SOMMAIRE

REMERCIMENTS

Avec le plus grand honneur que nous réservons cette page de gratitude et de reconnaissance à tout ceux qui ont contribué de prés ou de loin le bon déroulement et à la réalisation de notre projet.

Nous tenons à remercier sincèrement notre encadreur :

Mr HATEM ABIDI pour ses aides incessantes, de ses conseils, de ses connaissances intéressantes et pour tous les éclaircissements qu'il nous a fournis et disponibilité tout au long de notre travail.

Nous voudrons, à cette occasion, exprimer notre profonde reconnaissance à tout nos enseignant de département Génie électrique pour leur collaboration, leur disponibilité et leur soutien moral au long de notre étude.

Enfin, nous adressons nos vifs remerciements du jury qui accepté volontiers d'évaluer notre modeste mémoire.

DEDICACE

La vie n'est qu'un éclair,

Et le jour de réussite est un jour très cher

A ma mère

A mon père

Aucun hommage ne peut être à la hauteur de l'amour

Et de l'affectation dont ils ne cessent de nous combler

Qu'ils trouvent dans ce travail un modeste témoignage

De mon profonde amour

Que bien leurs procure bonne santé et longue vie

A mes chères frères

A mes chères sœurs

A tous mes amis

MESSAOUD

DEDICACE

En témoignage de respect et d'amour Je dédis ce travail, en espérant qu'il sera à la hauteur de leur attente :

A ma chère mère

La personne qu'elle ma encourage de continuer quand les choses vont mal.

A mon père

La personne qu'il s'est fatigué pour que j'atteindre ces points.

A mon frère

A ma sœur

Et à tous mes amis.

RIDHA

INTRODUTION

L'évolution de l'industrie des circuits intégrés durant la dernière décennie a été tellement rapide qu'il est maintenant possible d'intégrer plusieurs systèmes complexes sur une seule puce. Cette évolution vers des niveaux d'intégration de plus en plus élevés est motivée par les besoins de systèmes plus performants, légers, compacts et consommant un minimum de puissance. Dans de telles circonstances, la gestion de la complexité avec les outils d'aide à la conception traditionnels devient une tache pénible, coûteuse voire impossible, quand on considère l les contraintes de mise en marché d'un produit

Ce rapport présente notre projet qui vient d'être réalisé et qui consiste à commander un moteur pas à pas dans deux sens et un moteur à courant continue par la modulation de largeur d'impulsions en utilisant un circuit programmable FPGA

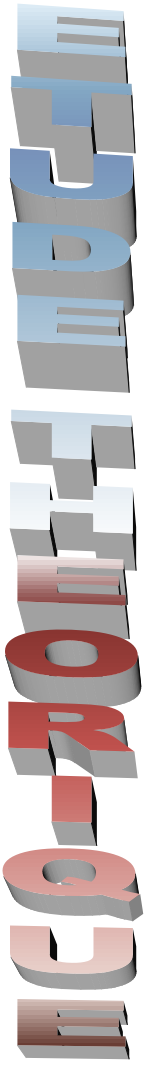
Dans ce manuel nous avons effectué l'étude théorique du moteur pas à pas aussi la modulation de largeur d'impulsion qui permet de varier la vitesse d'un moteur à courant continue

Nous avons consacré quatre thèmes à savoir

* / le circuit FPGA
* / le moteur pas à pas
* / modulation largeur d'impulsion
* / moteur a courant continue

Entre autre, ce rapport traitera, la partie commande de deux moteurs, la présentation sera détailler pour les circuit intégrés spécialiser, la carte de commande et de puissance

Enfin, nous citerons le programme VHDL qui permet de générer les signaux permettant de commander le moteur pas à pas avec ses option de sens de rotation , aussi de faire tourner le moteur à courant continue avec variation de vitesse et affichage de rapport cyclique


ETUDE THEORIQUE

CH I : Circuits logiques programmables FPGA

Les FPGA (abréviation anglaise de qui signifie réseau des portes programmables sur site) sont aussi des circuits logiques programmables par l'utilisateur. C'est une évolution des CLPD mais avec un concept différent, ils offrent un niveau d'intégration très élevé par rapport aux CPLD.

1) Structure des FPGA :

Un circuit FPGA contient un très grand nombre de macro cellules (environ 32*32 CLB) avec une très grande souplesse d'interconnexion entre eux. Dans le FPGA, le temps de propagation dans les couches logiques du circuit dépend de l'organisation et de la distance entre les macro cellules interconnectées.

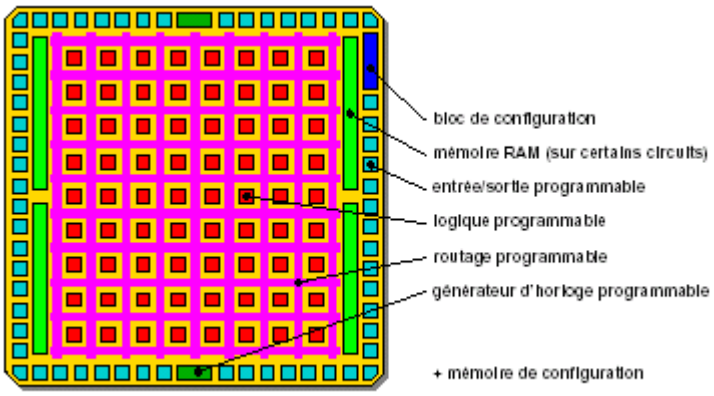


Fig 1 : structure des FPGA

2) Les ressources de FPGA :

Entrées-sorties : elles sont indépendantes des macro cellules et de groupes de macro cellules. Elles ont comme élément de base la porte 3-états. Elles peuvent intégrer beaucoup des éléments comme le choix du temps de montée, l'incorporation d'une résistance de rappel à la tension d'alimentation pour fixer le niveau de tension d'une broche, le choix de type de sortie : soit collecteur ouvert, soit 3-états.

Macro cellules : elles sont très nombreuses, et il n'y a pas de matrice ET et OU. La macro cellule est constituée d'une partie combinatoire et d'une partie séquentielle.

*La partie combinatoire permet de réaliser des fonctions de complexité moyenne. Les constructeurs ont proposé chacun une ou plusieurs solutions de synthèse dont les principales sont :

- La fonction de synthés à 4 ou 5 variables avec des portes classiques ET, OU et NON.
- La synthèse de fonctions à l'aide de multiplexeurs.
- La synthèse de fonctions combinatoires à l'aide de mémoire vive, on dit aussi réalisation des fonctions logiques par LUT, abréviation anglaise de « *LOOK-UP TABLE* » signifiant table d'observation (ou de réalisation).

*La partie séquentielle comporte une ou deux bascules généralement de type D. il est rare de trouver des macro cellules uniquement pourvue de la partie combinatoire. Compte tenu de nombre de macro cellules et de leur structure, leur association permet la réalisation de n'importe quel autre type de bascule.
Les macro cellules sont appelées :

Soit **CLB** : abréviation anglaise de « *Configurable Logic Block* », signifiant bloc Logique configurable. C'est la dénomination adoptée par Xilinx.
Soit **LC** : abréviation anglaise de « *Logic Cell* », signifiant cellule logique, c'est le nom choisi par Cyprès.
Soit **LE** : abréviation anglaise de « *Logic Element, signifiant* » élément logique, c'est l'appellation d'Altéra.
La dénomination la plus courante pour désigner une macro cellule est CLB. A cet effet, on appelle souvent les FPGA, **LCA**, abréviation anglaise de « *Logic Cell Array* » signifiant réseau de cellules logiques. Cette dénomination des FPGA est dépassée comme marque Xilinx.

Réseaux d'interconnexion : il doit permettre de connecter n'importe quelle CLB à une autre ou à une cellule d'entrée-sortie. Pour cela, un ensemble des lignes horizontales et verticales et un ensemble de points de connexions sont utilisés. Selon l'intégration véhiculée, on distingue plusieurs types de lignes définies par leur longueur relatives. Les principales sont :

- Les interconnexions à usage général : sont composées des segments

verticaux et horizontaux qui encadrent chaque CLB et qui peuvent être reliés entre eux par une matrice de commutation. Chaque segment peut être connecté à des segments qui lui sont adjacents ou perpendiculaires en utilisant des points de connexion

.

- Les lignes directes : fournissent des chemins entre les CLB adjacents et entre les CLB et les cellules d'entrée-sortie.

- Les lignes longues : sont des lignes verticales et horizontales qui n'utilisent pas les matrices de commutation. Elles parcourent toutes les zones d'interconnexion. Elles sont utilisées pour véhiculées les signaux qui doivent parcourir un long trajet. Elles égalisent les détails entre les signaux de façon à permettre un décalage minimum entre deux points distants de la ligne.
  Les lignes conviennent pour véhiculer les signaux d'horloge.

L'ensemble des points de connexion est appelé **PIP**, abréviation anglaise de « *Programme Interconnect Points* ».

3) Le Circuit EPF8282 d'Altéra :

Ce circuit contient :

208 Cellules logiques (LE) ;
78 Broches d'entrée-sortie ;
282 Bascules actives sur front ;
Des connexions commandées par des cellules de type SRAM ;
Le circuit EPF8282 fait partie de la famille **FLEX**800 des FPGA d'Altéra. Fl ex est l'abréviation anglaise de *« Flexible Logic Element matriX* » signifient matrice d'éléments flexibles. Pour rendre les circuits de la famille FLEX adaptable à une large gamme d'applications, cette famille combine (pour les CPLD, les avantages de grande vitesse de fonctionnement et de temps de propagation prédictible) et pour les FPGA le nombre important de registres et la souplesse de connexion.

Ce circuit contient 208 cellules logiques (LE) organisées en 26 LAB de 8 LE, les LAB sont organisées en deux lignes de 13 colonnes. Comme il a été indiqué dans la présentation générale, il n'y a pas de matrice ET et OU.

Chaque LE comporte une partie combinatoire et une partie séquentielle.
La fonction logique de 4 variables, un bloc pour le calcul anticipé de la retenue pour les opérations arithmétiques et un bloc pour la mise en cascade rapide les fonctions de décalage.

La partie séquentielle est constituée d'un registre qui peut être configuré en bascule D, T, JK ou RS.

Ce circuit dispose de 78 blocs d'entrée-sortie qui sont placées aux extrémités (colonnes et lignes) du réseau d'interconnexion.

La connexion entre les LAB d'une part et les LE et les blocs d'entrée-sortie, d'autre part est légèrement différente de celle que l'on a présenté pour les FPGA. Par contre la commande des points de connexion se fait toujours par superposition d'une couche de commande constituée de (connexion) cellules mémoire de type SRAM.

La notion de différents types de ligne est remplacée par la notion unique de groupes de lignes et colonnes qui traversent le circuit de part en part. cette notion est appelée en anglais « *Altéra Fast Track Interconnect* » signifiant piste rapide d'interconnexion. La connexion entre les lignes et colonnes se fait par la commande de multiplexeur. Cette commande provient des cellules mémoire (SRAM) de la couche de commande. Cette technique évite les délais introduits par les matrices de commutation entre les lignes d'usage général de la structure standard. Un groupe de lignes comporte 168 lignes et celui des colonnes est constitué de 16 colonnes.

Comme pour tout FPGA, la gestion de ces lignes et colonnes est complètement transparente. Cependant, le système de développement offre la possibilité d'imposer des contraintes de placement et de routage.

4) Outil de développement :

Comporte en général, un logiciel de conception, une carte d'émulation pour la vérification du fonctionnement électrique de l'architecture conçue, un micro-ordinateur hôte.

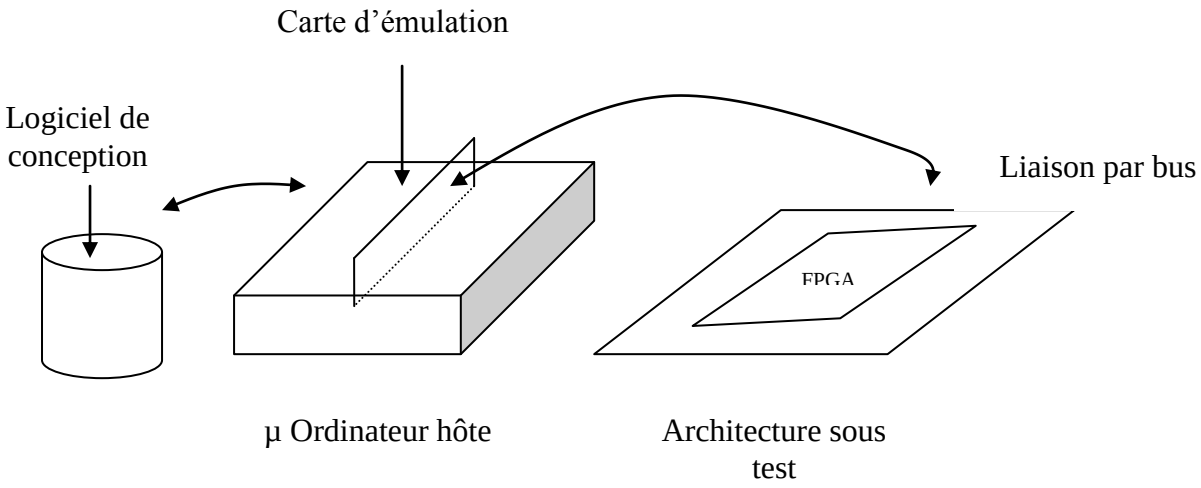


Fig2 : outil de développement

Le logiciel : sert à aider le concepteur à développer la méthode de conception de l'architecture.

La méthode de conception comporte quatre phases : une phase de saisie de l'architecture, une phase d'implantation, une phase de vérification et une phase des tests électriques.

La phase de saisie de l'architecture consiste à entrer l'application sous la forme de son schéma ou de ses équations booléennes de fonctionnement de son graphe d'étalon, enfin, sous forme d'un langage de description. Ce dernier semble s'imposer sous la forme du langage de **VHDL** (*Very High Development Library*), pour saisir l'architecture encore appelée « application ».

Pour les outils de saisie de schémas, d'équations, de graphes d'états, on peut citer des logiciels tels que view logic, Mentor Graphiques, Orad, ABEL…… chacun de ces logiciels à de plus en plus tendance à générer automatiquement le code VHDL.

*Le Synoptique : qui décrit la méthode de conception d'une architecture, il comprend trois étapes /

une étape indépendante de la technologie du circuit cible, contient la saisie de l'application, la génération des commandes fictives du réseau de blocs logiquespar programmation, encore appelé « synthèse », la simulation du circuit synthétisé ;

une étape qui dépend de la technologie et qui se développe après avoir choisi le constructeur de FPGA, contient la programmation fictive du câblage du circuit du constructeur choisi, encore appelé « routage », et la simulation de son fonctionnement réel.
Jusqu'à la fin de cette étape, la conception a la possibilité de modifier toute partie de l'application à tous les niveaux de description facilement et de façon économique ;
une étape de programmation physique et de tests électriques termine la réalisation du circuit prototype

6) Synoptique de la méthode de conception d'un circuit numérique FPGA

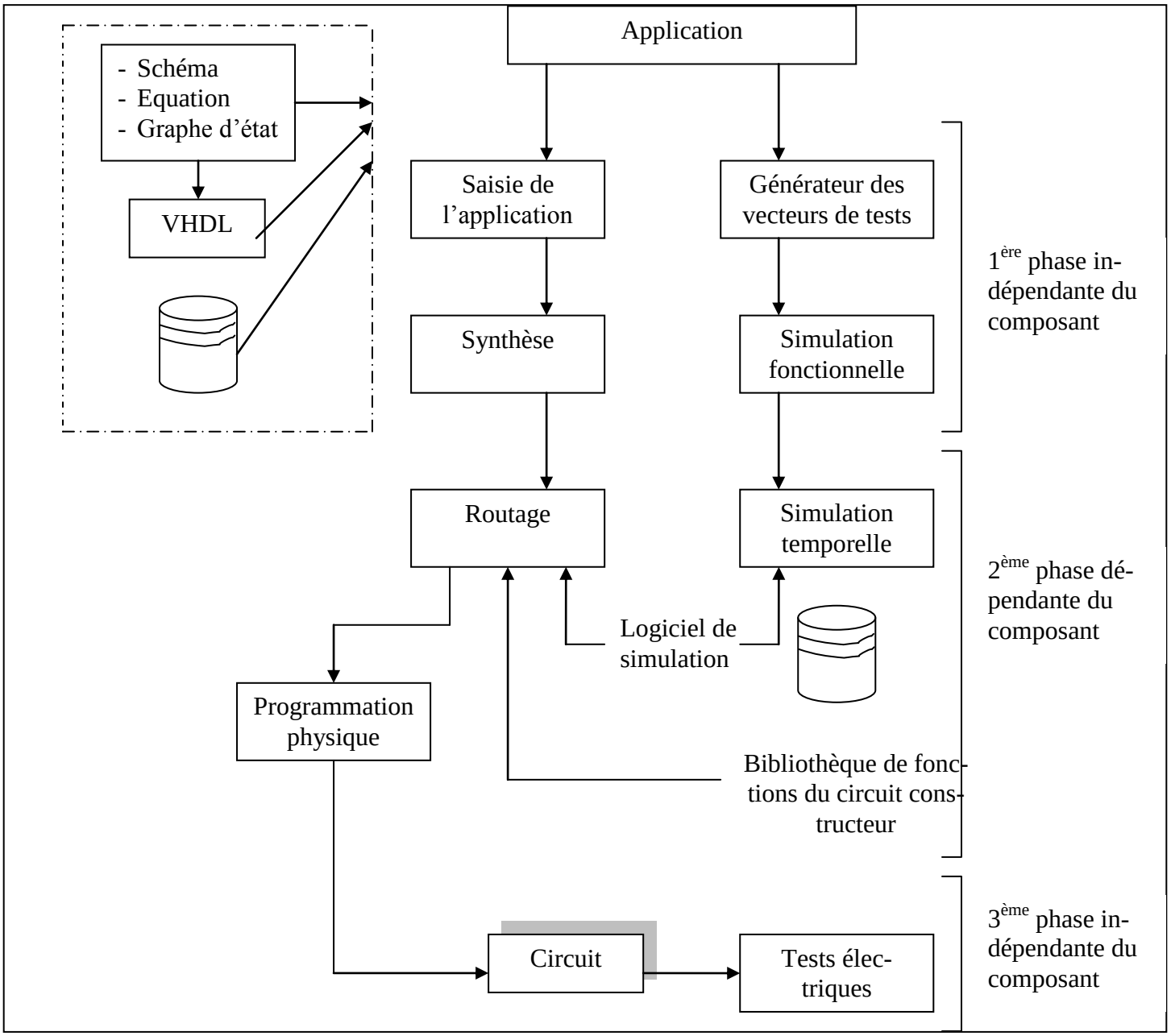


Fig3

DESCRIPTION DU STARTER KIT

1) INTRODUCTION

Ce starter Kit peut utiliser le logiciel du CPLD / FPGA pour étudier et concevoir les nouveaux circuits logiques et remédier à la complexité de conception des circuits TTL/CMOS aussi utilisé le graphisme et VHDL-ABEL-AHDL pour la conception des circuits comme on peut utilisé le port parallèle pour télécharger directement des applications vers la cible.

Le programme élaboré est chargé en FLASH EPROM et la carte devient autonome opérationnelle sous WIN95 /98/2000/NT.

2) SPECIFICATION DE LA CARTE EXPERIMENTALE FPGA

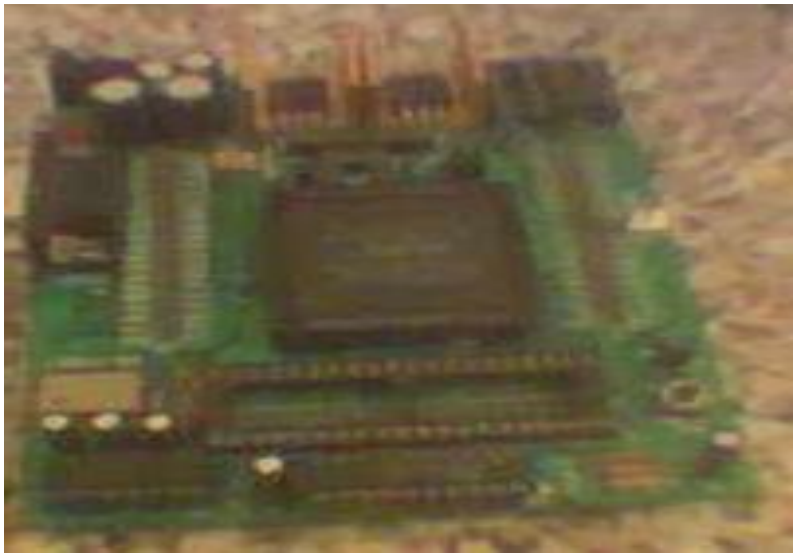


Fig4

4) Dispositif de la famille FLEX 8000 :

La famille de FLEX 8000 offre des dispositifs aux limites de 2,500 à 16,000 portes avec rapidité prévisible à la connections des retards. Elle combine la haute vitesse, Le chronométrage prévisible et la facilité d'utilisation d'EPLDS avec le haut compte de registre, bas le pouvoir (puissance) de réserver et 'dans le circuit' de reconfiguration de FPGAS.

L'architecture de FLEX 8000 consiste en élément de logique de silicium efficace au grain fin groupé dans des laboratoires très performants, à gros grain. Cette granularité duelle permet à la famille de FLEX 8000 d'offrir, et exécutée (performance) rapide d'EPLDS et la haut utilisation de ressource de tableaux de porte.

Avec la famille de FLEX 8000, nous pouvons facilement mettre en œuvre les demandes (application) qui exigent un haut flip-flop des comptes comme des chemins de données pipeline et des algorithmes de transformation / compression de données sans sacrifier des taux d'horloge de système rapide les dispositifs de FLEX 8000 sont idéal pour des demandes données intensives comme des contrôle graphique. Des cartes multimédia, des cartes de réseau et l'équipement de communication

Les circuits FPGA sont constitués d'une matrice de blocs logiques programmables entourés de blocs d'entrée sortie programmable. L'ensemble est relié par un réseau d'interconnexions programmable.
Les FPGA sont bien distincts des autres familles de circuits programmables tout en offrant le plus haut niveau d'intégration logique.

Il y a 4 principales catégories disponibles commercialement:

- Tableau symétrique.
- En colonne.
- Mers de portes.
- Les PLD hiérarchique.

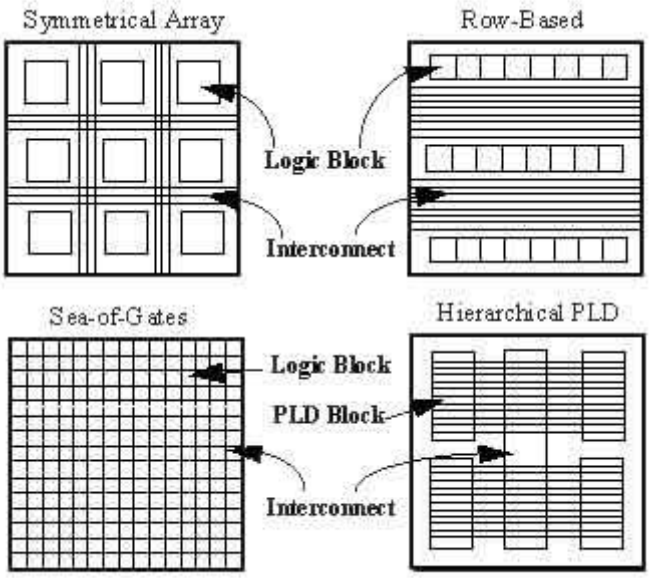


Fig5 : Les différentes classes de FPGA.

Voici la structure interne d'un FPGA de type matrice symétrique. Il s'agit de l'architecture que l'on retrouve dans les FPGA de la série XC4000 de chez Xilinx.

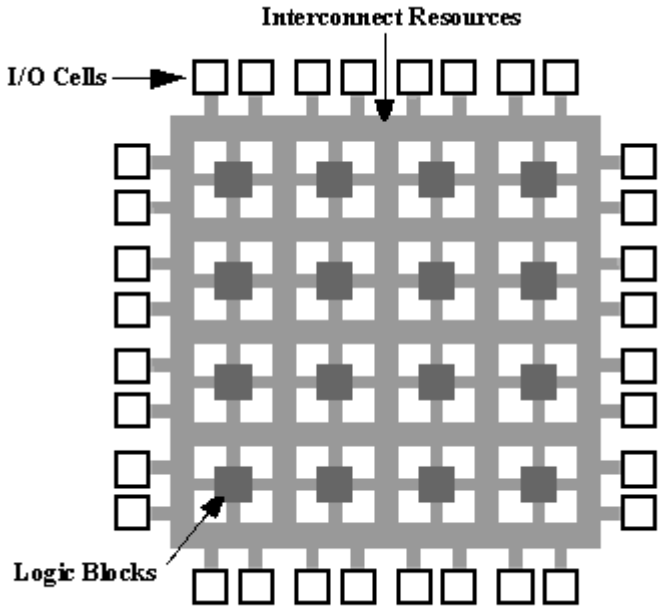


Fig6 : Structure interne d'un FPGA

L'utilisateur peut programmer la fonction réalisée par chaque cellule (appelée CLB par Xilinx: Configurable Logic Block):

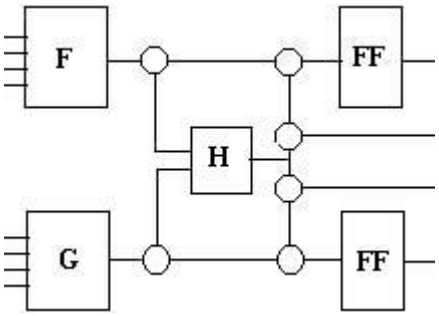


Fig7 : Schéma bloc d'une cellule

On programme aussi les interconnexions entre les cellules:

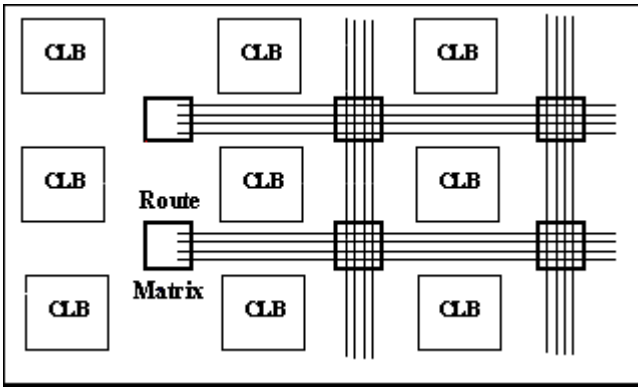


Fig8 : Les interconnexions entre les cellules d'un FPGA

Ainsi que les entrées et sorties du circuit. L'avantage des FPGA est de pouvoir être configuré sur place, sans envoi du circuit chez le fabricant, ce qui permet de les utiliser quelques minutes après leurs conceptions. Les FPGA les plus récents sont configurables en une centaine de millisecondes. Les FPGA sont utilisés pour un développement rapide et bon marché des ASIC .

Inventés par la société Xilinx, le FPGA, dans la famille des ASICs**,** se situe entre les réseaux logiques programmables et les prédiffusés**.** C'est donc un composant standard combinant la densité et les performances d'un prédiffusé avec la souplesse due à la reprogrammation des PLD. Cette configuration évite le passage chez le fondeur et tous les inconvénients qui en découlent.

LE LANGAGE DE PROGRAMMATION VHDL

Introduction

Le langage VHDL (Vhsic Hardwere Description Langage) est un sous produit né du besoin d'une outil de spécification a pour but de décrire un fonctionnement global de haut niveau sans souci des détails d'implantation, pour servir de cahier des charges pour la réalisation des projets complexes .Il est devenu un véritable outil de conception des circuits numériques .il permet d'autre part de concevoir et de vérifier un système électronique complexe. Cet ouvrage expose en trois parties qui conduisent de la simulation au circuit opérationnel. La première partie aborde les différentes applications qui permet de réaliser le langage VHDL (la spécification, la simulation, la synthèse logique). La deuxième développe l'utilité du langage VHDL, tant en synthèse qu'en vérification. De nombreux exemples permettent au lecteur de valider sa compréhension. La dernier partie est une présentation syntactique

2) Applications du langage :

Le VHDL est un langage unique permettant de faire :

De la spécification : le langage VHDL est très bien adapté à la modélisation de systèmes numérique complexes grâce à son niveau élevé d' abstraction .le partitionnement en plusieurs sous –ensemble permet de sub-diviser un modèle prêts à être développés séparément

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

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

De la preuve formelle : le langage permet de prouver formellement que deux descriptions sont parfaitement identiques au niveau de leur fonctionnalité.

3) Utilité du langage VHDL :

L'électronicien a toujours utilisé des outils de description pour représenter des structures logiques ou analogique. Le schéma structurel que l'on utilise depuis longtemps n'est en fait qu'un outil de description graphique.

Aujourd'hui, l'électronique numérique est de plus en plus présente et tend bien souvent à remplacer les structures analogiques utilisées jusqu'à présent. Ainsi,les fonctions numériques à réaliser ne cessent de prendre de l'ampleur. Il est dès lors indispensable d'utiliser un langage de haut niveau pour maîtriser la complexité grandissante des systèmes numériques. Le VHDL est l'un des langages modernes et puissants .IL est nettement plus performant que la description par schéma.

Le deuxième point fort du VHDL est d'être un langage de description comportementable de haut niveau. Les anciens langages, tel ABEL, ne disposait pas de celle fonctionnalités. En fait, un langage est dit de haut niveau lorsqu 'il fait le plus possible abstraction de l'objet pour lequel il est écrit. Dans le cas de VHDL, il n'est jamais fait référence au composant ou à la structure pour lesquels on l'utilise. Ainsi,il apparaît une notion très importante : la portabilité des descriptions VHDL.

Il est également important de noter, que le langage VHDL est normalisé. Il n'a pas la propriété d'une valeur d'outils. Cette norme a posé de grandes sociétés de logiciels à l'utiliser comme langage de déception de matériel pour leur outil. Le langage est ainsi devenu un standard reconnu par une majorité des valeurs d'outils de système.

Le langage VHDL a été initialement utilisé pour la spécification et la simulation de systèmes complexes. Ce langage de haut niveau permet un dé composition hiérarchique et dispose de la notion de temps. Cela a permis de maîtriser la complexité des circuits, particulièrement dans le cas des ASICs. Le même langage est utilisé pour toutes les étapes du développement (spécification, simulation et synthèse). Il permet également la réalisation de bibliothèques composantes réutilisable d'un à l'autre.

4) la portabilité des descriptions VHDL :

Les concepteurs souhaitent être indépendants des outils qu'ils utilisent . Ils demandent que leurs descriptions soient portables.

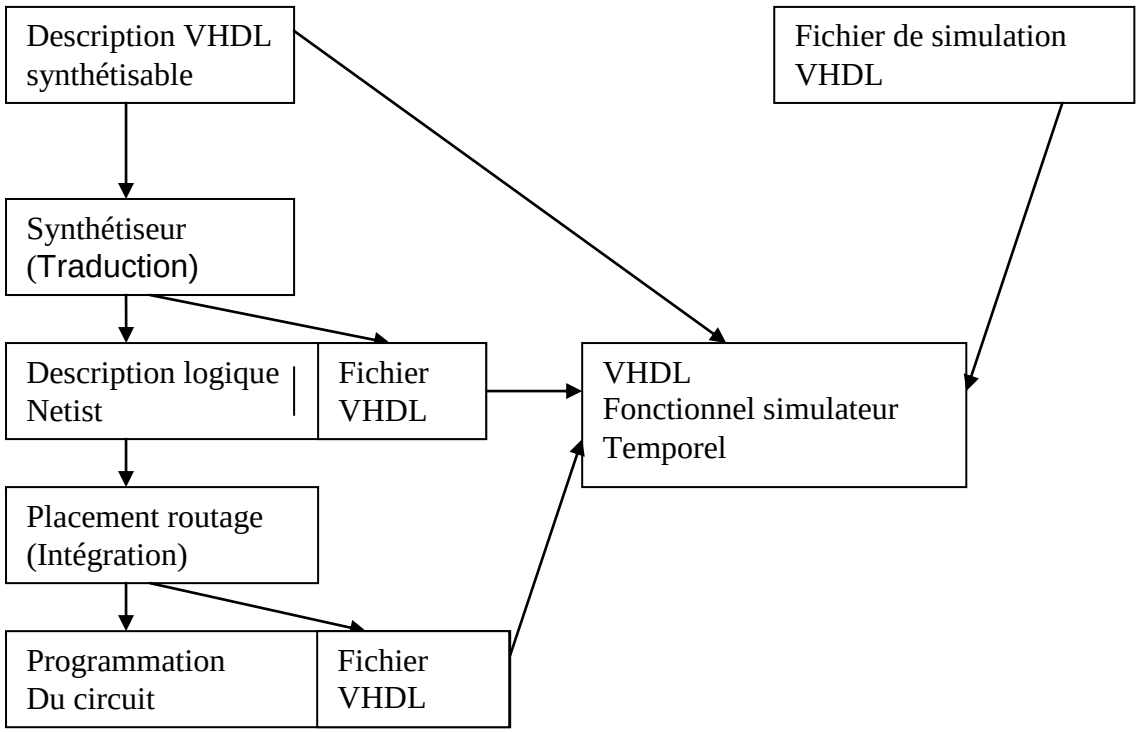
Dans le mande des systèmes numériques nous pouvons définir deux portabilités :

La portabilité vis-à-vis des logiciels.
La portabilité vis-à-vis des circuits.
Le langage VHDL est devenu un standard dans le nombre de la conception numérique. Il existe de nombreux outils qui acceptent ce langage comme entrée.Il est donc possible d'utiliser une même description avec plusieurs outils différents. Mais il faut encore garantir que chaque outil aura la même interprétation de la description VHDL. Dés lors, il est important de ne pas utilisée les spécificités d'un outil . En particulier il ne faut pas utiliser les bibliothèques propres à un outil mais uniquement les bibliothèques normalisées IEEE (std_logic_1164 et numeric_std).

Pour les circuits, la portabilité ne pourra plus être assurée dès que nous utilisons une ressource propre à une technologie pour respecter des contraintes de vitesse ou optimiser la surface.

5) simulation et synthèse :

Voici le déroulement des différentes étapes de développement d'un projet en VHDL.



Le développement en VHDL nécessite l'utilisation de deux outils : le simulateur et le synthétiseur.le premier va nous permettre de simuler notre description VHDL avec un fichier de simulation appelé (test-bench) .Cet outil interprète directement le langage VHDL .Le simulateur comprend l'ensemble du langage.L'objectif du synthétiseur est très différent. Il doit traduire le comportement décrit en VHDL en fonction logique de bases. Celles-ci dépendante la technologie choisie .Cette étape est nommée : 'synthèse'. Le langage VHDL permet d'écrire des descriptions d'un niveau comportemental élevé. La question est de savoir si n'importe quelle description comportementale peut être traduite en logique

L'intégration finale dans le circuit cible est réalisée par l'outil de placement et routage.

Celui-ci est fourni par le fabricant de la technologie choisie.

Avec les outils actuels, il est possible de disposer de fichiers VHDL à chaque étape. Le même fichier de simulation (test_benh) est ainsi utilisable pour vérifier le fonctionnement de la description à chaque étape.

6) Syntaxe générale :

La difficulté du langage VHDL est dans le syntaxe qui se résulte comme suit :

Library IEEE ;

Use ieee.std_logic _1164.all ;

Entity (nom du projet) is

Port

((Définition des variables)) ;

End (nom du projet) ;

Architecture arch of (nom du projet)

is

Begin

(sortie d'instruction)

End arch ;

7.1) Library:

La déclaration librairies commence tout code VHDL (sauf que n'utiliser le noyau du langage qui n'est pas suffisant pour les besoins de l'électronique digitale).

Le choix des librairies a un impact important sur la portabilité et les outils utilisables (synthèse et simulation).

Exp :

Library IEE ;

Use IEEE.STD_Logic_1164.all ;

Use IEEE.STD_logic_ARITH.all;

Use IEEE.STD_logic_signed.all;

7.2) Couple Entity -Architecture:

Un module est décrit en deux parties :

Interconnections : section entity

Contenu interne : section architecture.

Une entité et plusieurs architectures => le choix est par la section configuration.

7.3) Définition des variable :

Dans ce stade le programmeur doit déclarer les différents variables d'entre et de sortie ainsi les variables intermédiaires de son projet

Variables d'entrées : nom du variable (in put)

Variables de sorties : nom du variable (out put)

variables intermediares : dans certains cas on a besoin de variable intermédiaire pour stocker les résultas.Ces variables sont déclarés après les variables d'entrée\ sorties.

7.4) Vecteur d'E\S bus :

Dans les systèmes modernes toutes les connections sont réalisés a travers des regroupements de ligne appelée bus. Le VHDL permet de décrire des systèmes complexes gérant des vecteurs

Un bus est déclaré comme suit :

Nom_bus :std_LOGIC_VECTOR(width_Data_N-1 downto 0)

8)Les fonctions du VHDL :

8.1) La fonction case :

Cette fonction est édentique a la « case » du C. Elle permet de décrire la réponse de la sortie pour différent vecteur d'entrée.

Syntaxe :

Begin

Process « entrée »

Case « entrée »

When « entrée1 » => sortie <= « sortie1 »

When « entrée2 » => sortie <= « sortie2 »

End case

End process

End

8.2) La fonction Case 'When other' :

La fonction case est utilisée lorsqu'on décri une fonction littéralement, dans certain cas le système ne doit pas répondre a toute les combinaison d'entrée: on utilise la fonction « other » qui permet ainsi de définir la sortie du système a une entrée quelconque autre que les entrées spécifiés par la description

Syntaxe :

Begin

Process « entrées »

Case « entrée » is
When « entrée 1» => sortie <= «sortie 1»
When « entrée 2» => sortie <= «sortie 2»
When others '' enrée ''
End case
End process
End

8.3) Fonction IF:

La déclaration if permet de déclarer et d'exécuter le choix selon une ou plusieurs conditions. La syntaxe est :

Begin

Process « entrée »

Begin

If entrée1= condition then
Sortie <= condition1
Else
Sortie <= condition2
End if
End process End;

CH II : LE MOTEUR PAS A PAS

A- INTRODUCTION

A-1- Présentation

Les premiers moteurs pas à pas datent de 1930, leur véritable développement est lié à l'événement de la micro-informatique (microprocesseur). Le moteur pas à pas peut remplir deux fonctions :

Conversion d'énergie électrique en énergie mécanique (c'est le moteur classique)

Conversion de l'information numérique en un positionnement angulaire ou linéaire.

Le caractère synchrone du moteur pas à pas permet de faire fonctionner sans boucle de retour.

A-2-Généralités

Les moteurs pas à pas sont des moteurs spéciaux, composés simplement d'un stator réunissent des pièces polaires et des bobinages, et utilisés pour commander avec grande précision le déplacement et la position d'un objet.

Comme leur nom l'indique, ces moteurs tournent par incrément discret. Chaque incrément de rotation est provoqué par une impulsion de courant fournie à l'un des enroulements du stator

Le moteur pas à pas est l'organe de positionnement et de vitesse travaillant généralement en boucle ouverte.

Le principe de base est donc toujours la création d'un champ tournant comme dans les moteurs triphasés industriels ou dans les petits moteurs équipant les programmateurs mécaniques : les pôles magnétiques de rotation de même nom se repoussent et les pôles des noms contraires s'attirent, le champ magnétique entraînera le rotor alimenté dans le même sens.

Ceci traduit le fait qu'on transforme une grandeur numérique en une grandeur analogique .La fréquence de rotation, ou vitesse est donc commandée par des impulsions (consigne de rotation) contrôlées elle-même par un dispositif électronique en technologie câblée programmée.

B – Les différents types des moteurs pas a pas :

Il y a trois types :

Les moteurs à aimant permanent

Les moteurs à réluctance variable

Les moteurs hybrides

1) Moteur à aimant permanant :

Un aiment permanent est solidaire de l'axe du moteur (rotor) .Des bobines excitatrices sont placées sur les paroi du moteur (stator) et sont alimentées chronologiquement. Le rotor s'oriente suivant le champ magnétique crée par les bobines.

Les moteurs à aimants permanents se subdivisent en deux types principaux :

Les moteurs bipolaires

Les moteurs unipolaires

Le moteur bipolaire

Les bobines d'un moteur bipolaire sont alimentées une fois dans un sens, une fois dans l'autre sens.Ils créent une fois un pole nord, une fois un pole sud d'où le nom de bipolaire

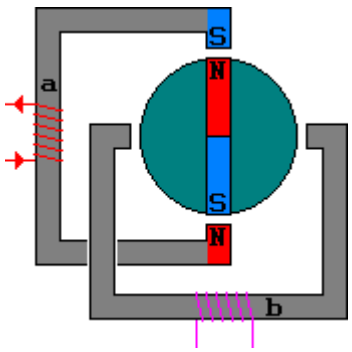


Fig 1 : moteur à aiment permanent bipolaire

Le moteur unipolaire

Les bobinages d'un moteur unipolaire sont alimentés toujours dans le même sens par une tension unique d'où le nom d'unipolaire

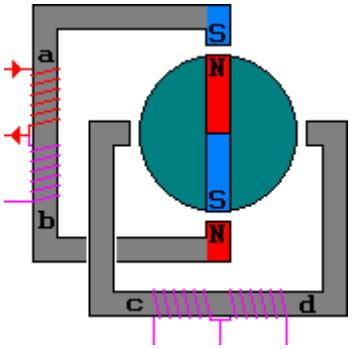


fig2 : moteur à aiment permanent unipolaire

.

C- Etude comparative

Nous effectuons une étude comparative entre les moteurs à aimant et le moteur à réluctance variable (les moteurs les plus utilisés)

Le tableau ci-dessous présente les avantages de l'un par rapport à l'autre :

Type de moteur	Moteur à aimant permanent	Moteur à réluctance variable	Moteur hybride
Résolution (nb de pas/tour)	Moyenne	Bonne	Elevée
Couple moteur	Elevé	Faible	Elevé
Sens de rotation	Il dépend : -du sens du courant pour les moteurs bipolaires -L'ordre d'alimentation des bobines	Il dépend uniquement de l'ordre d'alimentation des bobines	Il dépend : -du sens du courant pour les moteurs bipolaires -L'ordre d'alimentation des bobines
Fréquence de travail	Faible	grande	grande

D- Etude du courant dans un enroulement (phase) du moteur

E = L. di/ dt + r.i

L : L'inductance d'un enroulement du moteur

r : Résistance du moteur

Solution générale de l'équation sans second membre:

$$L\frac{di}{dt}+r\bullet i=0 \quad \Rightarrow \quad \frac{di}{i}=-\frac{r}{L}dt \quad \Rightarrow \quad Log\, i=-\frac{r}{L}t+K1 \quad \Rightarrow \quad i=K2\bullet e^{-\frac{r}{L}t}$$

Solution particulière de l'équation avec second membre:

$$i=\frac{E}{r}$$

Solution générale de l'équation avec second membre:

$$i=K2\bullet e^{-\frac{r}{L}t}+\frac{E}{r}$$

Détermination de la constante K2:

$$à\, t=0,\ i=0 \quad \Rightarrow \quad K2=-\frac{E}{r}$$

Equation finale

$$i=\frac{E}{r}-\frac{E}{r}e^{-\frac{r}{L}t}=\frac{E}{r}(1-e^{-\frac{r}{L}t})$$

$$\text{on pose: } \tau=\frac{L}{r} \quad \text{constante de temps (exprimée en secondes)}$$

$$i=\frac{E}{r}(1-e^{-\frac{t}{\tau}})$$

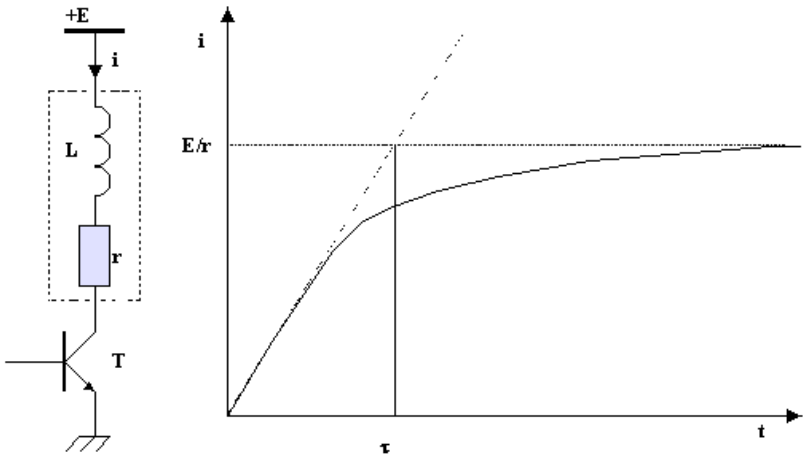


Fig3

- ***Réduction de la constante de temps***

On constate que la constante de temps τ = L/r peut être diminuée par l'addition d'une résistance r' en série avec l'enroulement. Cependant, la valeur du courant (régime permanent) est réduite. Le couple moteur est donc diminué. Pour le rétablir, il faut augmenter la tension d'alimentation du moteur.

On constate que la vitesse de montée du courant dans l'enroulement est plus élevée avec une résistance additionnelle r'. Le couple moteur s'établit donc plus rapidement. Les performances du moteur (fréquence maximale d'arrêt - démarrage et fréquence maximale de survitesse) sont considérablement améliorées.

Cependant, la résistance additionnelle dissipe inutilement une puissance $P=E^2/r'$.

E- Les avantages et les inconvénients des moteurs pas à pas :

L'avantage le plus important du moteur pas à pas est très simplifié puisqu'il existe des intégrés qui transforment directement un train d'impulsion en commande des phases, en tenant compte du sens désiré.

Un autre avantage réside dans le fait que le moteur pas à pas ne nécessite trop d'entretien, et que son usure est faible.De plus, il est possible de bloquer l'arbre sous tension sans que ceci ne nuise au moteur.

Comme inconvénients, il faut noter que la rotation se fait par coups et à une base vitesse, qu'il oscille lorsqu'il passe d'un pas à l'autre et que si le couple de charge est supérieur au couple moteur.

1- Circuit ULN 2003A

Pour commander le moteur pas à pas dans un sens bien défini et avec un mode de fonctionnement choisi il faut faire la commutation selon un organigramme convenable ,et puisque les signaux générés par le FPGA sont très faible .Au lieux d' utiliser des transistors de Darlington dont les bases sont attaqués par les signaux du FPGA et qui traduisent les chronogramme de chaque mode de fonctionnement du moteur on utilise un circuit spécialisé ,c'est le ULN2003A dans laquelle 8 transistors de Darlington sont intégré et 8 diode rapide (diode de roue libre) qui sont utiles pour le décharge des bobines du moteur au moment de commutation .Les caractéristiques de se circuit sont définis dans la fiche technique

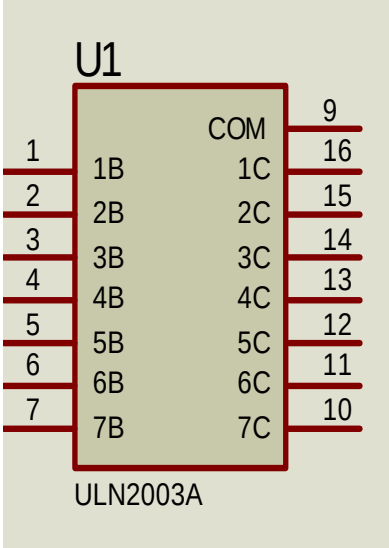


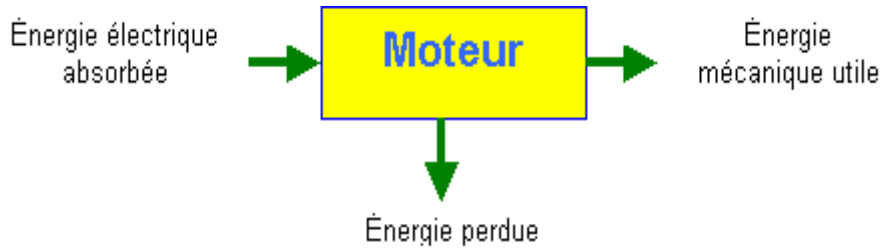
Fig8 : ULN 2003

2

Moteur à courant continu

1) Fonction d'usage :

Le moteur à courant continu est un convertisseur électrique en énergie mécanique avec quelques pertes.



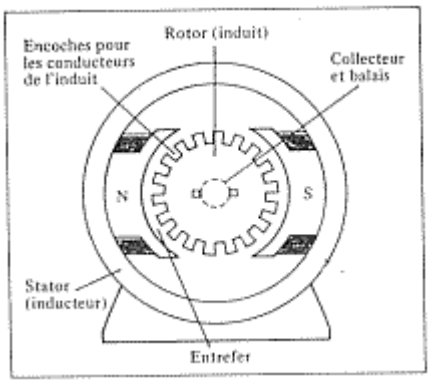


Fig1

2.) Principe :

Le moteur à courant continu est essentiellement composé de trois parties :

- ***Le stator ou inducteur :*** il est formé d'aimant pour les « petits » et de bobine parcouru par un courant pour les moteurs plus puissants. Grâce aux progrès sur les aimants ses derniers sont de plus en plus employés. Les aimants ou bobines permettent la création d'un flux magnétique.
- ***Le rotor ou induit :*** il est alimenté par une tension U et parcourue par un courant I, le circuit électrique appelé induit est obtenue en associant en série des

conducteurs logés dans des encoches du rotor.

- ***Le collecteur :*** il est formé d'un ensemble de lames de cuivre isolées latéralement les unes des autres et disposées suivant un cylindre en bout de rotor. Deux balais portés par le statuer frottent sur les lames des **collecteurs.**

3.) Représentation normalisée :

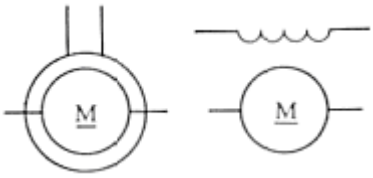


Fig2

4). Relations fondamentales :

Couple électromagnétique (C)

$$C = K\Phi I$$

Avec : I = courant absorbé

ϕ = flux inducteur en Webers (Wb)

K = coefficient de couplage électromagnétique (sans dimension)

C = couple électromagnétique en Newton.Mètre (N.m)

Le couple d'un moteur à courant continu est proportionnel aux flux et à l'intensité. Le flux crée par un aimant est constant dans ce cas, le couple est proportionnel au courant.

On pose $Kc = K\phi$ avec Kc qui est le coefficient de couple ou constante de couple.

$$C = KcI$$

La formule précédente devient alors :

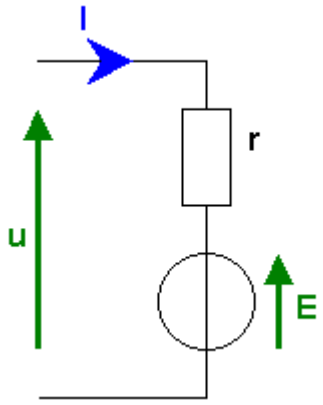


Fig3

U = rI + E

U = rI + KeN avec E = KeN

Avec E = en volt (V)

Ke =en volt par tour/seconde (V/tr/s)

N = vitesse du moteur en tour par seconde (tr/s)

Comme U = rI + KeN alors

U – rI = KeN $\Rightarrow$ N = (u – rI) / Ke

Avec I = constante à couple constant

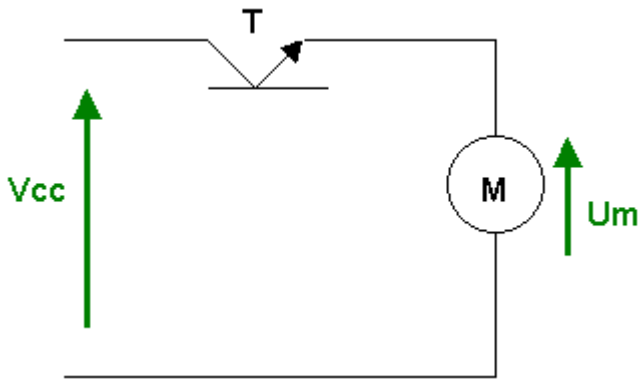
R = constante résistance de l'induit

Ke = constant

La variation de la vitesse n d'un moteur à courant continu est obtenu en faisant varier la valeur moyenne de la tension d'alimentation U.

On peut utiliser une tension continue variable mais le rendement peut être mauvais.

Pour résoudre ce problème, on place un transistor ballast entre la source d'alimentation et le moteur comme le schéma ci-dessous :

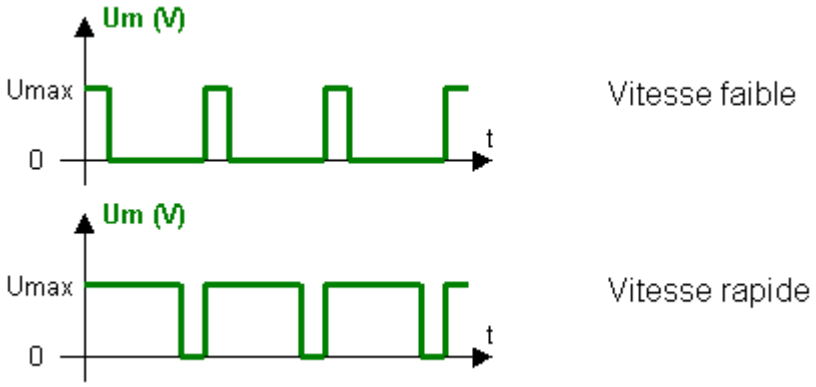


Vitesse maximale : Um » Vcc

⇒ bon rendement½ vitesse maximale : Um = Vcc/2

⇒ rendement de 50%

Pour résoudre ce premier problème, on utilise une commande MLI (Modulation de Largeur d'Impulsion) ou PWM (Pulse Width Modulation). Cette solution apporte un rendement de 100% quelque soit la vitesse du moteur.
Le MLI fait varier le rapport cyclique d'une tension ce qui fait aller plus ou moins vite le moteur.

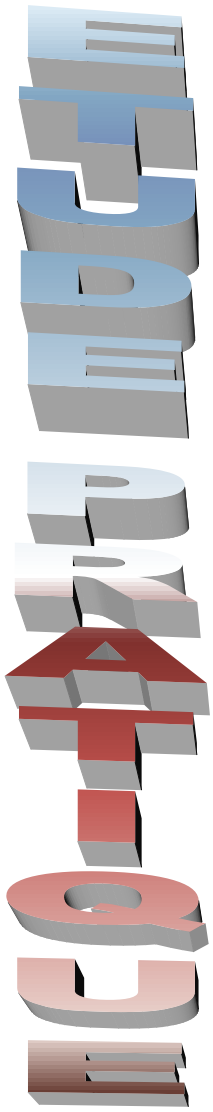


5) Principe général de PWM

Un modulateur PWM est en fait un compteur. On compare ce compteur à une valeur correspondant au rapport cyclique désiré. Si le compteur est en dessous de cette valeur, on met la sortie du circuit à 1, sinon on la met à 0. Le temps que met le compteur à reboucler donne la fréquence du PWM.

Généralement, on pré-divise l'horloge système par *n* pour aboutir à la fréquence PWM désirée. Pour un compteur 8 bits, la fréquence PWM sera donc : f pwm = (1/100)* f/n

Il existe plusieurs façons de réaliser un modulateur SSPWM (variation de phase, random XOR, NCO, ...). Nous avons pris une approche simple et facile à implémenter : on fera varier de façon aléatoire le coefficient de pré-division de l'horloge (*n*). Pour être suffisamment souple, on donnera en paramètres à notre modulateur les fréquences minimum et maximum admissibles



Schema Electrique

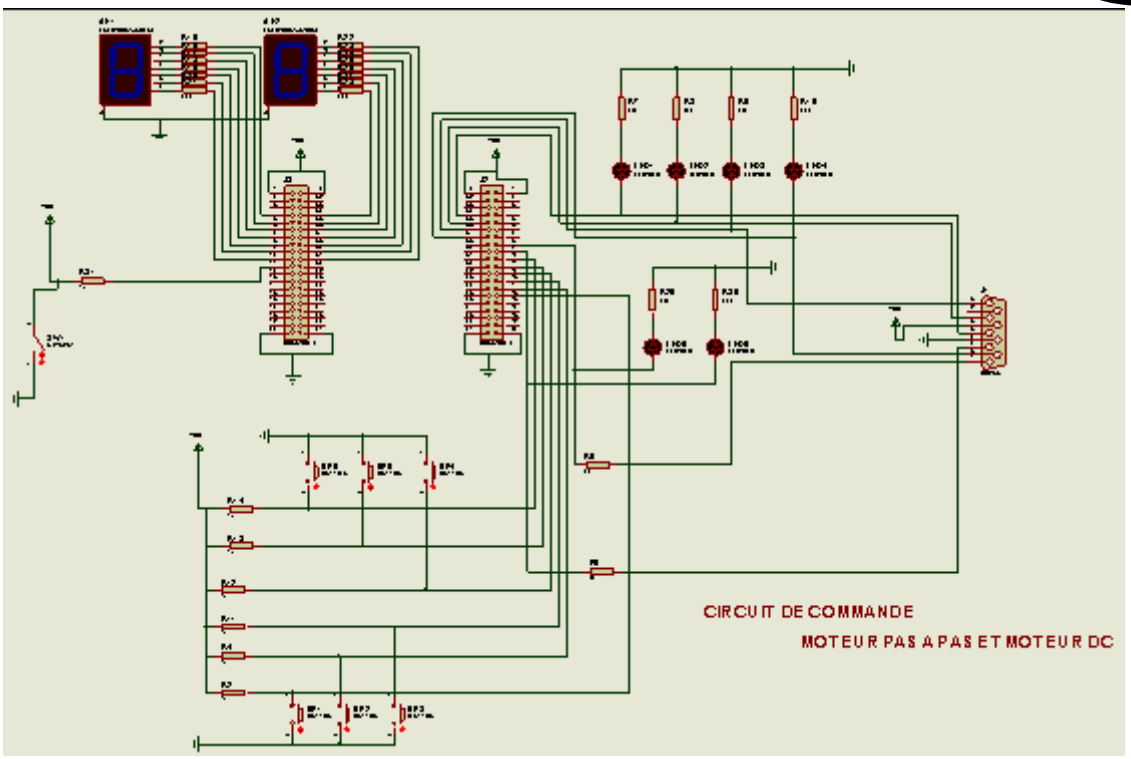


Fig1 : schema de circuit de commande à base de FPGA

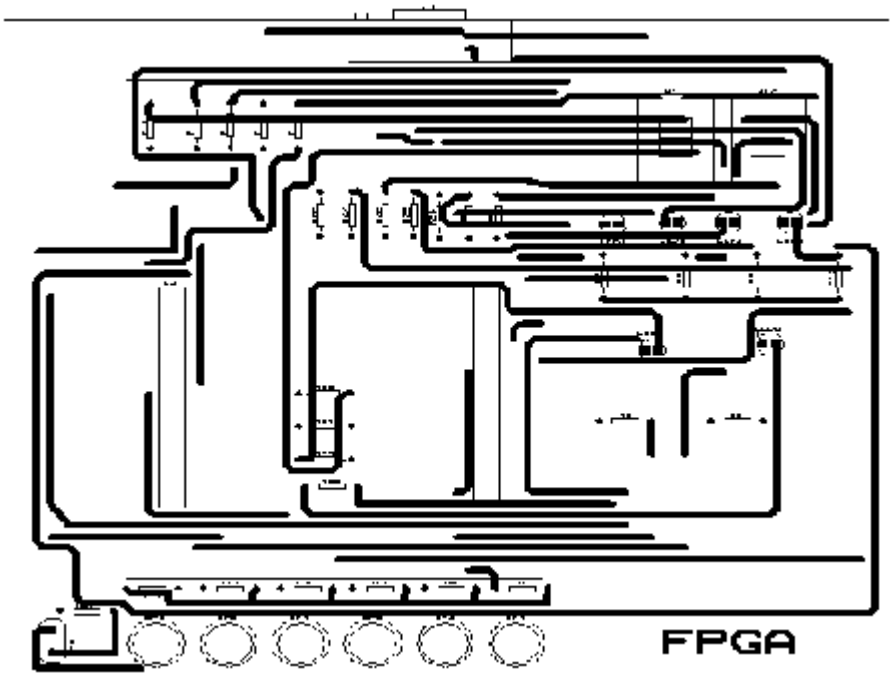


Fig2 : typon de la face composent

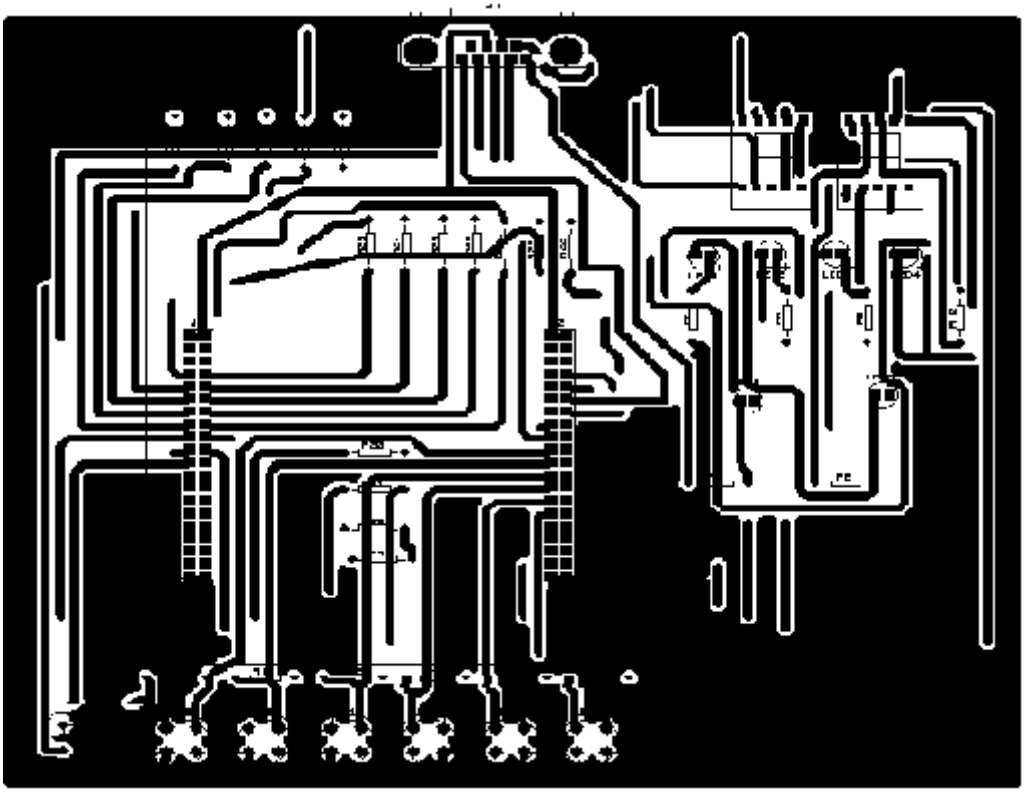


Fig3 : typon de la face cuivre

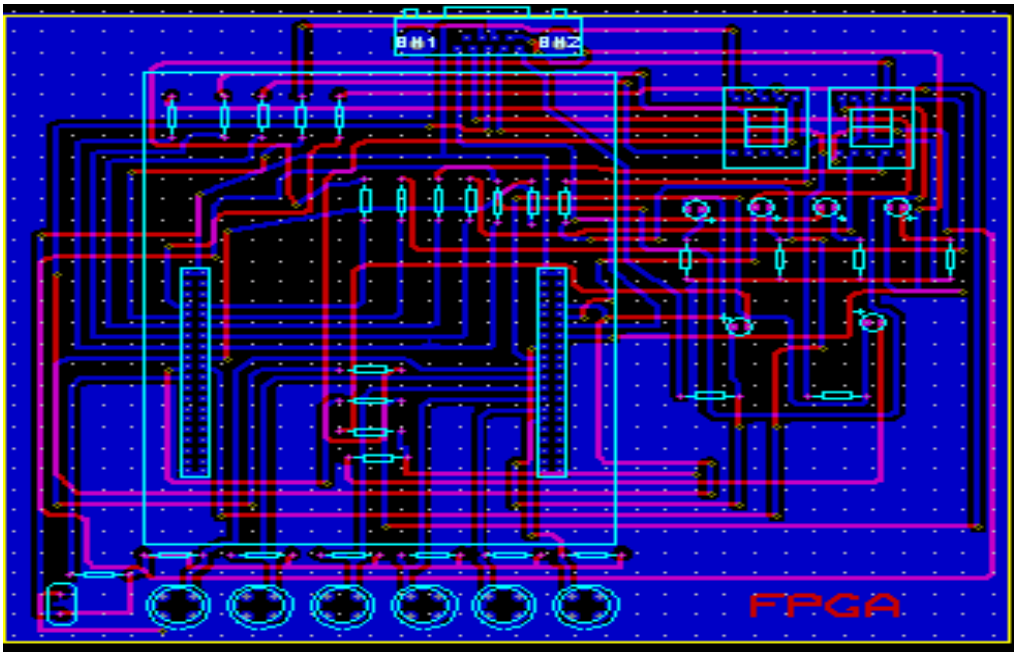


Fig3 : typon assembler

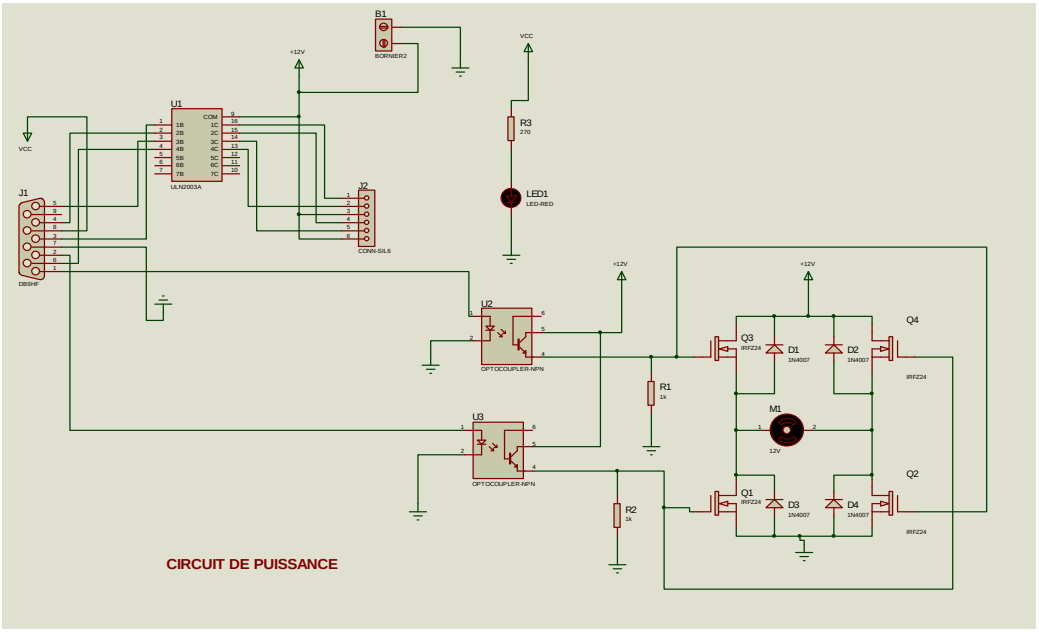


Fig4 : schema de circuit de puissance

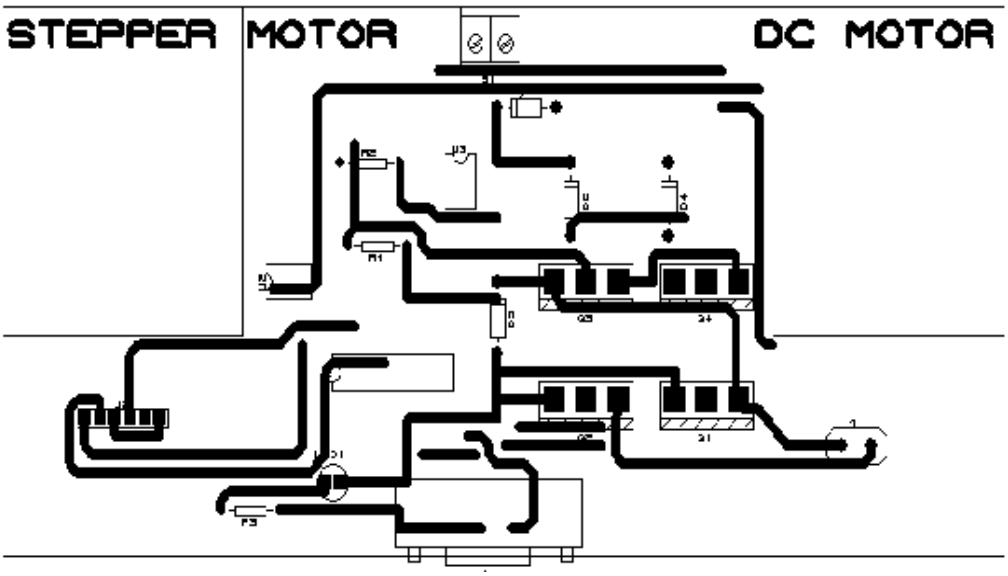


Fig5 : typon de la face composent

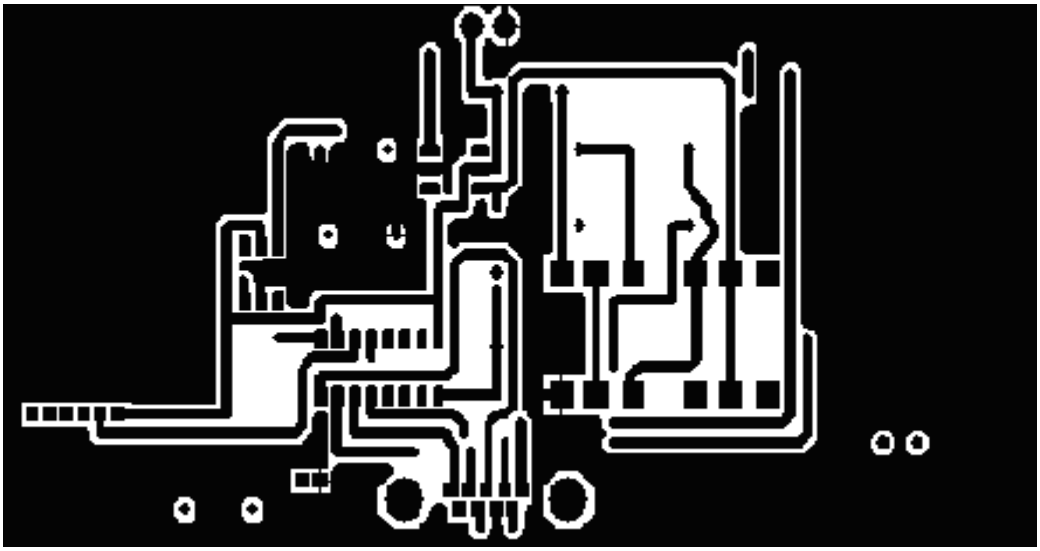


Fig6 : typon de la face cuivre

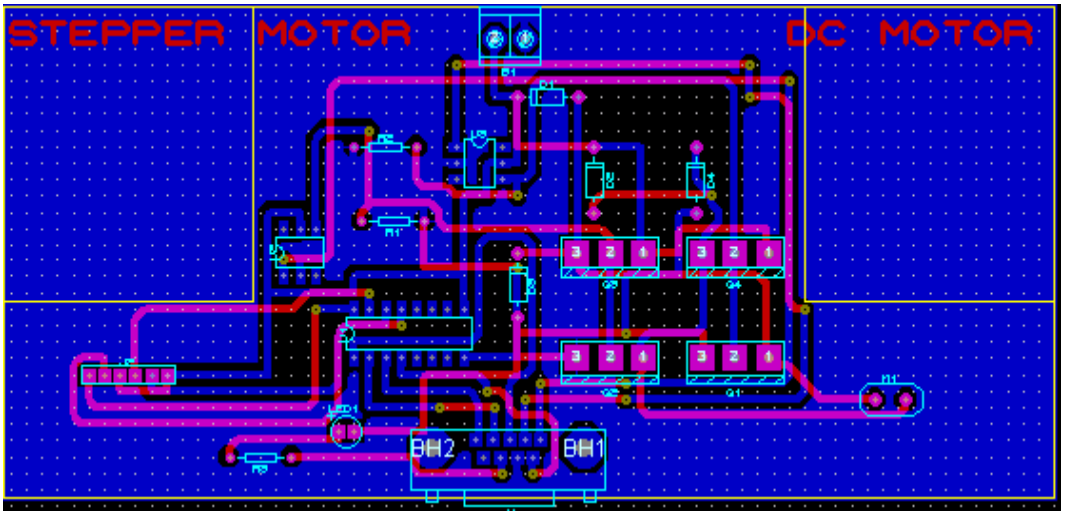
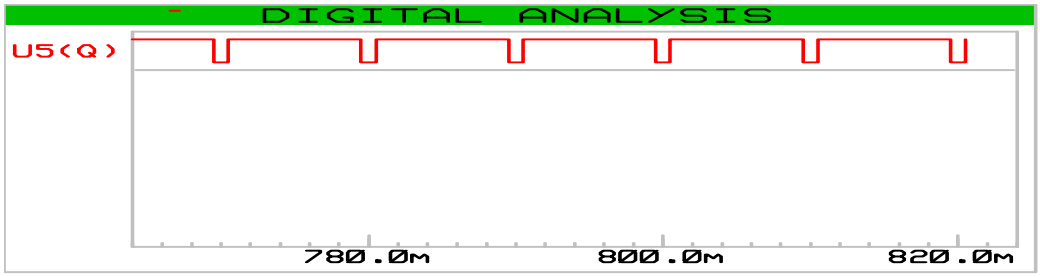
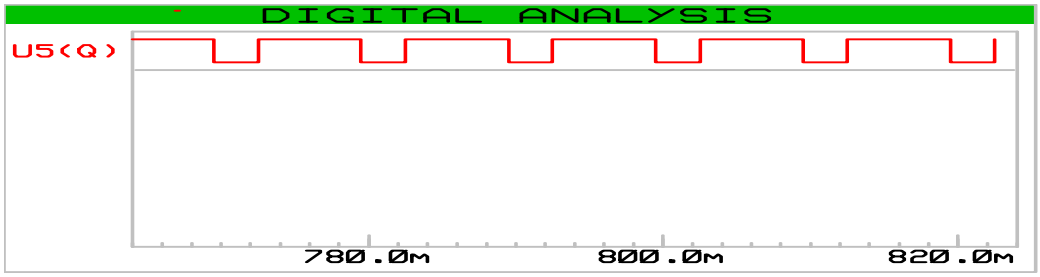


Fig7 : typon assemblé

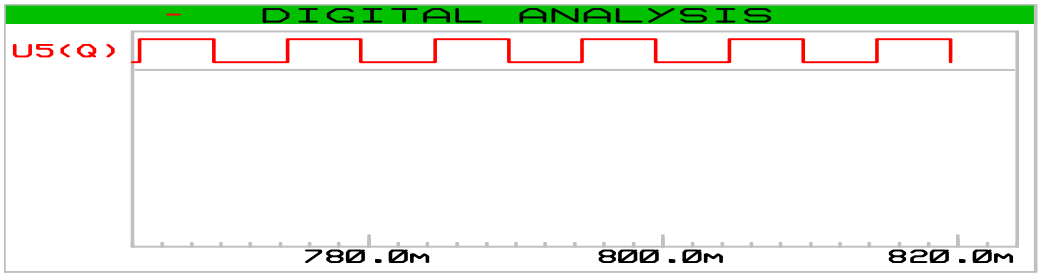
Courbes de variation de rapport cyclique



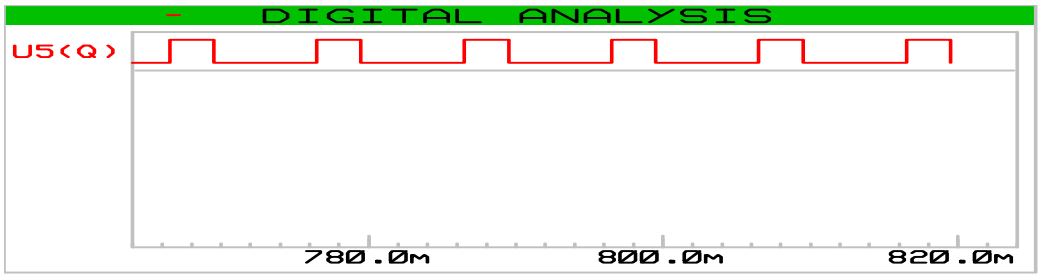
RAPPORT CYCLIQUE : α = 10%



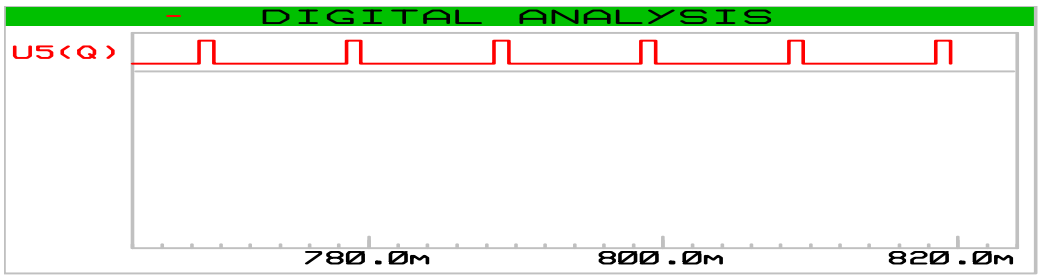
α = 30 %



α = 50 %



α = 70 %



α = 90 %

Liste des composants

29 Resistances

Quantité:	Références	Value
7	R2, R4, R11-R14, R31	1k
22	R5-R10, R15-R30	270

Quantité:	Références	Value
2	AF1, AF2	7SEG-COM-AN-BLUE
6	BP1-BP6	BUTTON
1	J1	DB9HM
2	J2, J3	CONN-DIL40
6	LED1-LED6	LED-RED
1	SW1	SW-SPST

1 Moteur

Quantité:	Références	Value
1	M1	12V

3 Resistances

Quantité:	Références	Value
2	R1, R2	1k
1	R3	270

3 Integrated Circuits

Quantité:	Références	Value
1	U1	ULN2003A
2	U2, U3	OPTOCOUPLER-NPN

4 Transistors

Quantité:	Références	Value
4	Q1-Q4	IRZ640

4 Diodes

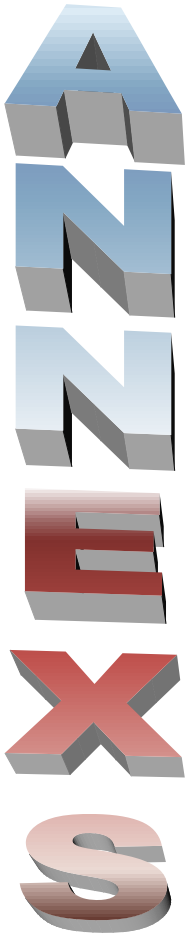
Quantité:	Références	Value
4	D1-D4	1N4007

Quantité:	Références	Value
1	B1	BORNIER2
1	J1	DB9HF
1	J2	CONN-SIL6
1	LED1	LED-RED

Brochage de FPGA

VCC	***VCC***	***ENREE***
I12	***CRYSTALE***	***ENTREE***
P01	***'a'DE 1^{IER} AFFICHEUR***	***SORTIE***
P02	***'A'DE 2EME AFFICHEUR***	***SORTIE***
P03	***'B' DE 1^{IER} AFFICEUR***	***SORTIE***
P04	***'B'DE 2EME AFFICHEUR***	***SORTIE***
P06	***'C'DE 1^{IER} AFFICHEUR***	***SORTIE***
P07	***'C'DE 2EME AFFICHEUR***	***SORTIE***
P08	***'D'DE 1^{IER} AFFICHEUR***	***SORTIE***
P09	***'D'DE 2EME AFFICHEUR***	***SORTIE***
P13	***'E'DE 1^{IER} AFFICHEUR***	***SORTIE***

P14	***'E'DE 2EME AFFICHEUR***	***SORTIE***
P15	***''F'DE 1^{IER} AFFICHEUR***	***SORTIE***
P16	***'F'DE 2EME AFFICHEUR***	***SORTIE***
P18	***'G'DE 1^{IER} AFFICHEUR***	***SORTIE***
P19	***'G'DE 2EME AFFICHEUR***	***SORTIE***
P20	***SWETCHEUR***	***ENTREE***
P44	***LED 1ET BOBINE 1***	***SORTIE***
P46	***LED 2 ET BOBINE 2***	***SORTIE***
P49	***LED 3 ET BOBINE 3***	***SORTIE***
P51	***LED 4 ET BOBINE 4***	***SORTIE***
P55	***LED 5 ET SENS (+ MCC)***	***SORTIE***
P57	***LED 6 ET SENS (-) MCC***	***SORTIE***
P71	***BOUTTON 1***	***ENTREE***
P69	***BOUTTON 2***	***ENTREE***
P66	***BOUTTON 3***	***ENTREE***
P64	***BOUTTON 4***	***ENTREE***
P62	***BOUTTON5***	***ENTREE***
P60	***BOUTTON 6***	***ENTREE***



FPGA : Fieled Programmable Gate Array

VHDL : Vhsic Hardwere Description Langage

VHSIC Very High Speed Integrated Circuit

PLD: Programmable Logique Device

ASIC: Application Specific Integrated Circuit

EPROM: Erasable Programmable Read Only Memory

EEPROM: Electrically EPROM

SRAM: Statique Randon Access Memory

CLB: Configurable Logique Bloc

IOB: Input Output Bloc

FF: Flip Flop

HDL: Hardware Description Language

IEEE: Institue of Electrical and Electronics Enginners

CPLD: Complex Programmable Logique Device

PROM: Programmable Read Only Memory