexes. La configuration des circuits FPGA est réalisée à travers l’utilisation de langages de description matériels comme le cas du langage VHDL (*Very high speed integrated circuits Hardware Description Language*) [11].

Dans ce chapitre, nous allons introduire l’architecture des circuits FPGAs ainsi que le langage VHDL utilisés dans le cadre de ce projet.

II.2. Circuits FPGA

II.2.1 Historique

La première technologie de composants programmables (PLD) a été développée dans les années 70s par la société MMI. Elle développe ainsi un circuit intégré qui rassemble des portes logiques "pré-câblées". Lorsque l'utilisateur reçoit son circuit intégré vierge, chaque entrée d'une porte est reliée à toutes les entrées du circuit intégré, via une grille de connexion. Par la suite, le programmeur va donc éliminer les connexions inutiles en les faisant littéralement fondre. Ne resteront que les connexions qui réalisent la fonction logique désirée. Cette technologie est appelée "grille de portes logiques programmables" (programmable array logic), souvent abrégé par "PAL". Ensuite, cette technologie a rapidement évolué vers les composants CPLD (complex programmable logic devices) et de nos jours, ce sont les FPGA qui sont de plus en plus populaires grâce à leurs compromis souplesse / prix / efficacité. Les premières versions de circuits FPGAs ont été proposées au début des années 80s. Le développement des FPGA remonte à Ross Freeman en 1985, il a donc créé un FPGA à base de mémoires SRAM, ainsi que des FPGA à anti-fusibles (1990 Actel) programmables électriquement par l’utilisateur et non effaçables Aujourd’hui, ils sont parmi les circuits intégrés les plus complexes en nombre de transistors. Outre, deux manufacturiers américains de FPGA se partagent la plupart du marché : Altera et Xilinx [11]**.**

II.2.2 Architecture

Un FPGA est un circuit à densité d'intégration très élevée dont l'architecture est formée d'un ensemble de blocs logiques programmables que l'utilisateur peut les interconnecter afin de réaliser les fonctions désirées.

Les circuits FPGAs se composent de plusieurs types de ressources matérielles. Les principaux éléments formant un FPGA sont (figure II.1) [12].

- Les blocs logiques programmables.
- Les blocs d'entrées/sorties.
- Le réseau d'interconnexion.

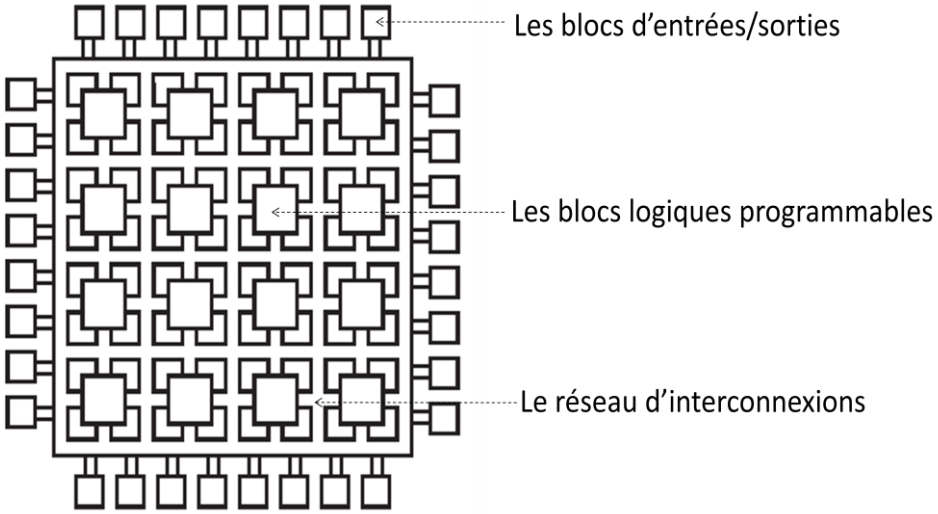


Figure. II.1 Architecture générale d'un FPGA.

Les éléments formant le circuit FPGA sont disposés par groupes et par collons, comme présente la figure ci-dessus.

On note que l'architecture des circuits FPGA se diverse selon le fabricant (exemples : Xilinx ou Altera), mais pas dans l'architecture générale. Cela d'une part. Et d'autre part, selon la technologie des composants utilisés [12].

II.2.2.1 Blocs logiques programmables

Ses blocs sont conçus pour la configuration et la programmation des fonctions logiques combinatoires et séquentielles.

Les blocs logiques configurables (CLB) sont souvent formés de : tables de conversions (LUT), de multiplexeurs et de bascules D. La figure ci-dessous présente la structure générale d'un CLB de la famille Xilinx [12].

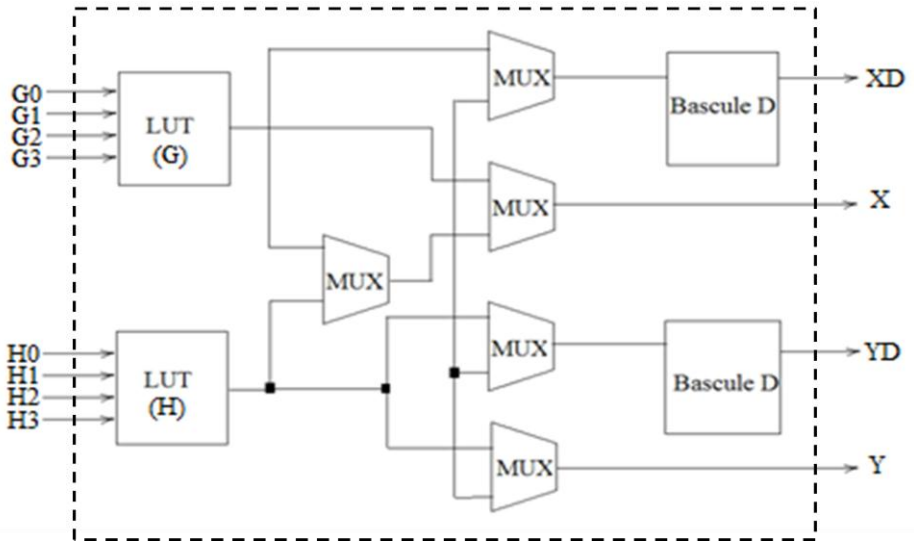


Figure. II.2 : Structure simplifiée d'un bloc logique programmable (CLB)

Dans cette structure (figure II.2), on trouve deux tables de conversions de 16 bits. Ses tables permettent d'implémenter les fonctions combinatoires désirés. Les multiplexeurs sont utilisés pour l'acheminement et la synchronisation des données. Afin d'avoir des sorties avec mémoration en utilise des bascules D.

II.2.2.2 Blocs d'entrées/sorties

Ses blocs permettent l'interfaçage des données entre les broches du circuit et la structure interne du composant. Chaque bloc permet de configurer la pine du circuit en entrée, en sortie ou à l'état bidirectionnel [12].

La figure II.3 montre le schéma général d'un bloc d'entrée/sortie (IOB).

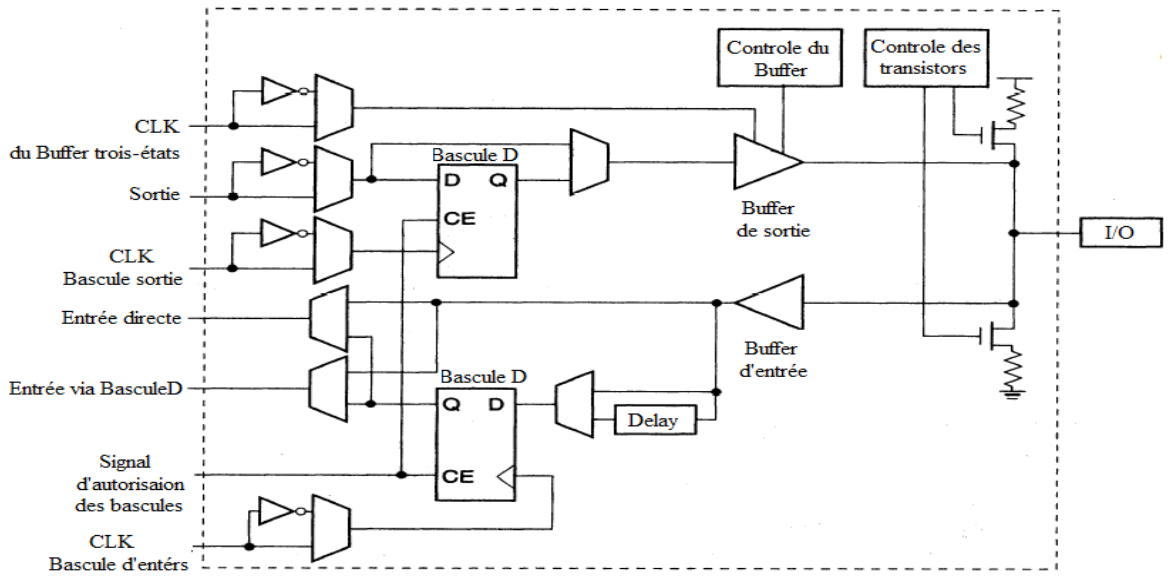


Figure. II.3 : Structure d'un bloc d'entrée/sorties (IOB).

La figure II.3 représente la structure d'un IOB dans le cas des FPGAs de la série XC4000E. Ce bloc IOB est basée sur l'utilisation de deux bascules permettent de prendre en charge le signal entrant ou sortant. Ces deux derniers signaux se configurent à l'état combinatoire ou séquentiel. On trouve aussi deux Buffers pour les deux signaux (d'entrée ou sortie) où le buffer de sortie à trois-états. Les deux transistors sont utilisés pour l'adaptation des signaux en TTL ou en CMOS. Un temporisateur est utilisé afin de créer un délai en entrée permettant ainsi de compenser les délais de distribution du signal d'horloge interne.

II.2.2.3 Réseau d'interconnexion

Les interconnexions (ou routage) sont réalisés à travers des matrices souvent appelées ‘’matrices d'interconnexion’’ (MC). Ces matrices sont disposées entre les blocs logiques configurables (CLB), comme présente la figure II.4 [12].

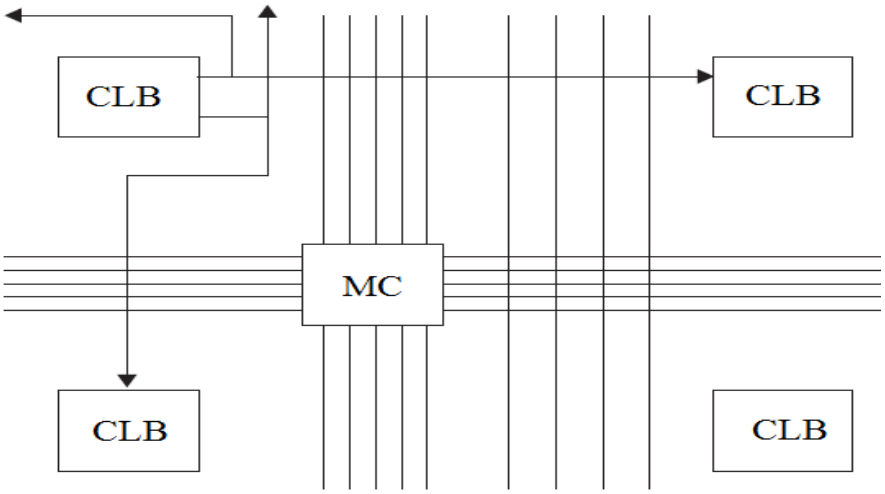


Figure. II.4 : Disposition des matrices d'interconnexion dans un FPGA.

Dans le cas de ce réseau, on distingue trois types de configurations (ou d'interconnexion) : la technologie des anti-fusibles, la technologie EPROM et la technologie SRAM.

II.2.2.4 Blocs de mémoires et multiplicateurs

L'évolution des performances technologiques des circuits FPGA nécessite l'intégration de différents modules comme les mémoires et les multiplicateurs. Ces blocs sont intégrés dans les circuits FPGA aux voisinages de CLBs et par colonnes (figure II.5) [12].

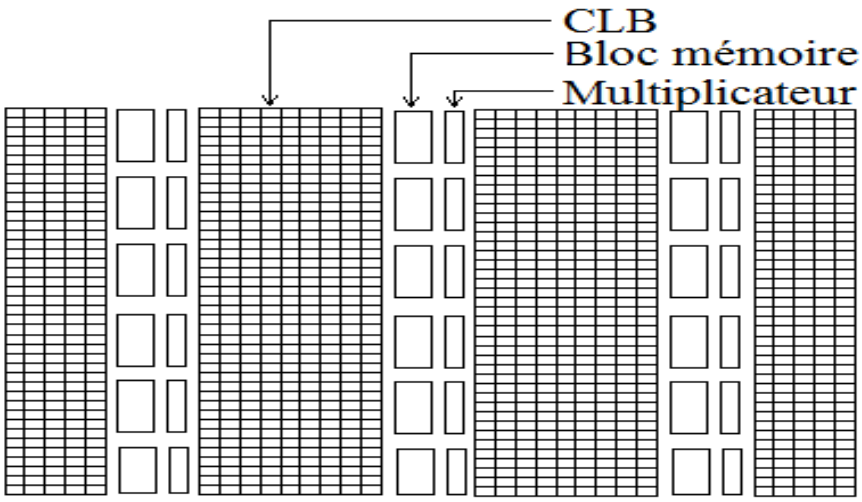


Figure. II.5 : Disposition des blocs mémoires et des multiplicateurs dans un FPGA.

II.2.3 Constructeurs

Le marché des circuits FPGAs est presque dominé par deux grands fabricants qui sont Xilinx et Altera. Le tableau II.1 donne un aperçu sur le classement des fabricants de circuits FPGA dans le monde [13].

Tableau. II.1 Classement des constructeurs de circuits FPGA.

Nom du fabricant	Classement	Pourcentage
Xilinx	1	38%
Altera	2	34%
Lattice	3	14%
Actel	4	6%
Autres	5	8%

La famille Xilinx est connue par deux types de circuits FPGA qui sont le Spartan et le Virtex. La famille Altera comprend trois types de circuits qui sont le Cyclone, le Stratix et Arria.

II.2.4 Domaines d'application

Les circuits FPGAs sont utilisés dans divers domaines d'applications [13] :

- Domaine militaire : les circuits FPGA de Xilinx ou Altera permettent aux concepteurs de répondre aux exigences de l'industrie aérospatiale et militaires. Cela afin de développer des systèmes de haute performance, d'une large plage de fonctionnement et d'une longue durée de vie du système.
- Domaine médical : les performances technologies des circuits FPGAs permette l'utilisation de ces types de composants dans les applications d'acquisition et du traitement d'images. Dans le domaine industriel, on trouve comme exemple le fabricant Altera qui fournit des circuits FPGAs avec des outils pour optimiser les performances technologiques des systèmes en vue de leurs utilisations dans des applications médicales.
- Domaine de la communication : beaucoup d'efforts de la part d'Altera et de Xilinx pour le développement des systèmes et des outils pratiques et flexibles utiles dans le domaine de la communication cellulaire mobile.

- Domaine de l'automobile : les circuits FPGAs pouvant être utilisés dans la technologie des automobiles évoluées tels que les véhicules hybrides et les véhicules électriques.
- Domaine de la recherche & développement (R&D) : actuellement, les circuits FPGAs sont très utilisés dans divers domaines de la R&D. Citant comme exemples : la commande des systèmes à temps réel, le traitement d'images (du son et de la vidéo), l'acquisition des données ainsi que le test/mesure. Ce type de composant joue un rôle incontournable dans la conception et le prototypage des circuits ASICs.

II.3. Langage VHDL

II.3.1 Définition

Le VHDL est un langage de description matériels, soit en Anglais ''Hardware Description *Language* (VHDL). Il s'agit d'un langage qui permet de décrire le comportement d'un circuit ou d'un système électronique, comme les systèmes intégrés sur puce (SOC). Le VHDL dérive de l'abréviation VHSIC qui signifie ''Very High Speed Integrated Circuits''. La création du langage VHDL remonte aux années quatre-vingt par le département de la défense des états unis. Vers la fin des années quatre-vingt, le VHDL a été publié sous forme d'une norme internationale par l'institut IEEE.

En effet, le langage VHDL joue un rôle indispensable dans la conception et l'implémentation des circuits sur des composants programmables tels que les CPLDs et les circuits FPGAs [13]**.**

II.3.2 Structure d'un programme VHDL

Une description VHDL est souvent formé de trois parties : la partie architecture, la partie entité et la déclaration des bibliothèques (figure II.6) [14].

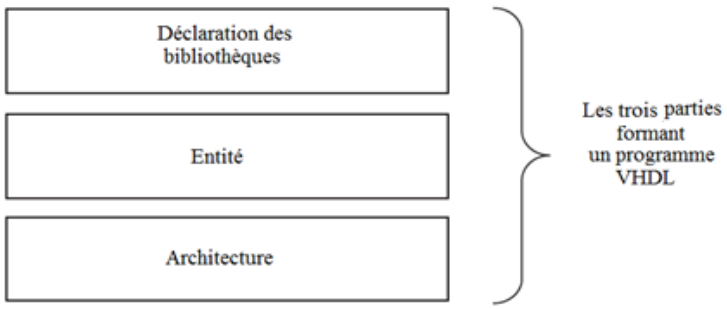


Figure. II.6 : Schéma des principales parties formant un programme VHDL.

II.3.2.1 Bibliothèques & packages

Un programme VHDL nécessite l'utilisation d'une ou de plusieurs bibliothèques utilisées pour la description et la synthèse des circuits numériques. Ces bibliothèques sont formés de packages qui permettent la définition des fonctions, des signaux, des composants ainsi que des sous programmes pour la réalisation des opérations logiques ou arithmétiques [14].

Puisque le VHDL est un standard de l'institut IEEE, la bibliothèque principale est nommée « IEEE ». Cette dernière est souvent déclarée dans la première ligne du code. En effet, la bibliothèque IEEE contient plusieurs packages, où l'utilisation de ces derniers est réalisée avec l'instruction « USE ».

Les principaux bibliothèques et packages sont représentés par le tableau II.2.

Tableau. II.2 Principales bibliothèques & packages du langage VHDL.

	Nom	**Syntaxe**	**Remarques**
Bibliothèque	ieee	Library ieee ;	Bibliothèque principale
Package	std_logic_1164	Use ieee.std_logic_1164.all;	Utiliser dans les applications et les systèmes logiques
	ieee.numeric_std	Use ieee.numeric_std.all;	Utiliser dans les opérations numériques comme les circuits-compteurs
	std_logic_arith	Use std_logic_arith.all	Utiliser dans les applications et les opérations arithmétiques comme la comparaison. Il contient plusieurs fonctions de conversion de données permettant de convertir un type en un autre

II.3.2.2 L'entité

Soit en Anglais ''ENTITY'', cette partie permet la déclaration des entrées/sorties des circuits ou des systèmes considérés. Elle permet également la définition du nom du code VHDL.

L'entité contient des entrées/sorties, ces derniers sont considérés comme étant des signaux électriques. Un signal est un objet manipulé par un code VHDL, il est souvent représenté par un fil de connexion. Il est caractérisé par le sens et le type. [14]

La figure II.7 décrit les principaux types de signaux.

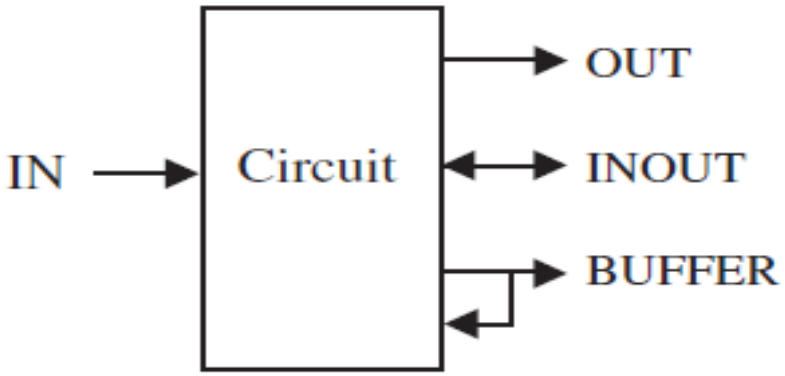


Figure. II.7 : Type de signaux électriques.

II.3.2.3 L'architecture

Elle permet de décrire le comportement du circuit ou du système considéré. Généralement, elle contient les fonctions des signaux de sorties en fonctions des signaux d'entrées.

Dans la partie architecture souvent, on trouve des Process, soit en Français Processus. Un Process peut être considéré comme une boite qui contient des instructions à l'intérieur et qu'exécutent de manière séquentielle [15].

II.3.3 Différents styles

Dans le langage VHDL, il existe trois principaux types de description [15] :

- La description par flot de données : le circuit ou le système considéré est décrit à travers des expressions booléennes (les opérateurs logiques) des fonctions de sorties en fonction des entrées.
- La description comportementale (appelée aussi procédurale) : le comportement du circuit considéré est décrit selon des conditions (IF), des cas (CASE, WHILE), et des boucles (FOR). On trouve aussi des PROCESS.
- La description structurelle : Afin de simplifier la description d'un circuit ou d'un système électronique compliqué, la description structurelle consiste à

partager le circuit en plusieurs blocs, souvent appelés « composantes ». Puis, chaque composante sera décrite par un code VHDL séparé. Un programme principale sera élaboré pour la description des composantes et des ‘’port map’’ formant ainsi le circuit considéré.

II.4. Conclusion

Dans ce chapitre, nous avons décrit les circuits programmable FPGA (Field programmable Gate Array) ainsi que leurs structures. Nous avons montré qu'un FPGA est formé de trois principaux blocs : blocs d'entrées/sorties, blocs logiques configurables et les réseaux d'interconnexion. Aussi, nous avons parlé des différents domaines d'application de ce type de circuit, tels que le domaine médical, le domaine militaire, la recherche et le développement.

En plus, nous avons introduit le langage VHDL, la structure d'une description VHDL ainsi que les différents styles d'une architecture VHDL. Ceci car le couple FPGA-VHDL sont largement utilisés dans le cadre de ce projet.

Chapitre III

La carte de développement et les outils de conception

III.1 Introduction

Le VHDL dérive de l'abréviation VHSIC qui signifie ‘’Very High Speed Integrated Circuits”, il s'agit d'un langage permettant la description du comportement d'un circuit ou d'un système quelconques, comme les systèmes intégrés sur puce, ce langage joue un rôle indispensable dans la conception et l'implémentation des circuits et des techniques de commande des machines sur des composants programmables comme les circuits FPGAs.

Dans ce chapitre, nous allons présenter le matériel et les outils Software utilisés dans le cadre de ce projet, citant la carte Basys 2 et le Spartan 3E et ses caractéristiques, l'outil ISE Design ainsi que l'outil Adept [15]**.**

III.2 L'implémentation sur FPGA

Il s'agit de faire implémenter des circuits numériques ou des systèmes sur un circuit FPGA come le cas du Spartan 3E de la firme Xilinx.

Les outils considérés se présentent comme suit [16] :

- La carte Basys2 avec le FPGA « Spartan 3E ».
- Le logiciel ISE Design Suite.
- Le logiciel Adept de Digilent.

L'implémentation d'un système quelconque sur un FPGA de type Xilinx passe par les étapes présentées sur le schéma de la figure III.1.

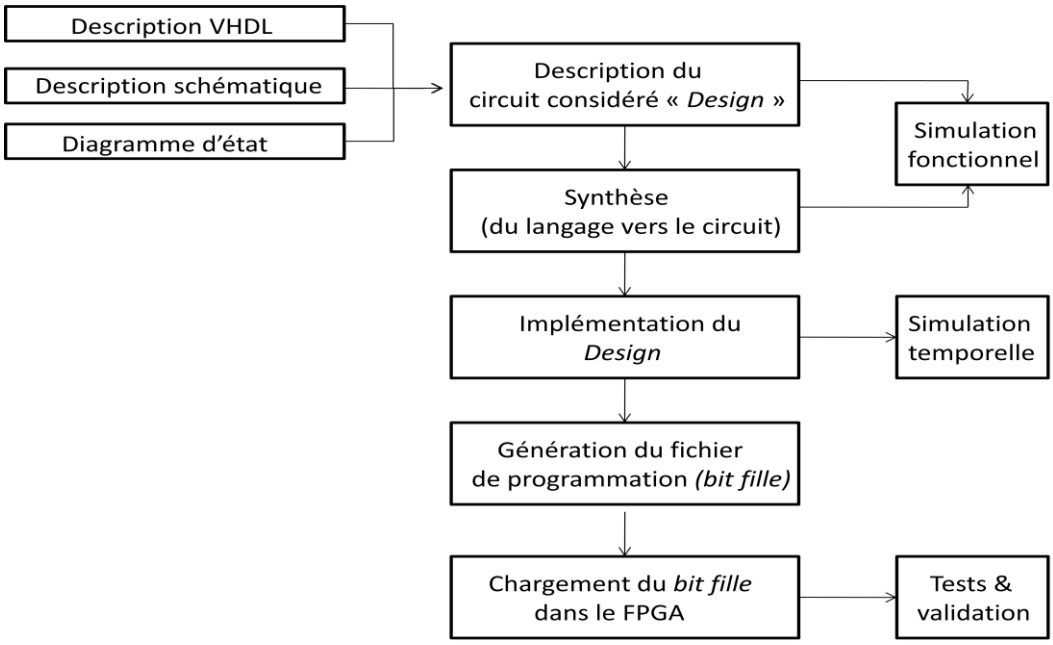


Figure III.1 Schéma des principales étapes de la conception des systèmes sur FPGA.

La première étape consiste à décrire le circuit ou le système numérique considéré. Cela à travers un code VHDL, ou via la description par schématique ou la description en utilisant le diagramme des états.

La deuxième étape est la synthèse, il s'agit de faire en déduire la structure du schéma considéré à travers le code VHDL élaboré. À cette étape une simulation fonctionnelle du circuit considéré peut être réalisée.

La troisième étape permet de générer le fichier de construction. Ce dernier est un fichier d'extension ''.UCF'', il permet de relier les pines du circuit FPGA avec les signaux 'd'entrées/sorties déclarés au niveau de l'entité du code VHDL. À cette étape une simulation temporelle peut être réalisée.

À travers la simulation temporelle réalisée dans l'étape précédente, un fichier de programmation est généré, ce dernier est un fichier d'extension ''.BIT'', c'est le fichier qu'il faut charger dans la carte FPGA.

La dernière étape consiste à tester et à vérifier le bon fonctionnement du circuit implémenté dans le FPGA [16].

III.3 L'outil ISE Design Suite

La plate-forme la plus utilisée est ISE Design Suite, ce logiciel vise principalement le développement de micro logiciels embarqués pour les familles de produits de circuits intégrés Xilinx FPGA et CPLD [17].

L'outil ISE Design offre la possibilité de mettre en œuvre des conceptions schématiques en VHDL et en Ver log. Il intègre différents outils permettant de passer à travers tout le flot de conception d'un système numérique :

- Un éditeur de textes, de schémas et de diagrammes d'état.
- Un compilateur VHDL/Verilog.
- Un outil de simulation.
- Des outils pour la gestion des contraintes temporelles.
- Des outils pour la synthèse.
- Des outils pour la vérification.
- Des outils pour l'implémentation sur FPGA.

III.3.1 Lancement et création du projet

Le lancement de l'outil ISE se fait en cliquant deux fois sur l'icône présentée sur la figure III.2.

Figure III.2 Icône de l'outil ISE.

Après le lancement de l'outil ISE, l'interface se présentée comme montre la figure III.3.

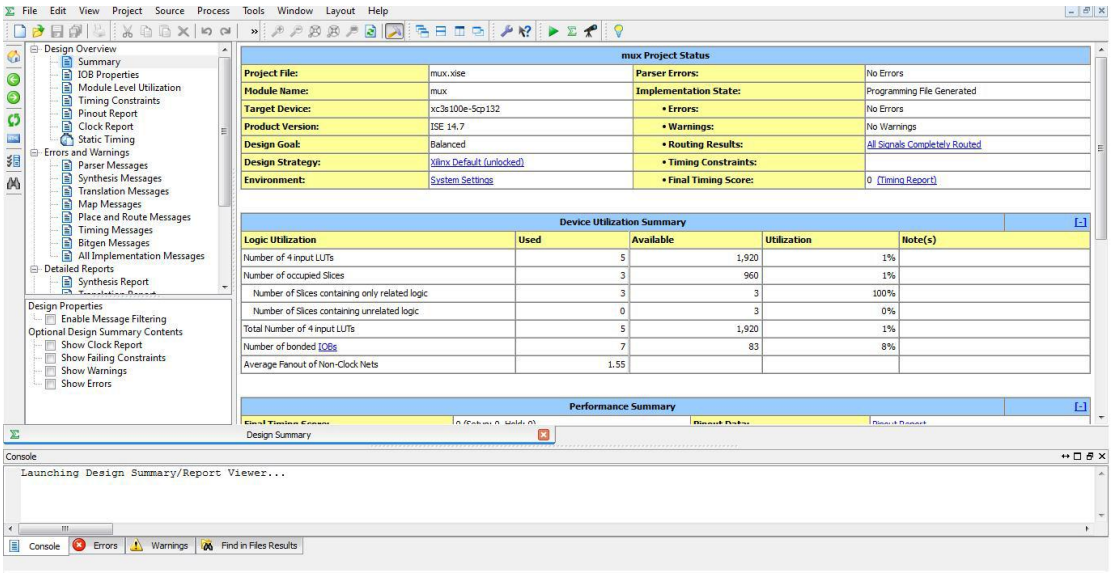


Figure III.3 Interface de l'outil ISE.

La création du projet se fait par le menu File->New Project (figure III.4).

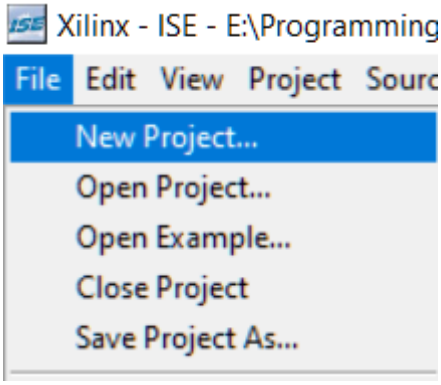


Figure III.4 Menu de création du projet.

Puis, on doit définir le nom et les paramètres du projet comme la famille du FPGA considéré et l'outil de programmation (VHDL ou Verilog). La fenêtre des paramètres du projet est décrite sur la figure ci-dessous (figure III.5).

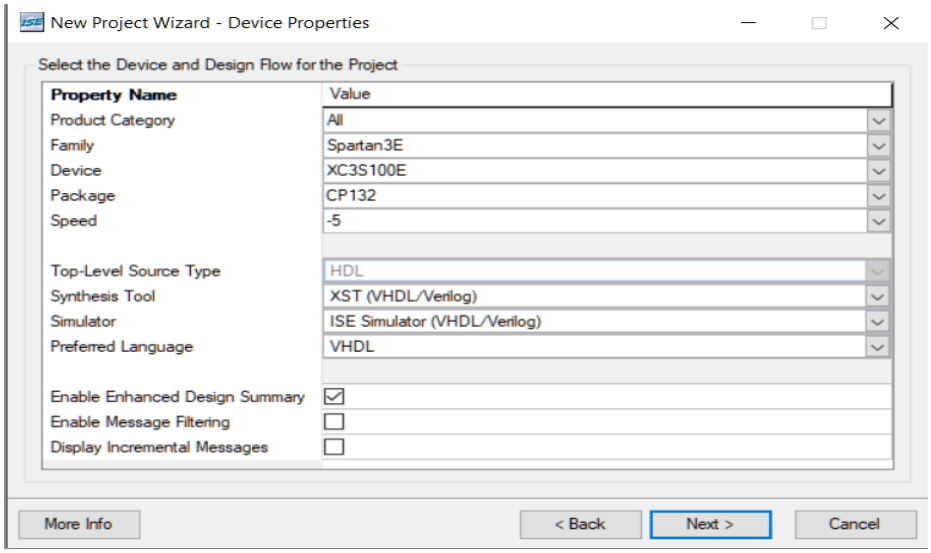


Figure III.5 Fenêtre des paramètres du projet.

La création du projet se termine avec une fenêtre qui décrit les paramètres du projet (figure III.6).

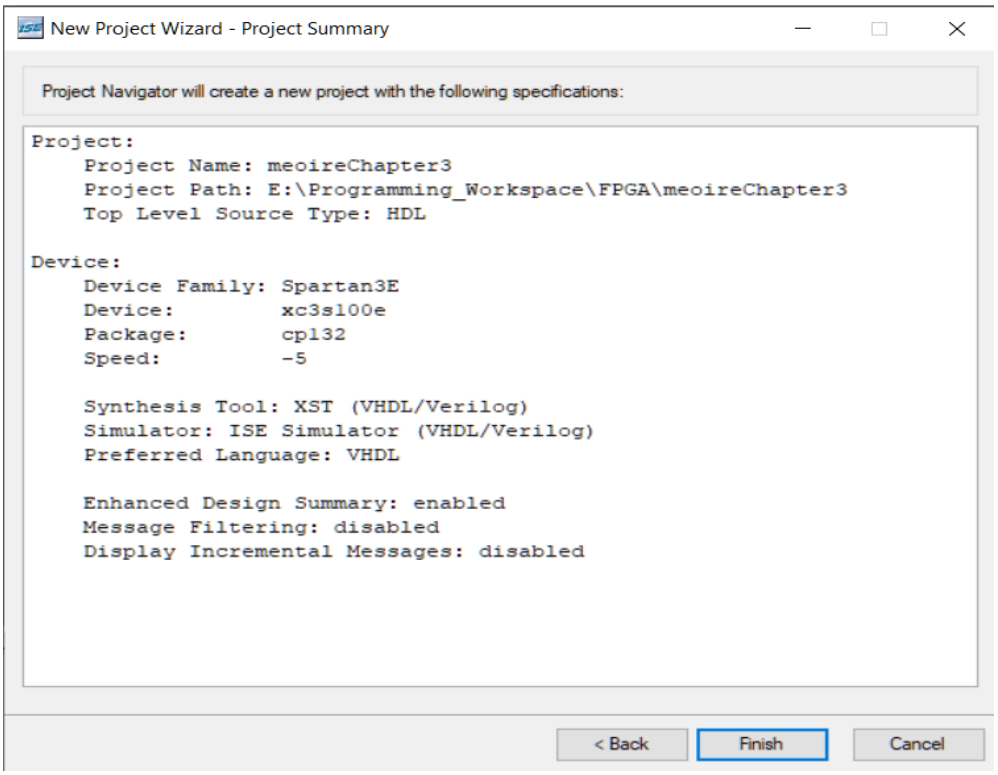


Figure III.6 Fenêtre de paramètres finaux du projet.

III.3.2 Création du fichier du code source

Afin de créer le fichier du code source en VHDL, il faut sélectionner le menu Project->New sources, une fenêtre s'affichera donc, comme montre la figure III.7.

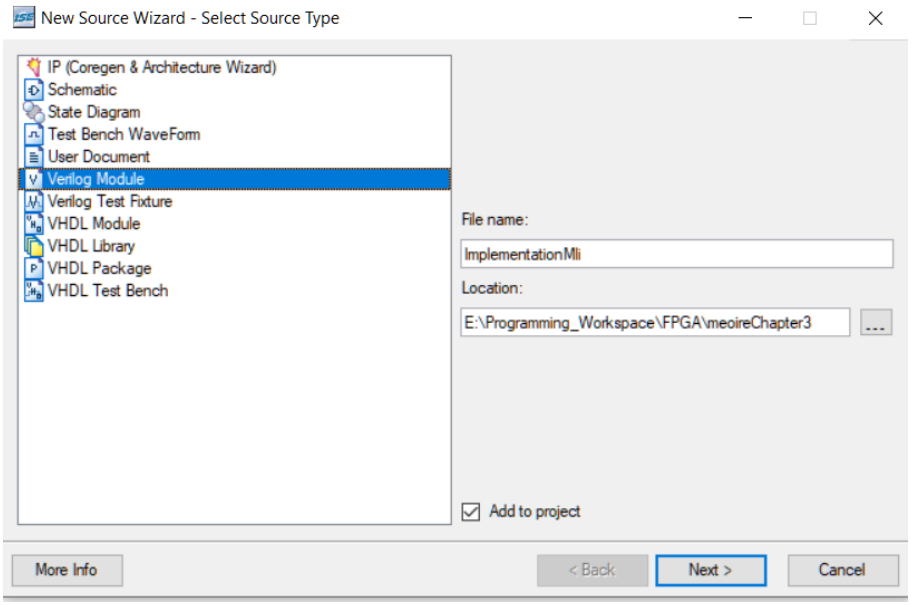


Figure III.7 Fenêtre permet la création du fichier du code source.

Après la création du fichier du code source (en VHDL), une fenêtre s'affichera pour permettre l'écriture du code VHDL, comme présente la figure III.8.

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25  ---- Uncomment the following library declaration if instantiating
26  ---- any Xilinx primitives in this code.
27  --library UNISIM;
28  --use UNISIM.VComponents.all;
29
30  entity module is
31  end module;
32
33  architecture Behavioral of module is
34
35  begin
36
37
38  end Behavioral;
39
40
```

Figure III.8 Fenêtre du code VHDL.

Sur la figure III.9, nous présentons un exemple de programme VHDL d'un multiplexeur 2 vers 1.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   entity multiplexeur is
4
5   PORT ( Y0 : IN STD_LOGIC ;
6   Y1 : IN STD_LOGIC ;
7   SEL : IN STD_LOGIC ;
8   Q : OUT STD_LOGIC
9   ) ;
10  end multiplexeur;
11
12  architecture Behavioral of multiplexeur is
13  SIGNAL A : STD_LOGIC ;
14  SIGNAL B : STD_LOGIC ;
15  BEGIN
16  Q <= A OR B ;
17  A <= NOT SEL AND Y0 ;
18  B <= SEL AND Y1 ;
19  end Behavioral;
```

Figure III.9 Exemple d'un code VHDL (MUX 2 vers 1) sous ISE Design.

La synthèse et la création des modules du projet se font en appuyant sur le bouton vert du menu de la figure III.10.

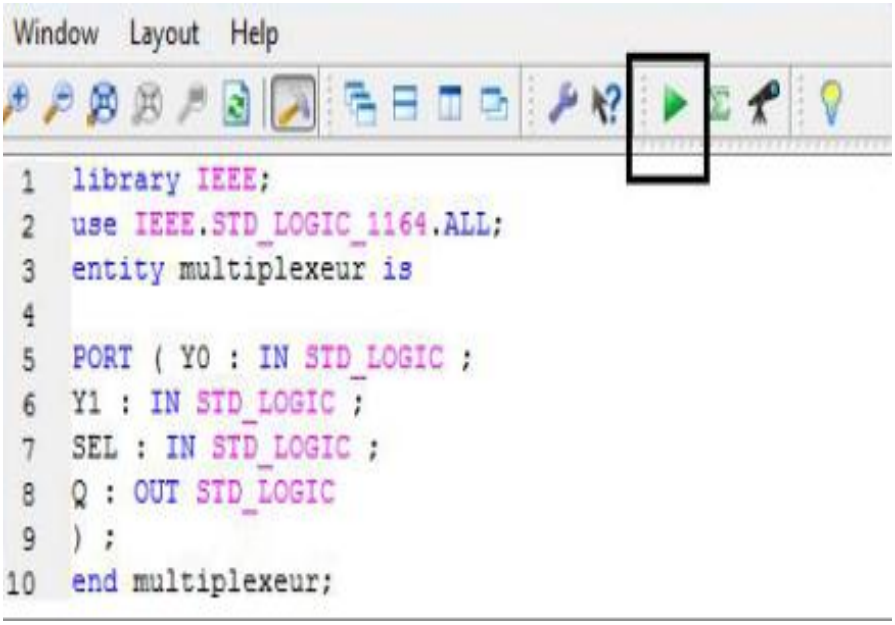


Figure III.10 Fenêtre décrivant la manière de compilation du projet.

Après cette étape et dans le cas d'une bonne compilation, on obtient une fenêtre avec les résultats de la compiltaion / Build (figure III.11).

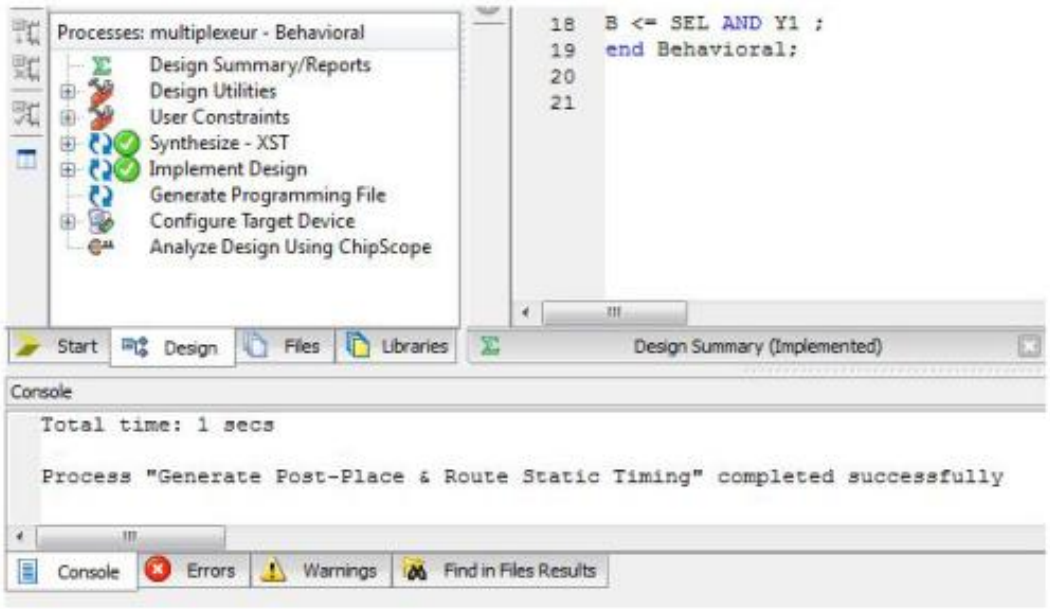


Figure III.11 Fenêtre décrivant les résultats de la compilation/ Build.

III.3.3 Création du fichier ".UCF"

À cette étape, la conception du projet sur l'ISE est réussite, mais elle reste la création du fichier ''.UCF'' permettant de relier les entrées/sorties du module VHDL avec les pines du FPGA considéré au début de la création du projet.

La création du fichier ‘’.UCF’’ est similaire à celle du fichier source, il faut sélectionner le menu Project-> New sources, une fenêtre s’affichera donc, comme illustre la figure III.12.

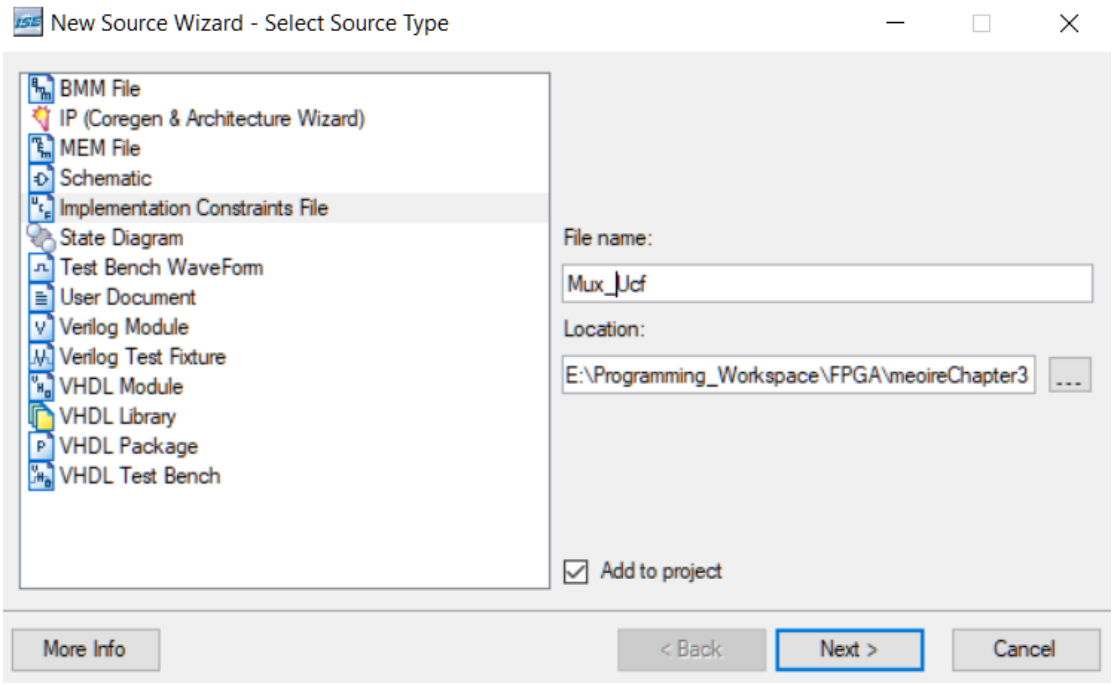


Figure III.12 Fenêtre de la création du fichier ‘’.UCF’’.

Après la création du fichier ‘’.UCF’’, nous utilisons souvent les instructions (NET et LOC) représentés sur la figure III.13.

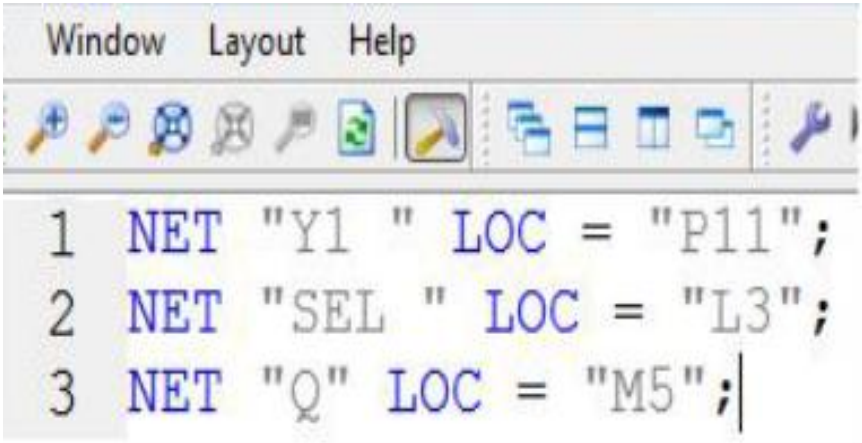


Figure III.13 Fenêtre du code du fichier ‘’.UCF’’.

III.3.4 Exécution complète du projet

La dernière étape consiste à faire une exécution complète du projet afin de générer le fichier « bitestream » permettant de programmer le FPGA considéré au début de la création du projet. Cette exécution est illustrée sur l’exemple de la figure III.14.

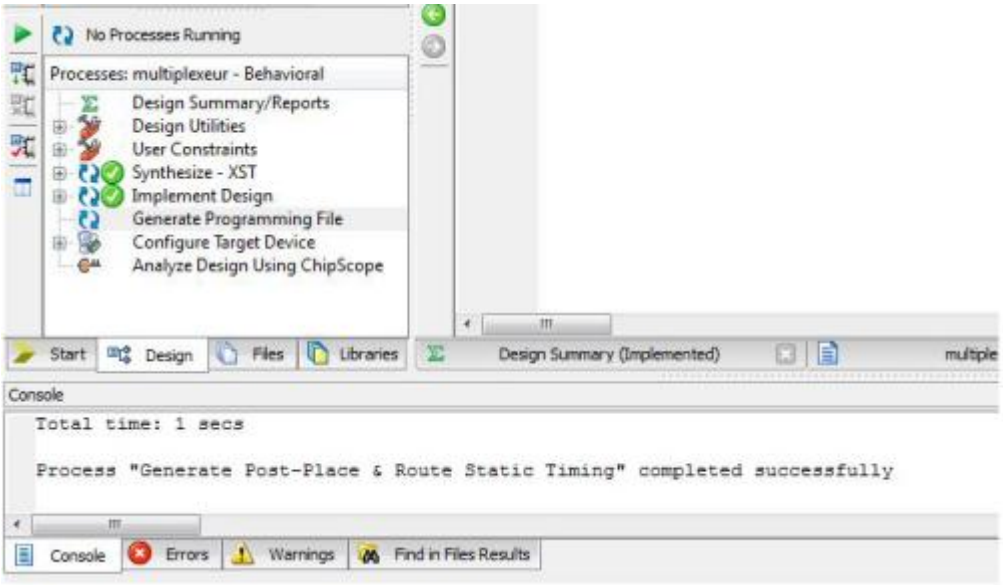


Figure III.14 Fenêtre de l'exécution (réussie) du projet complet.

En effet, la conception du projet sur ISE permet donc, de :

- Regrouper les composantes obtenues lors de la synthèse dans des blocs spécifiques du FPGA (le Mapping).
- Choisir des endroits spécifiques sur le FPGA où disposer les blocs utilisés, et choisir les pattes du FPGA correspondant aux ports d'entrée et de sortie.
- Etablir des connexions électriques entre les blocs utilisés (Routage).
- Convertir toute ces configurations en un fichier binaire (appelé ''bite fille'') pour la programmation du FPGA considéré.

III.4 L'outil Adept

L'outil Adept est une application développée par Digilent afin de permettre l'interfaçage des cartes à base de circuits FPGA avec les PCs. Elle permet donc la programmation de la carte BASYS 2 ainsi que le circuit Spartan 3E.

La programmation de la carte BASYS 2 avec l'outil Adept est décrite par le schéma de la figure III.15 [16, 18].

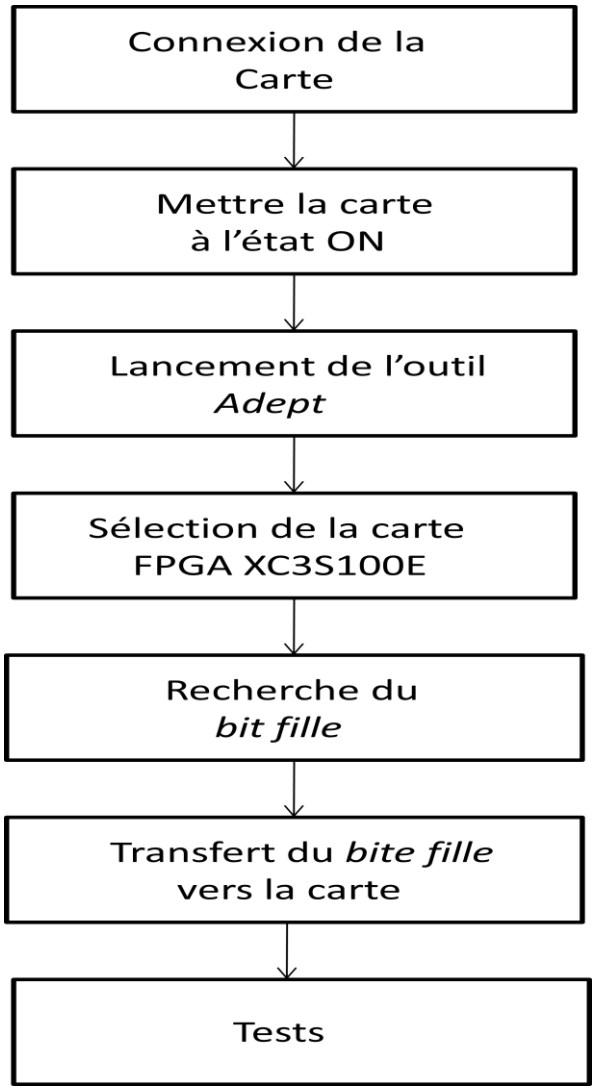


Figure III.15 Principales étapes de la programmation de la BASYS 2.

La première étape consiste à connecter la carte BASYS 2 avec le PC via le port série de l'USB 2.0.

- Dans la deuxième étape, il faut allumer la carte par le Switch On/OFF présenté sur la figure.
- Dans la troisième étape, il faut lancer l'outil Adept (double cliquer sur l'icône présentée sur la figure III.16).

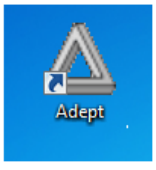


Figure III.16 Icône de l'outil Adept.

Après le lancement du logiciel, l'interface de l'outil Adept affichera la carte considérée, comme présente la figure III.17.

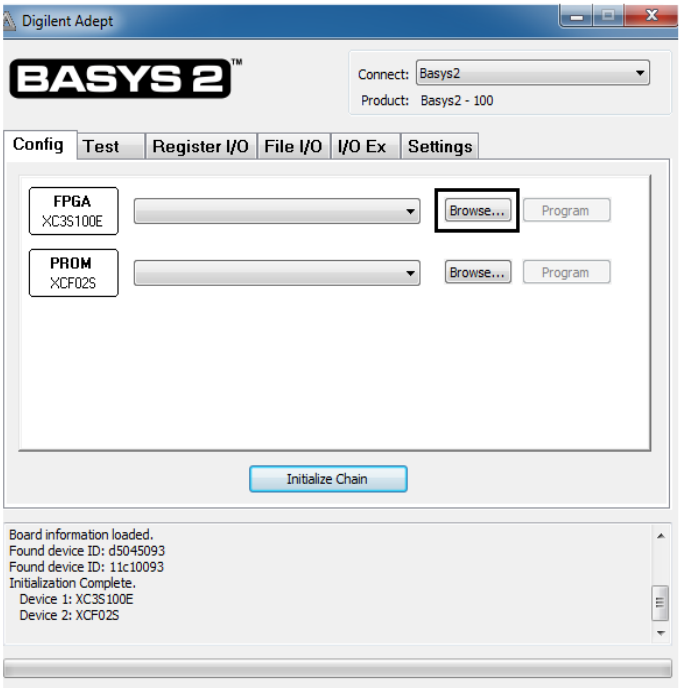


Figure III.17 Interface de l'outil Adept.

La recherche du « bite fille » est réalisée en appuyant sur le bouton « Browse » présenté sur la figure ci-dessus (quatrième étape).

- Dans la cinquième étape, la programmation est réalisée en appuyant sur le bouton « Program » présenté sur la figure III.18.

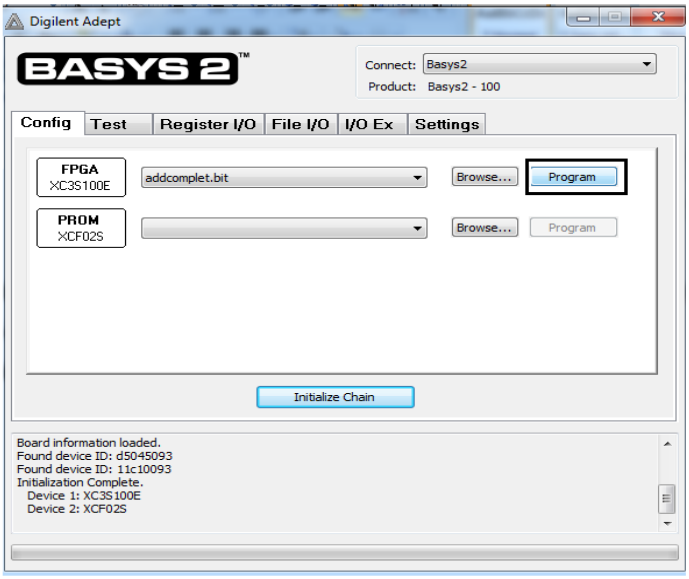


Figure III.18 Interface de l'outil Adept en phase de programmation.

La dernière étape permet de tester et de vérifier le fonctionnement du circuit implémenté sur le FPGA en utilisant les composants d'entrées/sorties de la carte).

III.5 La carte Basys 2

On considère la carte BASYS 2 de la société DIGILENT. Cette carte se base sur l'utilisation d'un FPGA de la famille Spartan de Xilinx (figure III.19) [19].

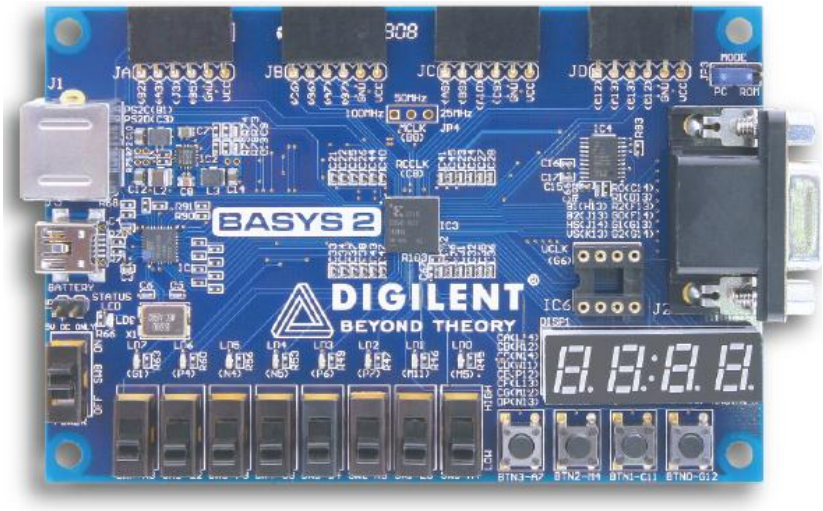


Figure III.19 Carte BASYS 2.

En effet, la carte BASYS 2 se base sur un FPGA Xilinx de la famille Spartan 3E-250 et des composants d'entrées/sorties comme montre la figure III.20.

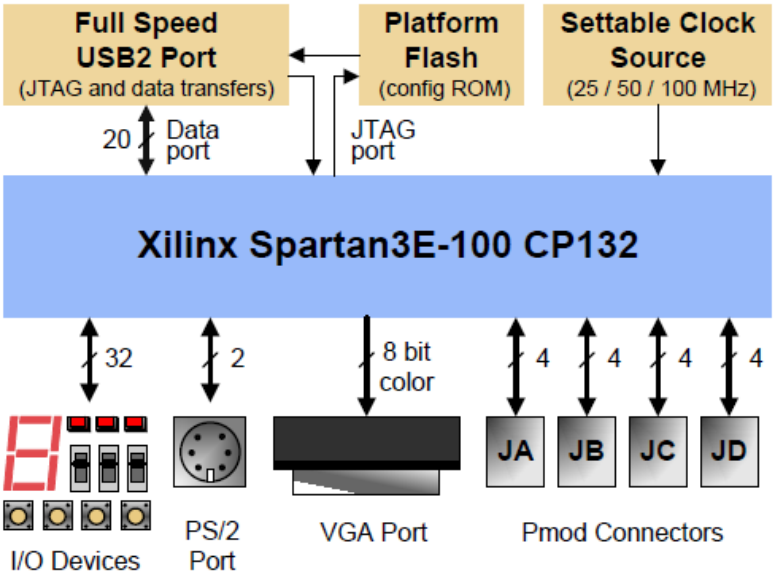


Figure III.20 Principaux composants formant la carte BASYS 2.

III.5.1 Les composants d'entrées/sorties

Les entrées/sorties logiques sont réalisées par : [19]

- Huit interrupteurs dénommés SW0 à SW7 ;
- Quatre boutons poussoirs dénommés BTN0 à BTN3 ;
- Huit LEDs (LD0 à LD7) ;
- Quatre afficheurs 7 segments multiplexés (DISP1) ;

Ceci comme montre la figure III.21.

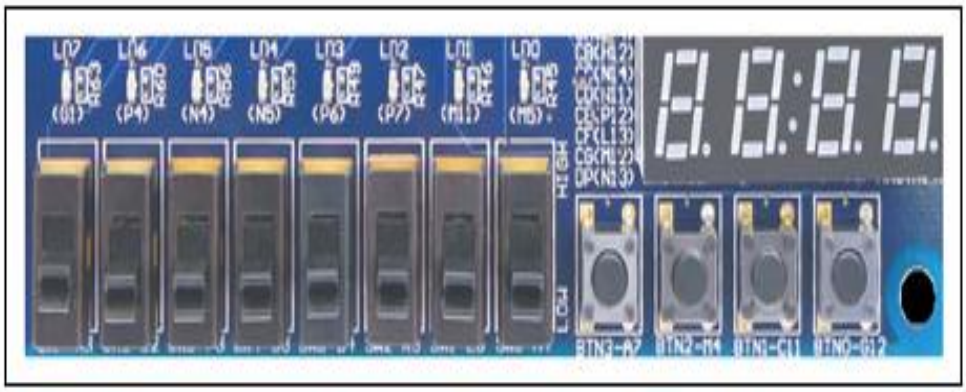


Figure III.21 Composants d'entrées/sorties de la BASYS 2.

III.5.2 L'oscillateur

Un oscillateur à base de Silicium est considéré afin de générer un signal d'horloge d'une fréquence stable de 25, 50 et 100MHz (figure III.22).

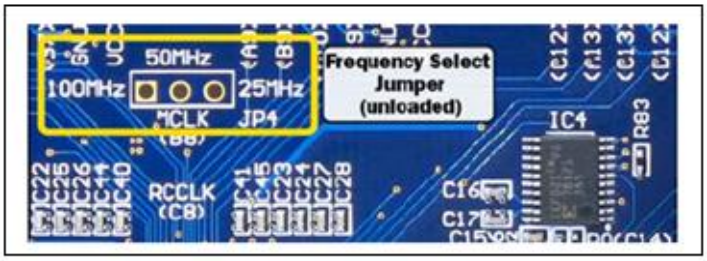


Figure III.22 Composant de sélection de fréquences de l'oscillateur.

III.5.3 L'alimentation

La carte est alimentée à travers une batterie de 5.5V ou le port USB 2.0 permettant de réaliser une communication série avec un ordinateur (figure III.23).

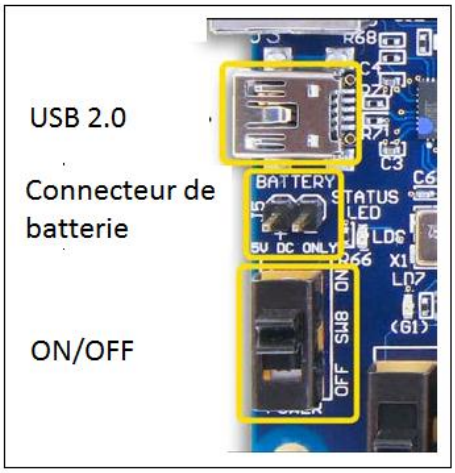


Figure III.23 Alimentation de la BASYS 2.

III.5.4 Le Spartan 3E

La carte BASYS 2 considère un FPGA de la société XILINX de la famille Spartan 3E-100 CP132 (figure III.24).

Les caractéristiques générales du circuit Spartan 3E sont :

- 100000 portes logiques.
- 240 CLB.
- 960 SLICE.
- Multiplicateurs de 18bits.
- 72 K-bite de blocs mémoire de type RAM. [20]

Les pines du circuit Spartan 3E sont connectées aux composants d'entrées/sorties de la carte comme présente la figure III.24.

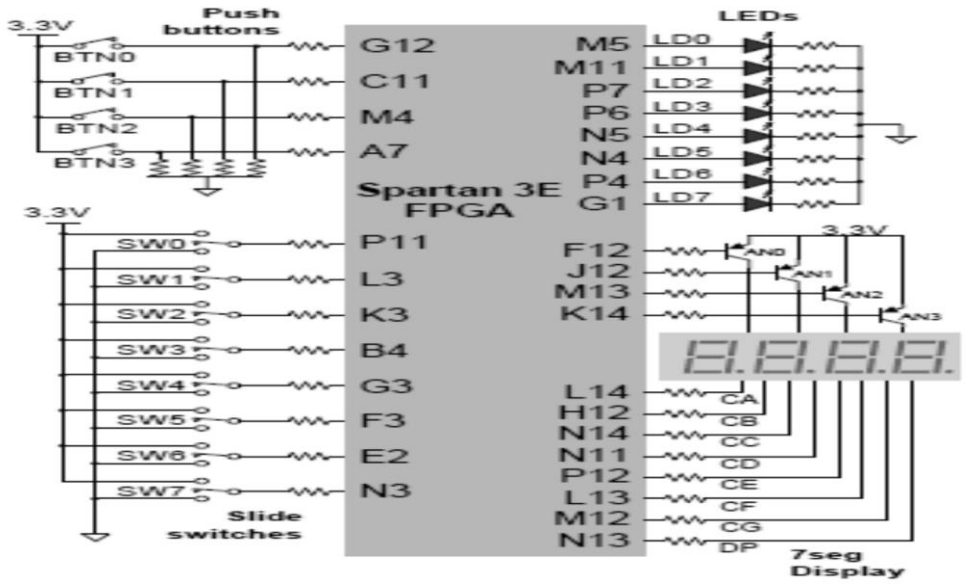


Figure III.24 Brochage et connexion des composants d'entrées/sorties avec le Spartan 3E.

III.5 Conclusion

Dans ce chapitre, nous avons décrit le matériel et le Software utilisés dans le cadre de ce projet. Nous donc avons exposé la carte Byasys 2 et le FPGA Spartan 3E considéré comme cible sur lequel nous implémentons notre technique.

Nous avons présenté l'outil ISE Design et également et commet utilisé ce logiciel pour la création et la compilation d'une description VHDL en considérant un FPGA de la famille Xilinx. Dans ce cas, nous avons illustré les différentes étapes qui permettent la conception et l'élaboration d'un projet sous l'environnement ISE Design tout accompagné de la description de la carte Spartan 3E.

Dans le chapitre suivant (chapitre IV), nous allons illustrés l'ensemble des résultats obtenus par simulation ainsi que par l'expérimentale.

Chapitre IV

Conception, implémentation et réalisation

IV.1. Introduction

Notre objectif principal est d'implémenté une MLI et de réaliser un générateur d'impulsions modulés en largueur à base du FPGA-Spartan3E et à l'aide du langage VHDL. Dans ce cas, nous allons mettre en œuvre une carte Basys 2 et un oscilloscope afin de visualiser et de vérifier les signaux obtenus.

Les étapes proposées pour réaliser cette conception sont englobées dans les points suivants :

- Développement d'un organigramme du programme principal écrit en VHDL.
- Création du code VHDL sous XILINX 9.2i.
- Synthèse et implémentation du programme sur le FPGA Spartan3E.
- Visualisation des signaux numériques du FPGA en utilisant TESTBENCH.
- Visualisation des signaux par un oscilloscope.

IV.2. Description du système

Afin de réaliser le générateur, on doit implémenter la MLI Standard sur le circuit FPGA-Spartan3E. Pour cela, on considère un système composé de 10 signaux d'entrés, d'un signal de sortie. La fonction de ses signaux est illustrée sur le tableau IV.1.

Tableau IV.1 : Description des signaux d'entrées –sorties du système.

Signal	**Type**	**Description**
Data0	Entrée	Data0.7 est un bus de 8 bits qui sert a sélectionné le rapport cyclique du MLI, ce qui fait 256 niveaux du rapport cyclique. La taille de bus est limitée par le nombre des switches qui existent dans la carte. Un exemple du choix du rapport cyclique est de mettre le bus Data a (10000000) b qui est 192 en décimal ce qui fait un rapport de 75%.
Data1	Entrée	
Data2	Entrée	
Data3	Entrée	
Data4	Entrée	
Data5	Entrée	
Data6	Entrée	
Data7	Entrée	
Reset	Entrée	Remise à zéro du compteur et de l'impulsion à l'état 0.
Clock	Entrée	L'horloge
PWM	Sortie	Signal module en largeur.

Le schéma global du système considéré est décrit par la figure IV.1.

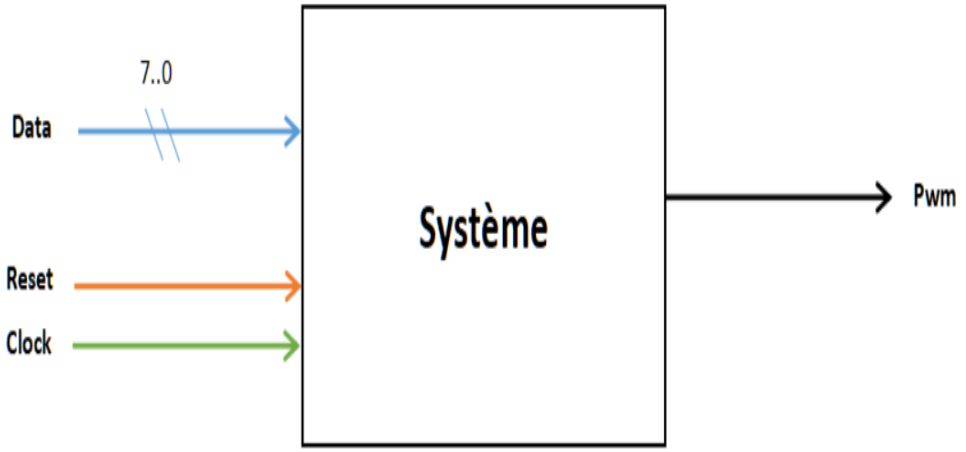


Figure IV.1 : Description global du système.

Le système est formé de trois principaux composants, qui sont :

- Un registre qui sert à stocker les valeurs Data (fixant le rapport cyclique) ;
- Un compteur qui sert à faire le comptage à partir de l'horloge (CLK) ;
- Un comparateur qui sert à comparer la valeur du compteur avec la valeur du registre, si la valeur (registre)= valeur (compteur), la sortie est « 1 », sinon « 0 » ,et une bascule RS pour la mise en 1 ou 0 selon la sortie du comparateur.

La structure interne du système est représentée par la figure IV.2.

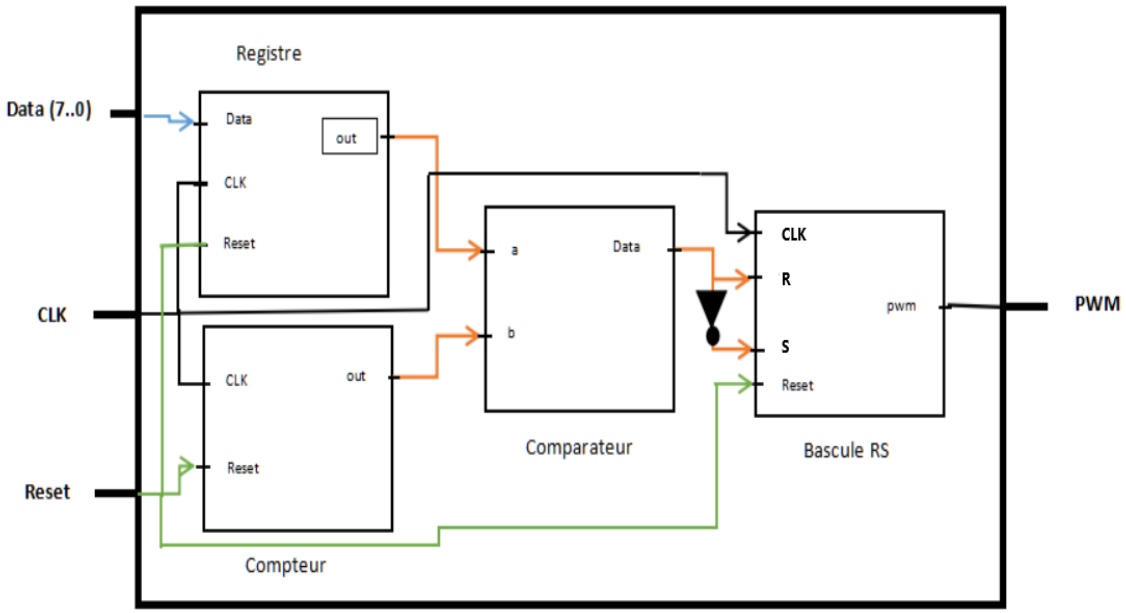


Figure IV.2 : Structure interne du système.

IV.3. L'organigramme

Après avoir définit la structure interne du système, l'étape suivante est d'élaborer un organigramme décrivant les principaux blocs formant le système et simplifiant ainsi l'écriture du programme en VHDL. Ce dernier est constitué de quatre composants comme l'avait mentionné précédemment, la valeur de sortie de chaque composant est mis à zéro, leurs organigrammes sont définis comme suivant :

a) **le registre :** il sert à décrire les valeur des entrées « Data », ce composant va tester la présence d'un front montant d'horloge, dans le cas du front montant, la valeur d'entrée « Data » et mémorise, sinon il va garder la valeur précédente et la valeur initiale de ce registre et met à zéro, ceci comme montre l'organigramme de la figure IV.3.

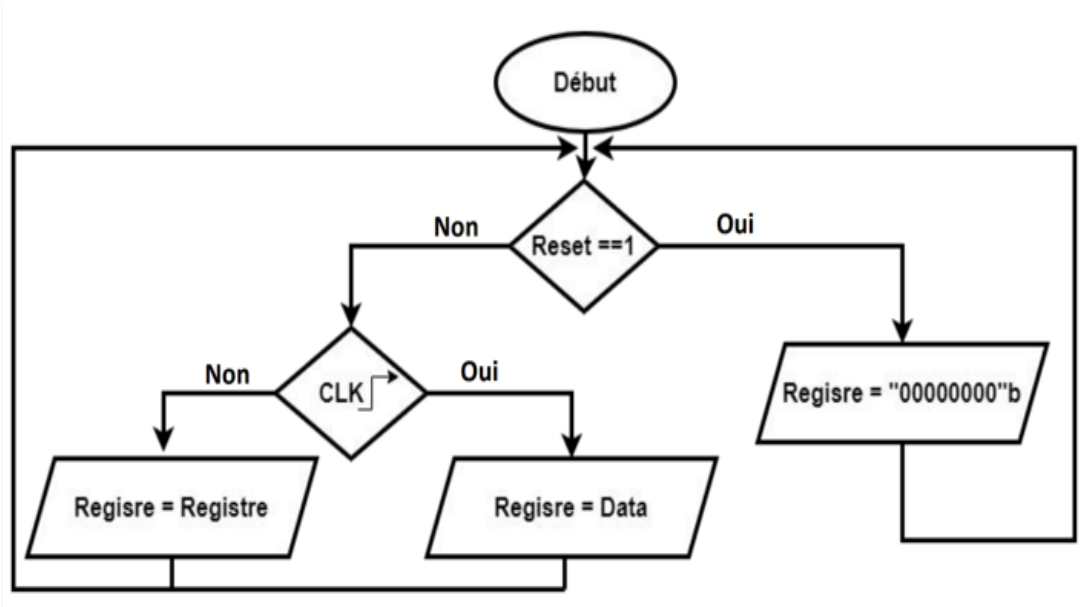


Figure IV.3 : Organigramme décrivant le fonctionnement du registre.

b) **Compteur :** il permet de compter les cycles d'horloge, sa valeur initiale est met à zéro, le compteur va s'incrémenter dans le cas du front montant jusqu'à sa valeur maximale, ensuite lors du débordement, il va se réinitialiser à zéro (figure IV.4).

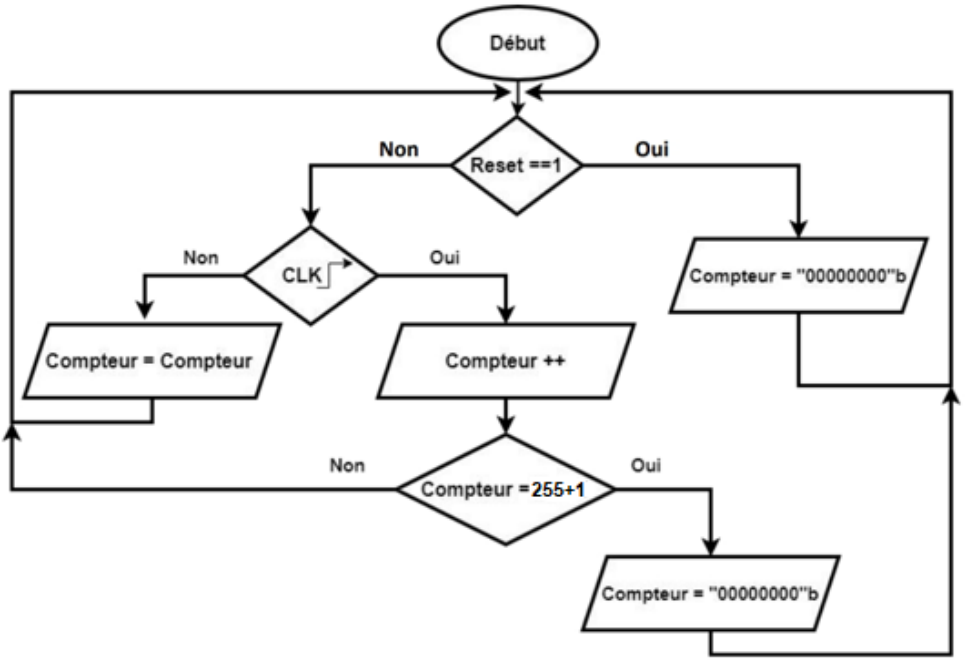


Figure IV.4 : Organigramme décrivant le fonctionnement du Compteur.

c) **Comparateur** : il a comme but de comparer les sortie du registre avec celle du compteur, si les sorties sont égaux, la sortie du compteur est met a « 1 », sinon la sortie est met à « 0 » (figure IV.4).

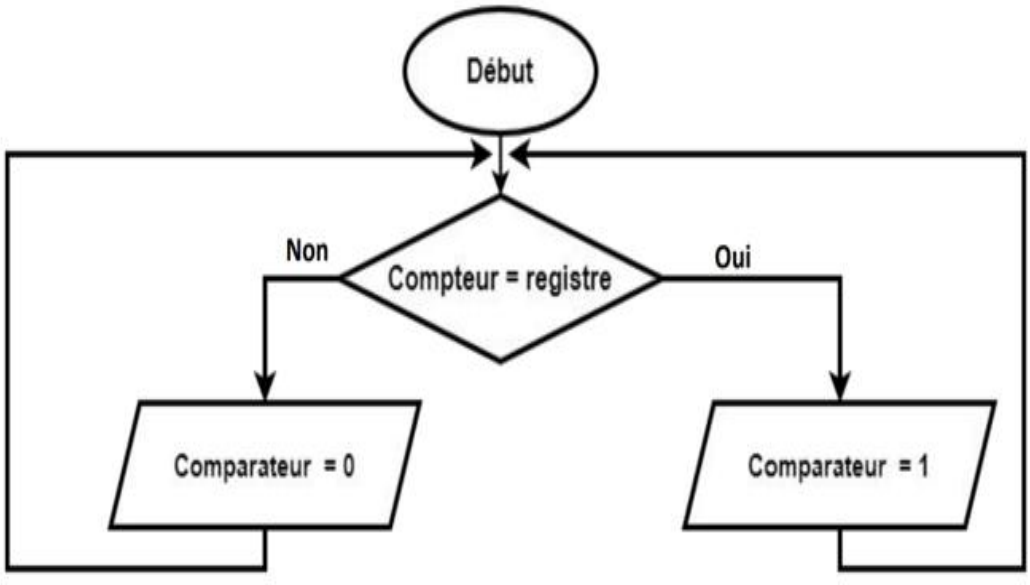


Figure IV.5 : Organigramme décrivant le fonctionnement du comparateur

d) **Bascule RS :** elle permet la mise à « 0 » ou à « 1 » créant ainsi la MLI, elle sert à mémoriser la valeur du MLI, ça se fait par le basculement de sa sortie lorsqu'il y aura un front montant dons son entrée (la valeur de comparateur) comme illustre la figure IV.6.

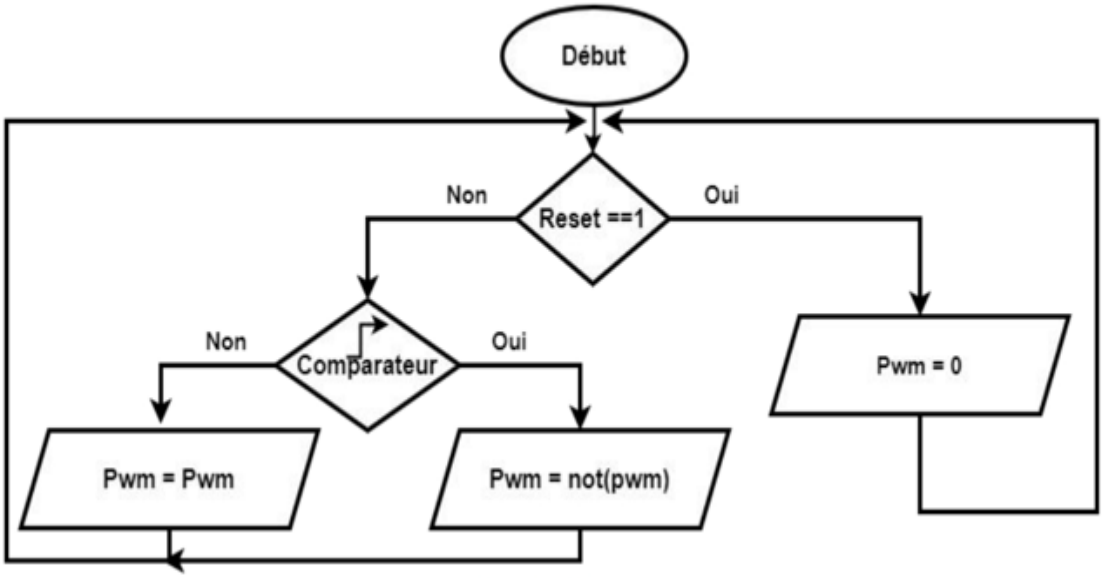


Figure IV .6 : Organigramme décrivant le fonctionnement de la bascule « RS ».

Après élaboration des organigrammes relatifs à chaque composante, il reste de définir l'organigramme global du système (figure I.V.7).

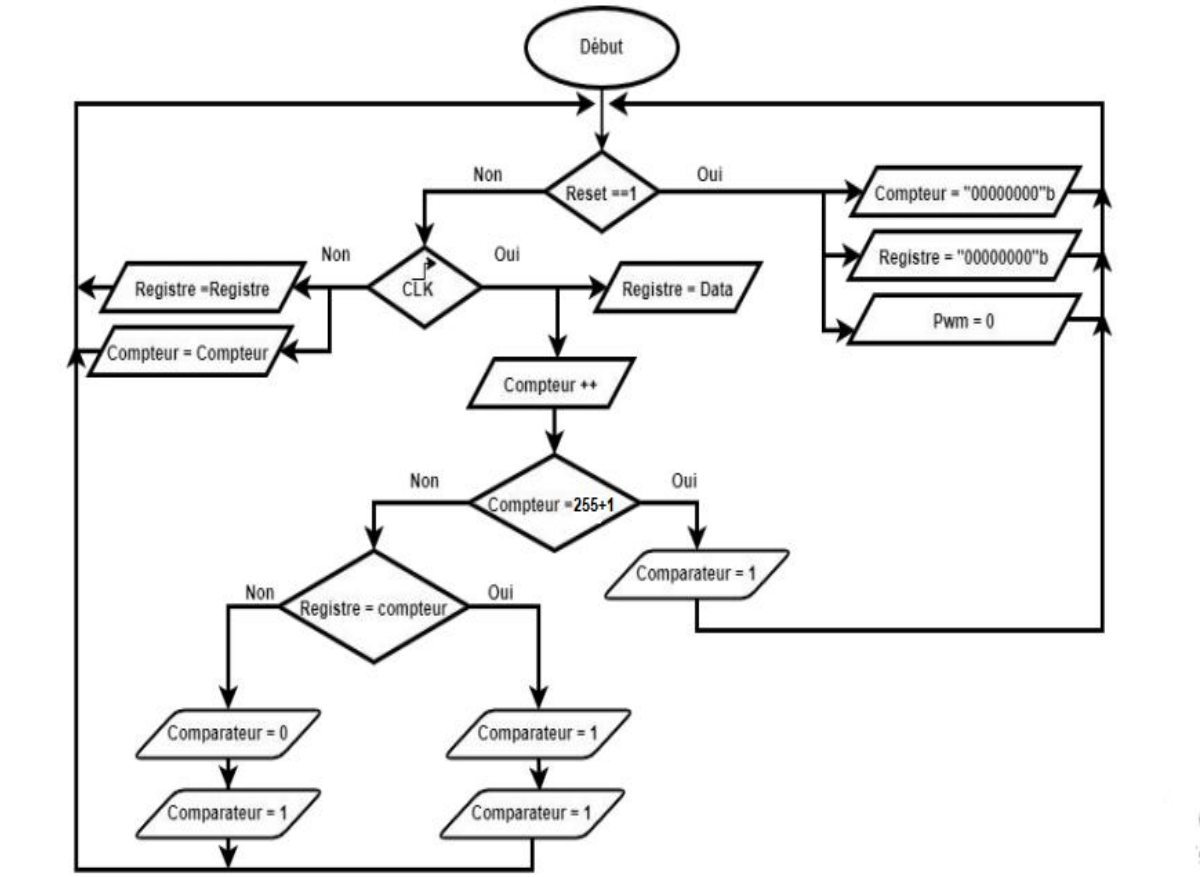


Figure IV.7 : Organigramme global.

IV.4. Synthèse et simulation

Pour implémenter la MLI sur FPGA, il faut tout d'abord créer le code VHDL qui traduit l'organigramme précédent, dans ce cas, le langue VHDL est utilisé à l'aide du logiciel Xilinx 9.2i. Toute description (code) VHDL est composé de :

- partie entité " Entity en anglais" : dans cette tranche, les signaux entrées/sorties du circuit sont déclarés.

-partie architecture : cette partie représente l'algorithme qui décrit le fonctionnement du circuit.

La démarche pour créer le code ce fait comme suivant :

Tout d'abord avant créer le code, il faut importer les paquets qui permettent d'utiliser les opérations arithmétiques et les opérations logiques.

Puis on commence par la création de l'entité (le boîtier du circuit), ça se fait par la déclaration des signaux d'entrée/sortie discutés auparavant.

Après la déclaration des signaux, il reste à implémenter l'algorithme qui sert à manipuler ses signaux afin de pouvoir construire le signal MLI souhaité, ça se fait par la création d'architecture :

Au début, les signaux interne qui portent les valeurs de registre, compteur, comparateur et la sortie de la bascule sont déclarés, puis l'algorithme est implémenté, elle est composée de quatre processus, chaque processus décrit le fonctionnement de chaque composant du système.

En fin les signaux sont synchronisés.

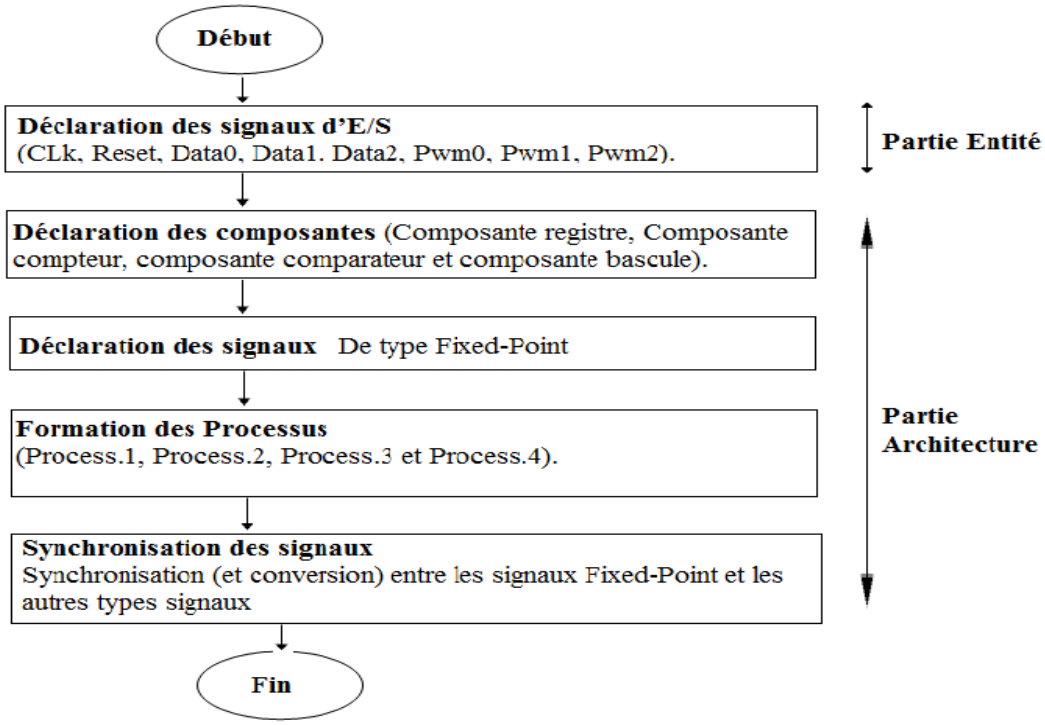


Figure IV.8 : Organigramme du programme principal de la technique MLI sous VHDL.

IV.5. Résultat de l'implémentation

Dans cette partie, nous présentons les principaux résultats obtenus à partir de l'implémentation d'un MLI sous VHDL de type Spartan3E et sur un FPGA-XILINX, cela en utilisant l'outil ISE Design de XILINX.

IV.5.1. Schémas RTL

La figure IV.9 présente le schéma RTL représentant la ''boite noire'' du système implémenté sous ISE Design en utilisant le code développé en langage VHDL.

Figure IV.9 : Schéma RTL du système.

IV.5.2 Schéma RTL global

Sur la figure IV.10 nous présentons le schéma global du système implémenté sous ISE Design en utilisant le code développé en langage VHDL.

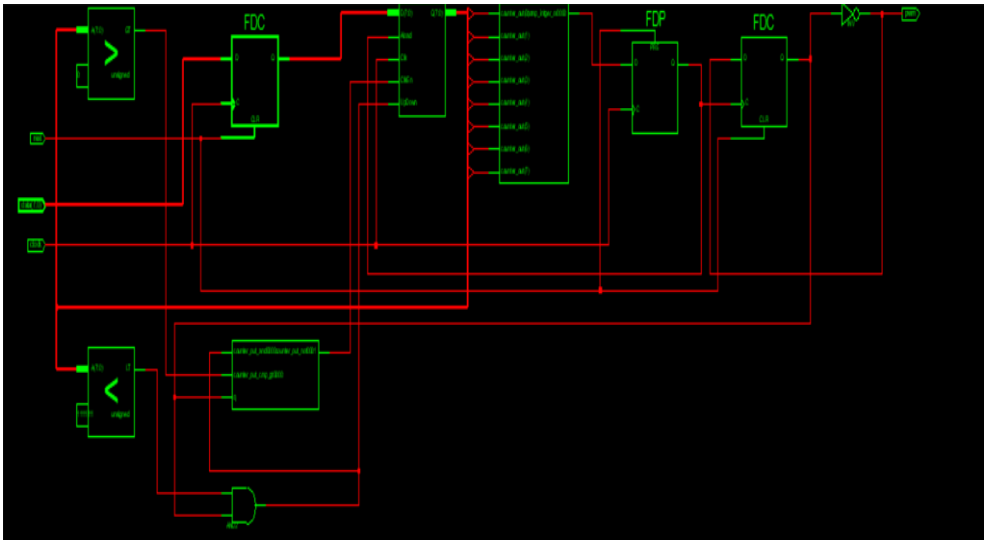


Figure IV.10 : Schéma RTL global du système implémenté.

La figure IV.10 montre le schéma globale du circuit MLI et ses composants, le circuit est peu différente à celui discuté précédemment, ce c'est à cause de l'algorithme implémentée, ça nécessite la vérification du signal de sortie du MLI, pour cela un Test Bench a été fait pour visualiser le signal MLI avec des valeurs Data différentes.

IV.5.3.Visualisation des signaux via Test bench

Le test bench est un outil qui permet de visualiser le comportement temporel des signaux du circuit, dans les figures qui suivent, on peut voir les générations de ces signaux comme il était prévu, notamment le signal MLI (PWM dans les figures).

Dans la figure IV.11, la valeur du « data » est égale à $(192)_{10}$= $(11000000)_2$ ce qui donne une MLI avec un rapport cyclique de 75%.

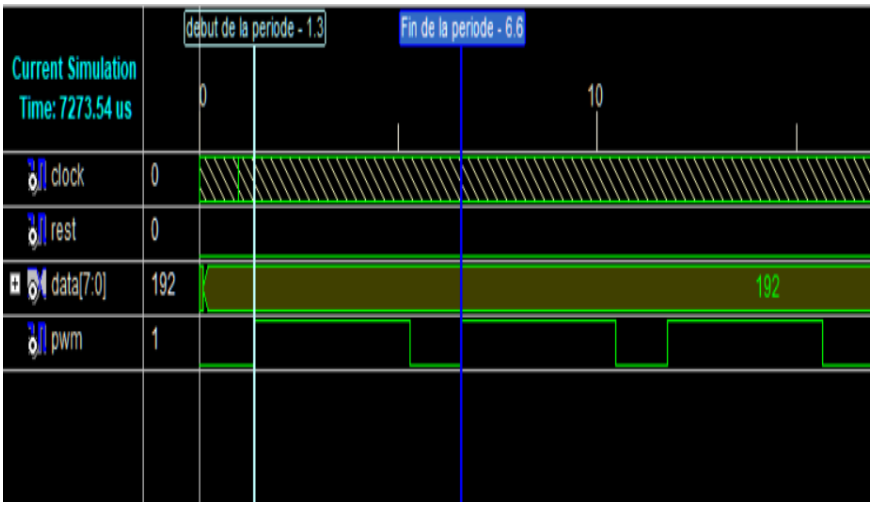
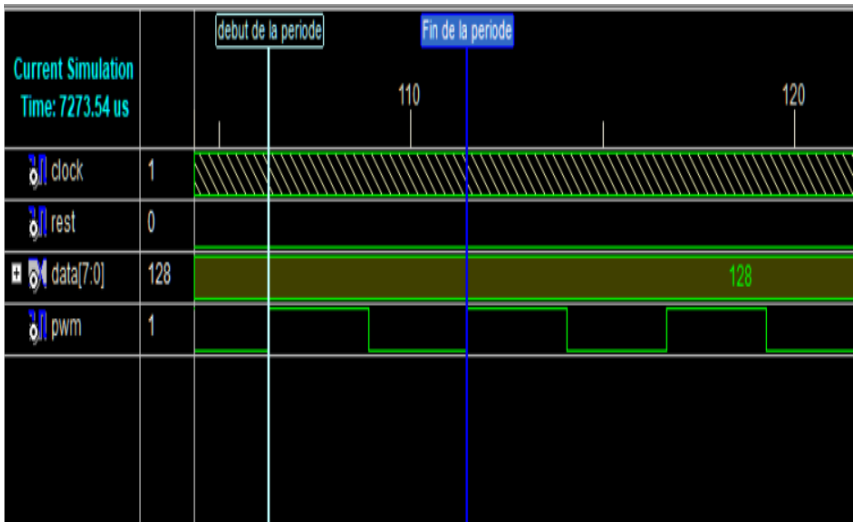


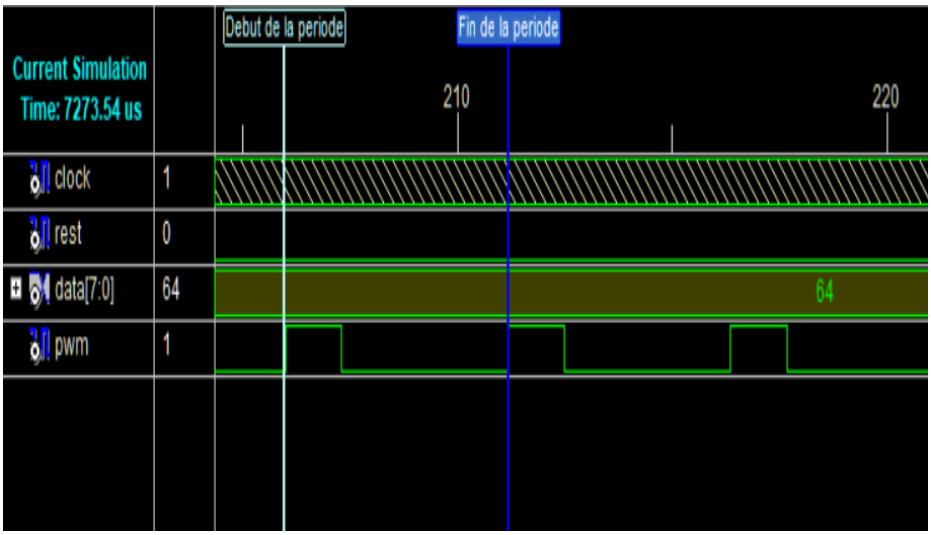
Figure IV.11 : Variation des signaux, PWM0 (rapport cyclique de 75%).

Dans la figure IV.12, la valeur « data » est égale à $(128)_{10}$= $(10000000)_2$ ce qui fait un rapport cyclique de 50%.



FigureIV.12 : Variation des signaux, PWM0 (rapport cyclique de 50%).

Dans la figure IV.13, la valeur « data » est égale à $(64)_{10}$= $(01000000)_2$ ce qui fait un rapport cyclique de 25%.



FigureIV.13 : Variation des signaux, PWM0 (rapport cyclique de 25%).

IV.6. Implémentation sur Spartan3E

Après avoir simulé les le circuit et tester l'exactitude des signaux, notamment le signal MLI, c'est le temps d'implémenter ce circuit dans la carte Spartan3, pour cela il faut créer un autre code pour relier les signaux entrés/sorties déclarés auparavant (clock, reset, data et PWM).

Les principales lignes du code de fichier de contraint "UCF", sa syntaxe est comme suit :

```
NET "clock" LOC = "B8";
NET "reset" LOC = "G12";
NET "data<7>" LOC = "N3";
NET "data<6>" LOC = "E2";
NET "data<5>" LOC = "F3";
NET "data<4>" LOC = "G3";
NET "data<3>" LOC = "B4";
NET "data<2>" LOC = "K3";
NET "data<1>" LOC = "L3";
NET "data<0>" LOC = "P11";
NET "PWM" LOC = "B2» ;
```

Le code utilise les directives "NET" et "LOC", la première sert à relier le signal avec une adresse spécifiée par la directive "LOC", un exemple de sa ; considérant la première ligne : NET "clock" LOC = "B8", sa signifie : relie le signal "clock" avec l'emplacement

"B8" ou se trouve l'horloge interne du FPGA. La figure suivante illustre le fonctionnement du code.

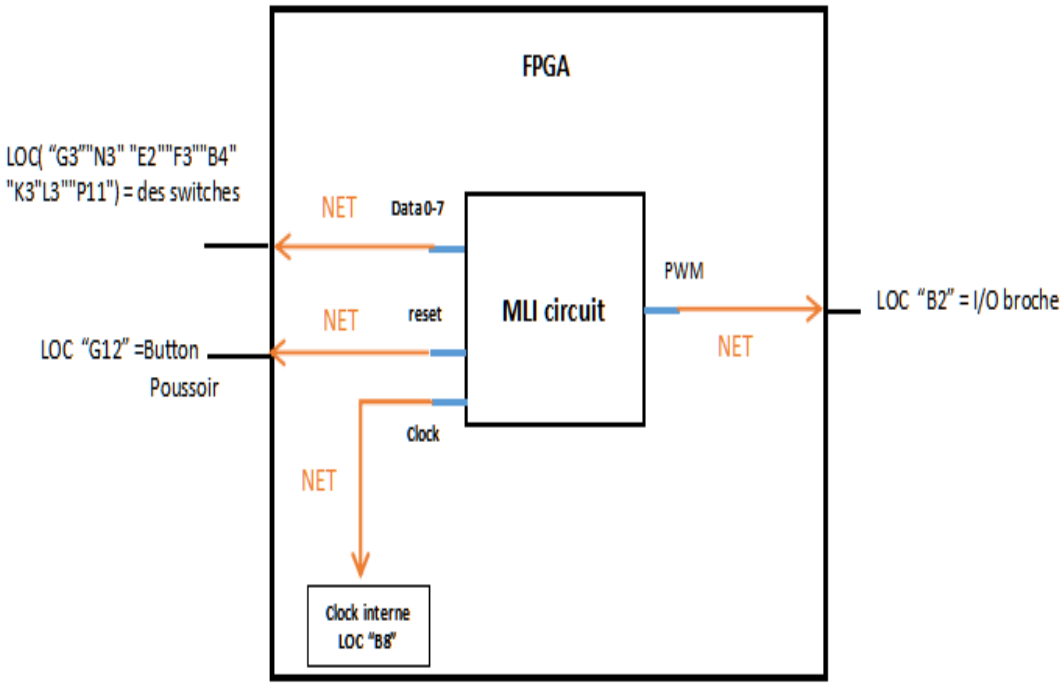


Figure IV.14 : Schéma descriptif du fichier de construction.

La dernière étape consiste à faire une exécution complète du projet afin de générer le fichier binaire permettant de programmer le FPGA considéré au début de la création du projet, cette étape consiste à :

- Regrouper les composantes obtenues lors de la synthèse dans des blocs spécifiques du FPGA (le Mapping).
- Choisir des endroits spécifiques sur le FPGA où disposer les blocs utilisés, et choisir les pattes du FPGA correspondant aux ports d'entrée et de sortie.
- Établir des connexions électriques entre les blocs utilisés (Routage).
- Convertir toute ces configurations en un fichier binaire (appelé ''bite fille'') pour la programmation du FPGA considéré.

Après avoir générer le fichier binaire, c'est le temps de programmer la carte Spartan-3E,pour le faire, un logiciel fourni par le fabricant de la carte, ce programme est appelé DIGILENT Adept. ce logiciel est livré avec une interface simple qui facilite la programmation de la carte.

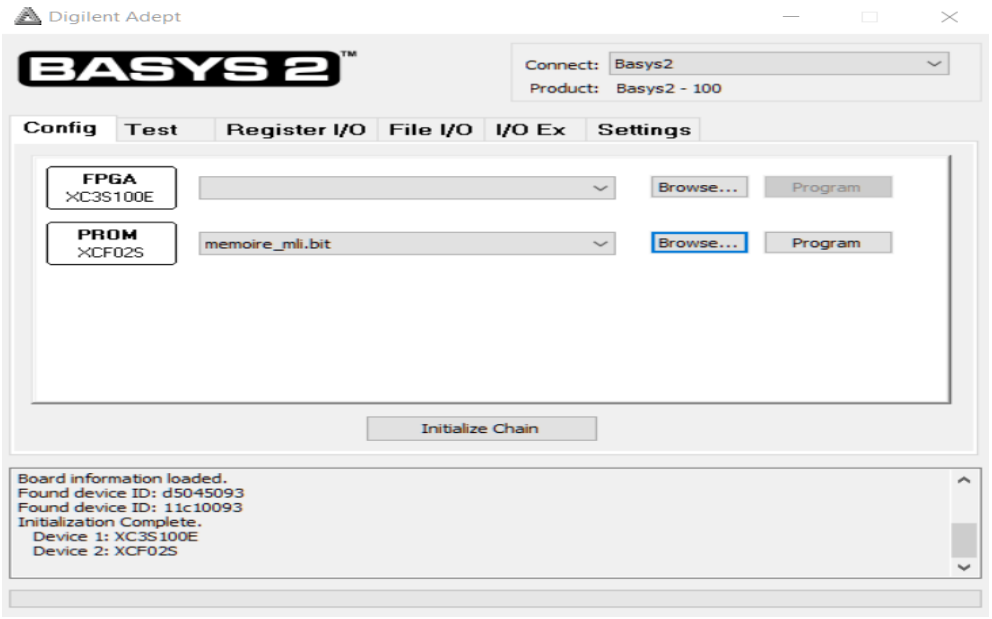


Figure IV.15 : Interface de l'outil Adept.

Le logiciel permet de détecter la carte lors la connexion avec l'ordinateur, et puis programmer la carte avec deux option.

- La premier option est de programmer le FPGA de la carte directement, avec cet option le code s'exécute d'une façon normale, sauf que la carte doit être reprogrammé lors l'absence de l'alimentation (mémoire volatile).
- La deuxième option permet de programmer la mémoire flash de la carte, contrairement à la première option, le programme reste lors l'absence de l'alimentation (mémoire non-volatile).

IV.7. Partie expérimentale

La figure IV.16 montre le montage réalisé afin de visualiser les signaux de sorties issues de la carte Basys2 à base du FPGA – Spartan 3E. Dans ce cas, nous utilisons une carte Basys2, un oscilloscope et un PC portable. Cela afin de visualiser le signal MLI issue de la carte en utilisant l'oscilloscope, le Bitestream a été transférer vers la carte à l'aide de l'outil Adept.

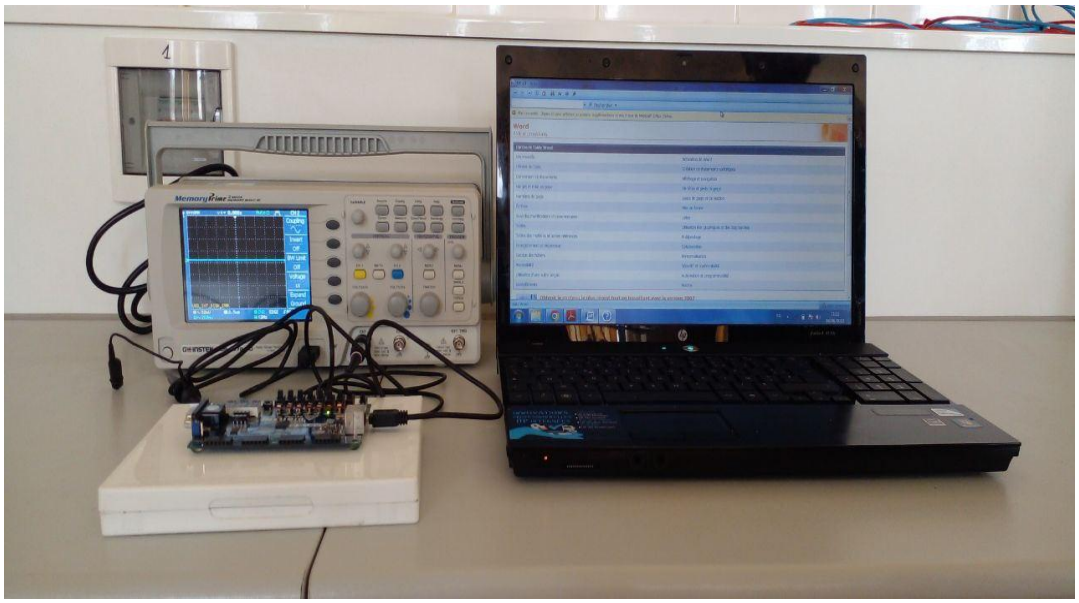


Figure IV.16 : Image du test réalisé.

Les figures IV. 17, 18 et 19 montrent les signaux MLI relevés respectivement, pour DATA = 01000000b, DATA = 10000000b et DATA = 11000000b. On peut nettement voir l'exactitude et la précision des rapports cycliques des signaux obtenus respectivement pour 25%, 50% et 75%, et qui correspondant bien aux résultats de la synthèse et de la simulation.

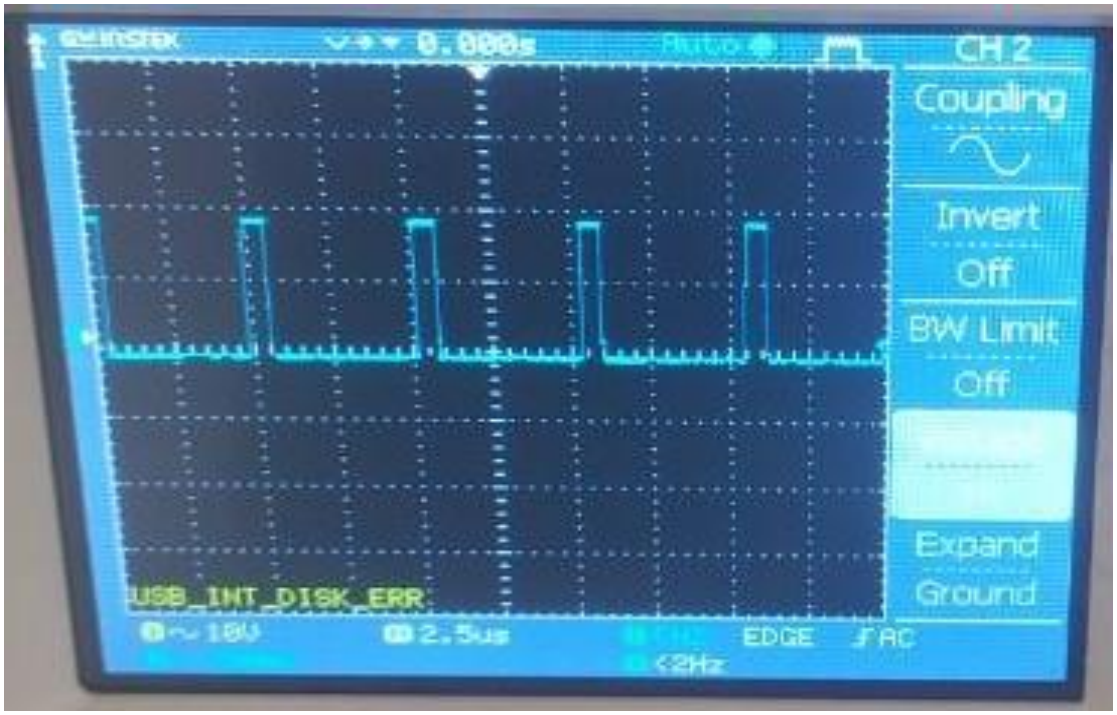


Figure IV.17 : Signal MLI avec un rapport cyclique de 25%.

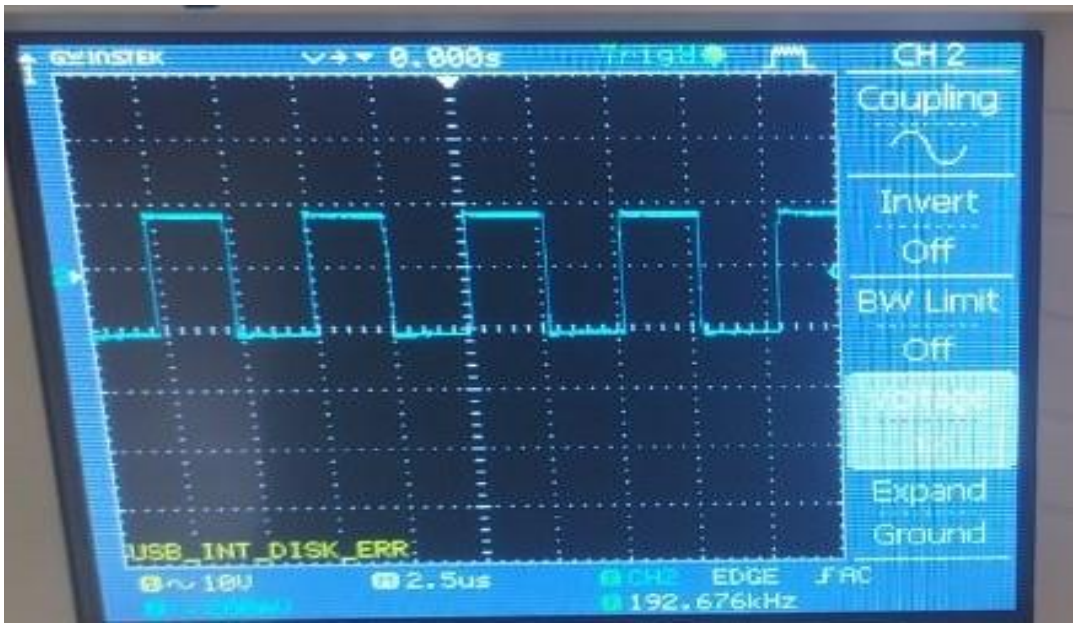


Figure IV.18 : Signal MLI avec un rapport cyclique de 50%.

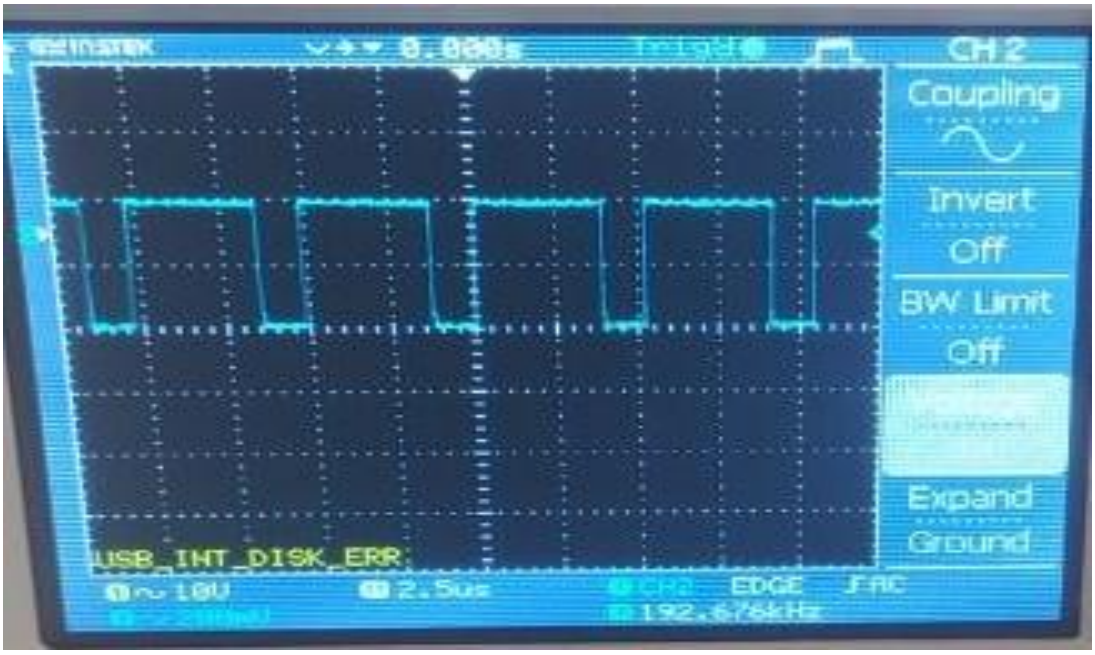


Figure IV.19 : Signal MLI avec un rapport cyclique de 75%.

IV.8. Conclusion

Dans ce chapitre, nous avons implémenté la MLI Standard (impulsion modulé en largueur) sur un FPGA de la famille Spartan3E et via l'utilisation du langage VHDL. Cela, pour la réalisation pratique d'un générateur MLI à base de FPGA.

Dans la première partie du chapitre, nous avons fixé la stratégie permettant d'implémenté la MLI sous VHDL ainsi que d'élaborer un organigramme afin de programmer la technique sous VHDL. En effet, la stratégie employée consiste à considérer un système formé de quatre blocs : un registre, un compteur, un comparateur et une bascule RS, et chaque composant à un rôle bien déterminé. Cela afin de générer des signaux MLI.

Dans la deuxième partie du chapitre, nous avons présenté les résultats de l'implémentation, cela en commençant par les schémas RTL. Et en terminant par la présentation de la variation des signaux obtenus avec deux manières : simulation et expérimentale.

Les résulta obtenus montrent l'exactitude et la précision du générateur élaboré en utilisant la carte Basys 2 à base d'un FPGA de la famille Xilinx « Spartan3E ».

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons réalisé un générateur d'impulsion modulé en largueur (MLI) à travers l'implémentation de la MLI sur une carte FPGA XILINX-SPARTAN-3E. Nous ainsi utilisé le langage de description matériel VHDL à l'aide de l'outil ISE Design de XILINX.

En effet, la technique MLI est largement utilisée dans divers systèmes et applications d'électromécaniques et d'électrotechnique, citant comme exemples les systèmes de communication et les convertisseurs de puissance (DC-DC, DC-AC, etc.). Cette technique permet donc de contrôler la quantité de puissance envoyée à la charge. D'une autre part, le principe de la technique MLI est basé sur une comparaison entre une tension de référence ajustable et un signal triangulaire dont l'amplitude et la fréquence sont constants.

Nous avons, donc, implémenté, pour le MLI, un compteur (08 bits) qui s'incrémente pour chaque Top d'horloge et un registre fixant la valeur du rapport cyclique. Puis, à chaque incrémentation, la valeur du compteur est comparée avec celle du registre. Si le compteur est supérieur au registre, la sortie (via une bascule RS) est au niveau bas. Aussi, lorsque le compteur est remis à zéro, la sortie est au niveau haut. Et dans le cas général, la sortie est au niveau haut.

Les bons résultats obtenus prouvent la fiabilité et l'exactitude du générateur réalisé. Aussi, l'utilisation du couple VHDL/FPGA permet la conception, l'implémentation et la validation des systèmes et des techniques de commande et surtout d'aller vers des fréquences élevées et stables de la tension de sortie. De plus, selon l'approche adoptée nous avons profité au maximum des avantages offerts par le FPGA et cela par rapport à une exécution séquentielle sur un microcontrôleur ou un microprocesseur.

À l'issue de ce travail, nous avons :

- Approfondi nos connaissances sur le langage VHDL.
- Appris une base sur la configuration des périphériques, l'utilisation des FPGAs, conception et commande des machines avec des circuits reconfigurables comme le cas des FPGAs.

À la fin, nous proposons comme perspectives :

- La conception et l'implémentation de la technique MLI aléatoire qui fait l'objet de plusieurs recherches & développements.

Références bibliographies

[1] J. I. Tamboli, S. R. Jagtap, et al. 2012. Pulse Width Modulation Implementation using FPGA and CPLD ICs. *International Journal of Scientific & Engineering Research*, vol. 3, n° 8, p.1-5.

[2] B. Smaani. 2019. Electronique numérique avancée : FPGA et VHDL. Support de cours, université de Boumerdes, Algérie, 70 p.

[3] E. Koutroulis, A. Dollas, et al. 2006. High–frequency pulse width modulation implementation using FPGA and CPLD ICs. *Journal of systems architecture International Journal of Scientific & Engineering Re*search, vol. 52, p.332-334.

[4] M. Lavabre. Electronique de puissance, conversion de l'énergie. Cours et exercices résolus, DUT-BTS, écoles d'ingénieurs. France: Casteila, 2012. 357p.

[5] https://sin.ledantec-numerique.fr/wp-content/uploads/cours_MLI.pdf (date de consultation : 05/05/2022)

[6] https://www.techno-science.net/definition/3195.html (date de consultation : 10/05/2022)

[7] D. Snaoui. 2015. Commande Numérique en Force à Base de la Carte FPGA d'une d'une Architecture de Télé Architecture de Télé-opération opération à un Seul Degré de Liberté. Mémoire de Master, Université Mouloud Mammeri de Tizi-Ouzou, Algérie.

[8] https://www.techno-science.net/definition/3195.html (date de consultation : 10/05/2022)

[9] https://www.interface-z.com/conseil/pwm.php (date de consultation : 15/05/2022)

[10] https://www.hydroquebec.com/data/affaires/pdf/variateurs-vitesse.pdf (date de consultation : 21/05/2022)

[11] https://lifesaver.fandom.com/fr/wiki/Modulation_de_Largeur_d%27Impulsion (date de consultation : 25/05/2022)

[12] A. Saadi. 2012. Commande d'un onduleur triphasé via un circuit FPGA. Mémoire de Master, Université Constantine 1, Algérie.

[13] https://www.univ-chlef.dz/ft/wp-content/uploads/2020/04/Circuits-programmables.pdf (date de consultation : 01/06/2022)

[14] J.M. Retif, B. Allard, et al. 1996. Use of ASIC's in PWM techniques for power converters. Proceedings of the 22nd International Conference on Industrial Electronics, Control, and Instrumentation, Taipei, Taiwan, 09 août 1996, p.138148.

[15] P. J. Ashendenal. The VHDL cookbook. First Edition. Australia: Department of Computer Science, University of Adelaide, 1990.

[16] D. L. Perry. VHDL: Programming by Example. Fourth Edition. Chicago; San Francisco : McGraw-Hill, 2002. 497p.

[17] A. Houari. 2016. Méthodologie de développement et d'implantation sur puce FPGA d'algorithme de commande. Mémoire de Magister, Université Mohamed Khider Biskra - Algerie

[18] A. Houari. 2016. Méthodologie de développement et d'implantation sur puce FPGA d'algorithme de commande. Mémoire de Magister, Université Mohamed Khider Biskra - Algerie

[19] Digilent Basys2 Board Reference Manual, Digilent, 2010.

[20] I. Mezzah. 2008. Etude et conception d'un microcontrôleur IP sur FPGA. Mémoire de Magister, Université Ferhat Abbas, Algérie.

Résumé

Le but de ce projet est de réaliser un générateur d'impulsion modulé en largeur MLI en utilisent la carte de développement Basys2. Nous avons implémenté la MLI standard sur le FPGA Spartan-3E à l'aide du langage VHDL et l'outil ISE design de Xilinx. Nous avons ainsi implémenté un compteur qui s'incrémente pour chaque « Top » d'horloge (CLK) et un registre fixant la valeur du rapport cyclique. Puis à chaque incrémentation, la valeur du compteur est comparée à celle du registre. Si la valeur du compteur est supérieure à celle du registre, la sortie est au niveau bas, sinon si le compteur est remis à zéro, la sortie est au niveau haut.

Mots clés : MLI standard, générateur, VHDL, FPGA, Spartan-3E.

ملخص

الهدف من هذا المشروع هو تجسيد مولد النبضات بعرض معدّل MLI باستخدام لوحة تطوير Basys2. قمنا بتنفيذ تقنية MLI بواسطة Spartan-3E FPGA باستخدام لغة برمجة VHDL وأداة تصميم Xilinx ISE. لذلك قمنا بتنفيذ عداد يتزايد لكل نبضة على مدار الساعة CLK وسجل يحدد قيمة دورة العمل. ثم عند كل زيادة, تتم مقارنة قيمة العداد مع قيمة السجل. إذا كانت قيمة العداد أكبر من قيمة التسجيل، يكون الناتج منخفضًا، وإلا إذا تم إعادة تعيين العداد، يكون الناتج مرتفعًا.

***الكلمات المفتاحية**: تقنية MLI، مولد، VHDL ، FPGA، Spartan-3E.*

Abstract

The aim of this project is to realize a pulse width modulation (PWM) generator using the Basys2 development board. We have implemented the standard PWM on the Spartan-3E FPGA using the VHDL language and the Xilinx ISE design tool. Hence, we have implemented a counter that increments for each clock pulse and a register setting the value of the duty cycle. Then at each CLK increment; the value of the counter is compared to that of the register. If the counter value is greater than the register value, the output is low, otherwise if the counter is reset, the output is high.

Keywords: Standard PWM, generator, VHDL, FPGA, Spartan-3E.