الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Khider – Biskra

Faculté des Sciences et de la technologie

Département : Génie électrique

Ref :……

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم: الهندسة الكهربائية
المرجع:……

Mémoire présenté en vue de l'obtention
Du diplôme de

Magister en : Electronique

Option : Signaux et communication

Méthodologie de développement et d'implantation sur puce FPGA d'algorithme de commande

Présenté par :

Houari ABBAD

Soutenu publiquement le 29/06/2016

Devant le jury composé de :

Dr.Benakcha Abdelhamid	**Maitre de Conférences 'A'**	**Président**	**Université de Biskra**
Dr. Abderrazak **DEBILOU**	**Maitre de Conférences 'A'**	**Rapporteur**	**Université de Biskra**
Dr. Ouafi Abdelkarim	**Maitre de Conférences 'A'**	**Examinateur**	**Université de Biskra**
Dr. Ghodebane Hatem	**Maitre de Conférences 'A'**	**Examinateur**	**Université de Biskra**

Dédicace

Avant tout, je tien à remercie le bon dieu, et l'unique qui m'offre le courage et la volonté nécessaire pour affronter les différentes de la vie.

Je dédie ce modeste travail

A ma mère.

A mon père.

A mes frères

A mes sœurs

et toute la famille.

A mes amis et mes collègues de la promotion Magistère

Remerciements

Nous remercions le Dieu de nous avoir donné la force et le courage pour réaliser ce modeste travail.

Nous remercions Monsieur **Debilou Abderrazak** **et** **Mlle. Gargazi** qui on suivi de très près ce travail, pour son aide, ses orientations pédagogiques dans l'élaboration de ce mémoire, et tous les conseils qu'il nous a prodigués pendant toute la durée de ce travail malgré ses nombreuses occupations.

Nous adressons nos plus vifs remerciements aux membres du jury pour l'honneur qu'ils nous font en acceptant la charge de juger ce travail.

Nous sommes reconnaissants envers nos enseignants auxquels nous devons notre formation.

Enfin, nous ne pourrons terminer ces remerciements sans remercier notre famille pour leurs aides, compréhensions, encouragements et soutiens, qu'elles nous ont apportés tout le long de nos études et à toutes nos amies.

« Signaux et communication ».

Méthodologie de développement et d'implantation sur puce FPGA d'algorithmes de commande

- Mots Clefs

- FPGA
- Mesure de performances
- Commande
- Temps réel

Résumé

L'objectif de ce travail est de trouver une méthodologie pour l'implantation des algorithmes de commande sur cible FPGA, on a fait notre départ de travail par partager le projet ont deux grandes parties : une partie théorique et une partie pratique.

Pour la 1ère partie, on a fini le premier chapitre qui est consacré sur les réseaux logiques programmables PLD, le 2ème chapitre qui, sera destiné à la technologie et environnement de développement des FPGA et le dernier sur la Méthodologie de développement des algorithmes de commande et l'implantation sur FPGA.

La 2ème partie contient le dernier chapitre, notre motivation était de réaliser un dispositif de commande numérique à base d'une puce d'FPGA pour un onduleur triphasé pédagogique, c'est une pratique de ce que nous avant étudient dans la partie théorique, alors ont à débuté dans cette partie par préparer un cahier des charges pour faire une boucle de commande vectorielle pour moteur asynchrone, et voire ce que nous avant comme matérielle pédagogique (les cartes FPGA) dans le laboratoire de faculté, après ont à choisir la carte FPGA DE0 d'Altéra comme une cible, on même temps commencé à partager la boucle d'acquisition sur des petits blocs d'algorithmes pour faciliter l'écriture et la simulation du programme , finalement ces propositions d'avancement sont validées en simulation.

Methodology development and implementation on FPGA control algorithms

- Key Words

- FPGA
- Measuring performance
- command
- real time

Summary

The objective of this work is to find a methodology for the setting-up of the algorithms of command on target FPGA, we made our working departure to share the project have two big parts: a theoretical part and a part practises.

For the 1st part, we finished the first chapter which is dedicated on the programmable logical networks PLD, the 2nd chapter, which, will be intended for the technology and the environment of development of the FPGA and the last one on the Methodology of development of the algorithms of command and the setting-up on FPGA.

The 2nd part contains the last chapter, our motivation was to realize a digital control system with a flea of FPGA for an educational three-phase inverter, it is the practice of what we before study in the theoretical part, then have to begin in this part to prepare a specifications to make a buckle of vectorial command for asynchronous motor, and even that we before as material educational (cards FPGA) in the laboratory of faculty, Later have to choose the card FPGA DE0 of Altera as a target, one the same time begun to share the loop of acquisition on small blocks of algorithms to facilitate the writing and the simulation of the program, finally these proposals of progress are validated in simulation.

منهجية تطوير و برمجت خوارزميات التحكم على شرائح مصفوفة البوابات المنطقية القابلة للبرمجة

كلمات مفتاحية

- شرائح مصفوفة البوابات المنطقية القابلة للبرمجة
- قياس الأداء
- تحكم
- في الوقت الحقيقي

خلاصة

الهدف من هذا العمل هو البحث عن منهجية لبرمجة خوارزميات التحكم على شرائح مصفوفة البوابات المنطقية القابلة للبرمجة , حيث انطلقنا للعمل في المشروع بتقسيم هذا الأخير إلى جزئين ,جزء نظري وجزء تطبيقي.

بالنسبة للجزء الأول كان مخصص للفصل الأول الذي نشرح فيه شبكات المنطق PLD , الفصل الثالني لتكنولوجيا و محيط تطوير FPGA و الفصل الأخير الذي يهتم بمنهجية عمل المشروع و تنفيذه على FPGA .

الجزء الثاني و الذي يمثل آخر فصل لدينا و الذي فيه نريد تحقيق مشروع تحكم رقمي بالإستعانة بالشريحة FPGA وهذا تطبيق ما قد قمنا بدراسته في الجزء الأول (الجزء النظري) و أول ما بدأنا به هوإنشاء كراسة الشروط و المواصفات الفنية ثم البحث في المخبر على الأجهزة الإلكترونية و بالأخص شرائح FPGA وأنواعها و من ثم أنطلقنا بتنفيذ مشروعنا المتمثل في إنشاء برنامج للتحكم الشعاعي لمحرك كهربائي ثلاثي التغذية هذا قبل أن نقوم بإختيار الشريحة FPGA -DE0 للشركة Altera و من ثم كتابة البرنامج و محاكاته .

Table des matières

CHAPITRE III : Méthodologie de développement d'un algorithme de commande pour l'implantation sur puce FPGA

Table des Notations et symboles

SSI	Small Scale Integration
MSI	Medium Scale Integration
LSI	Large Scale Integration
VLSI	Very Large Scale Integration
ASIC	Application Specific Integrated Circuits
PLA	Programmable Logic Array
PAL	Programmable Array Logic
PLD	Programmable Logic Device
FPLA	Field Programmable Logic Array
SPLD	Simple Programmable Logic Device
Signal OE	Output Enable
CPLD	Complex Programmable Logic Device
FPGA	Field Programmable Gate Array
LUT	Lock UpTable
OTP	One Time Programming
RAM	Random Access Memory
ROM	Read Only Mémory
EPROM	Erasable Programmable Read Only Mémory
MOS	Métal-Oxyde- Semi-conducteur
EEPROM	Electrically Erasable Programmable Read Only Memory
NOVRAM	NOn Volatile RAM
ISP	In Sito Programmation
OLMC	Output Logic MacroCell, dénomination Lattice
SRAM	Static Random Access Memory
VHDL	**V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**angage
DSP	Digital Signal Processor
NRE	Les coûts d'ingénierie non récurrents
OEM	Original Equipment Manufecturer
PFU	Programmable Functional Unit
ECP	Electronic Check Presentment
FPSC	Field Programmable System Chip
CLB	Configurable Logic Block
ECU	Embedded Computational Units
LAB	Logic Array Blocs
CAO	La conception assistée par ordinateur
CTO	Chief Technology Officer
Co- Design	conception conjointe logiciel matériel
PCI	Peripheral Component Interconnect
PLL	Phase Locked Loops
DLL	*Delay Locked Loops*
MAC	Multiplieur Accumulateur
SOPC	System On a Programmable Chip

CPU	Central Processing Unit
ALM	Adaptive Logic Module
LEs	Logic Elements
LABs	Logic Array Blocs
TSMC	Taiwan Semiconductor Manufacturing Company
LUT	Look Up Table
RISC	Reduced instruction set computer
PIO	Parallel Input Output
mppSoC	parametric massive parallel
OFDMA	Orthogonal Frequency Division Multiple Access
MIMO	multiple-input multiple-output
IGBT	insulated-gate bipolar transistor
GFD	Graphe de Flot de Données
JTAG	Joint Test Action Group
MLI	modulation de largeur d'impulsions
PWM	pulse width modulation
SVpwm	space vector pulse width modulation

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Liste des tableaux

Introduction Générale

Introduction générale

Les entraînements électriques ont fait l'objet d'importantes révolutions, notamment dans le cadre de l'amélioration des performances de composants d'électronique de puissance et dans l'utilisation de nouvelles solutions numériques comme support pour l'implantation des algorithmes de commande. Les premières implantations d'algorithmes de commande furent réalisées avec des solutions analogiques. Ces solutions assuraient la réalisation de contrôles ayant une large bande passante vu leur rapidité et leur action en continu. Cependant, elles manquaient de fiabilité à cause de leur sensibilité aux perturbations et aux variations des paramètres de contrôle liées aux contraintes thermiques des circuits analogiques de contrôle. Pour remédier à ces inconvénients, les solutions numériques se sont naturellement imposées. Les premières réalisations numériques d'implantation d'algorithmes de commande de machines électriques ont été effectuées en utilisant les microcontrôleurs, les microprocesseurs et les DSP (Digital Signal Processor). Ces solutions numériques ont permis de résoudre les problèmes liés à l'utilisation des commandes analogiques. Par ailleurs, elles présentaient un grand intérêt économique et une meilleure flexibilité de conception. Cependant, malgré les avantages offerts par ces solutions numériques, certains avantages offerts par les implantations analogiques sont perdus. Cela est principalement dû au fait que la discrétisation et la quantification des algorithmes de commande à implanter, ainsi que les délais de temps de calcul, détériorent les performances de contrôle en termes de rapidité de correction et de résolution de contrôle.

Avec l'avancement technologique dans le domaine de la microélectronique, de nouvelles solutions numériques telles que les FPGA (Field Programmable Gate Array) ou les ASIC (Application Specific Integrated Circuit) sont disponibles et peuvent être utilisées comme cibles numériques pour l'implantation des algorithmes de commande. Le parallélisme inhérent de ces nouvelles solutions ainsi que leurs grandes capacités de calcul font que les délais de temps de calcul sont négligeables en dépit de la complexité des algorithmes à implanter. L'utilisation de ces solutions matérielles permet donc de retrouver certaines performances analogiques tout en gardant les avantages des solutions numériques. De plus, ces solutions permettent de répondre aux nouvelles exigences des contrôles modernes. En effet, outre l'amélioration des performances de contrôle à travers la réduction des temps de calcul, le parallélisme des solutions matérielles permet d'intégrer sur une seule et unique cible plusieurs algorithmes qui assurent différentes fonctionnalités et qui peuvent travailler indépendamment les uns des autres. Par ailleurs, par rapport aux solutions numériques standard utilisées dans les entraînements électriques à vitesse variable, les FPGA offrent au concepteur un accès à la partie architecture matérielle, puisque c'est le concepteur lui-même qui assure sa conception. Néanmoins, ce nouveau degré de liberté présente une difficulté de plus pour le concepteur puisque c'est à lui de mettre en œuvre l'architecture de contrôle. Pour ce faire, lors de l'implantation d'algorithmes sur cible FPGA, il est judicieux de se baser sur une approche méthodique plus automatisée et moins intuitive. Cette approche consiste en une méthodologie

de développement qui permet de résoudre l'adéquation entre l'algorithme de commande à implanter et son architecture en vue d'effectuer une implantation optimisée en termes de ressources consommées et de temps de calcul, tout en réduisant le temps de développement.

Pour les entraînements électriques à vitesse variable, plusieurs algorithmes de contrôle peuvent être utilisés. Ces algorithmes comportent souvent plusieurs boucles de régulation imbriquées. Il s'agit des boucles de régulation de courant, de vitesse, de position... La boucle de régulation du courant est souvent la plus difficile à implanter car elle constitue généralement la partie la plus complexe et la plus sensible de l'algorithme de commande. Les autres boucles de régulation sont relativement plus simples à implanter. Dans ce dossier, on s'intéresse particulièrement à l'implantation sur cible FPGA de techniques de contrôle du command type SVPWM les plus couramment utilisées pour la commande d'une machine asynchrone. Dans ce qui suit, l'apport et l'intérêt de l'utilisation des FPGA comme support pour l'implantation de ces techniques de contrôle sont discutés et analysés [42].

Le domaine des FPGA est dominé par le duopole que forment Xilinx/Altera, avec Actel et lattice jouant les solides seconds rôles .Cette situation perdure depuis vingt ans mais pourrait bien s'animer avec l'avènement d'architectures originales concoctées par de jeunes pousses comme Abound Logic, Achronix, Menta, SiliconBlue, Tabula ou Tier Logic [39].

Historique

Depuis 1983 et l'avènement des réseaux logiques programmables, plus de cinquante sociétés se sont lancées dans cette aventure et plus d'une quarantaine ont disparu du paysage. Celles-ci ont été soit rachetées, soit réorientées, voire ont tout simplement mis la clef sous la porte. L'attrait du caractère programmable des FPGA justifie, entre autres, le dynamisme en ce domaine. Cependant, malgré ces tentatives, jusqu'ici, régnait une certaine quiétude dans le monde des FPGA. Les deux acteurs d'origine, Xilinx et Altera, ont ainsi traversé cette vingtaine d'années en éternels frères ennemis, assumant pleinement leur statut incontesté de leaders du domaine. La société d'étude Gartner estime d'ailleurs à 86 % leur emprise sur le marché des FPGA. Les deux acteurs suivants, plus petits mais néanmoins bien établis, sont Actel et Lattice qui possèdent chacun 6 % de parts de ce marché.

En dépit des apparitions et disparitions d'intervenants, la situation semblait donc fermement installée, voire immuable, entre ces quatre sociétés de FPGA qui, chacune dans leur spécialité, faisaient évoluer leurs matrices en tirant le meilleur parti des avancées technologiques. Des progrès qui s'avèrent plus souvent redevables à la loi de Moore qu'à de profondes modifications de l'architecture de la matrice elle même. Vingt ans après, six « mousquetaires » pourraient changer la donne. En quelques mois, Abound Logic, Achronix, SiliconBlue, Tabula et Tier Logic, voire Menta (encadré), ont non seulement survécu à la crise de 2009 mais osent, pour certaines, présenter des architectures profondément novatrices associées à des outils de développement qui ne dépaysent pas le concepteur. Le phénomène est trop rare pour ne pas s'y arrêter un instant et étudier ces nouvelles offres ainsi que leurs chances de réussite. D'autant plus que de nombreux experts du domaine des FPGA s'interrogent car ces cinq sociétés réunissent deux des premiers critères d'un succès sur ce marché : avoir de «

bons outils » associés à des avantages technologiques clairs et convaincants. Un point important car les fournisseurs de logique programmable ont traditionnellement fourni aux utilisateurs des outils logiciels de développement à très bas coût, voire gratuits. Ainsi, le prix à payer pour entrer sur ce marché concerne non seulement la R&D au niveau silicium mais également les outils logiciels [39].

Problématique

Avec l'avancement technologique et la complexité des systèmes de commande moderne il est difficile de définir d'une manière universelle ou une structure générale pour de tels systèmes. Les algorithmes des applications de traitement de signal intensif, telles que les applications de télédétection (radar, sonar, etc.), et de l'image deviennent de plus en plus sophistiqués. Ils mettent en jeu un volume considérable de données. Ce domaine d'applications requiert une grande puissance de calcul, que seules les architectures parallèles spécialisées taillées sur mesure peuvent satisfaire les contraintes de vitesse, de performance et d'encombrement. La parallélisation et les architectures parallèles semblent être la voie la plus prometteuse pour répondre à la complexité applicative. Il est alors nécessaire de faire appel à une méthodologie pour la conception et l'implantation des architectures sur les cibles parallèle comme les FPGA et PLD.

Contribution

C'est dans ce cadre que se situe notre travail. Il s'agit de trouver une méthodologie de développement pour l'implantation d'un algorithme de commande sur puce FPGA, une méthode facile à appréhender par l'ingénieur électrotechnicien sans qu'il soit expert en microélectronique. Ce travail est basé sur une méthode de développement appropriée qui permet de répondre aux différentes contraintes de conception architecturales des algorithmes de commande complexe.

Plan

On a fait notre départ de travail par partager le projet ont deux grandes parties (partie théorique et partie pratique), le manuscrit est organisé selon le plan suivant :

Pour la 1re partie on a fini le premier chapitre qui est consacré sur les réseaux logiques programmables PLD, la 2éme chapitre qui sera destinés à la technique et environnement de développement des FPGA et le dernier sur la méthode de développement et d'implantation sur puce FPGA.

La 2émme partie contient notre dernier chapitre, notre motivation était de réaliser un dispositif de commande numérique à base d'une puce d'FPGA pour un onduleur triphasé pédagogique, c'est une pratique de ce que nous ayons étudié dans la partie théorique.

On terminera par une conclusion générale et d'une bibliographie indiquant quelques sources d'information utilisées.

CHAPITRE I

Composants à réseaux logiques programmables

I.1. Introduction

L'électronique moderne se tourne de plus en plus vers le numérique qui présente de nombreux avantages sur l'analogique : grande insensibilité aux parasites et aux dérives diverses, modularité et (re)configurabilité, facilité de stockage de l'information etc...

Les circuits numériques nécessitent par contre une architecture plus lourde et leur mode de traitement de l'information met en œuvre plus de fonctions élémentaires que l'analogique d'où découle des temps de traitement plus long. Aussi les fabriquants de circuits intégrés numériques s'attachent-ils à fournir des circuits présentant des densités d'intégration toujours plus élevée, pour des vitesses de fonctionnement de plus en plus grandes.

D'abord réalisées avec des circuits SSI (Small Scale Integration) les fonctions logiques intégrées se sont développées avec la mise au point du transistor MOS dont la facilité d'intégration a permis la réalisation de circuits MSI (Medium Scale Integration) puis LSI (Large Scale Integration) puis VLSI (Very Large Scale Integration). Ces deux dernières générations ont vu l'avènement des microprocesseurs et microcontrôleurs.

Bien que ces derniers aient révolutionné l'électronique numérique par la possibilité de réaliser n'importe quelle fonction par programmation d'un composant générique, ils traitent l'information de manière séquentielle (du moins dans les versions classiques), ne répondant pas toujours aux exigences de rapidité.

Au début des années 70 sont apparus les premiers composants (en technologie bipolaire) entièrement configurable par programmation. La nouveauté résidait dans le fait qu'il était maintenant possible d'implanter physiquement par simple programmation, au sein du circuit, n'importe quelle fonction logique, et non plus de se contenter de faire réaliser une opération logique par un microprocesseur dont l'architecture est figée.

D'abord dédiés à des fonctions simples en combinatoire (décodage d'adresse par exemple), ces circuits laissent aujourd'hui au concepteur la possibilité d'implanter des composants aussi divers qu'un inverseur et un microprocesseur au sein d'un même boîtier ; le circuit n'est plus limité à un mode de traitement séquentielle de l'information comme avec les microprocesseurs. L'intégration des principales fonctions numériques d'une carte au sein d'un même boîtier permet de répondre à la fois aux critères de densité et de rapidité (les capacités parasites étant plus faibles, la vitesse de fonctionnement peut augmenter).

La plupart de ces circuits sont maintenant programmés à partir d'un simple ordinateur type **PC** directement sur la **carte** où ils vont être utilisés. En cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement.

De nombreuses familles de circuits sont apparues depuis les années 70 avec des noms très divers suivant les constructeurs : des circuits très voisins pouvaient être appelés différemment par deux constructeurs concurrents, pour des raisons de brevets et de stratégies commerciales. De même une certaine inertie dans l'évolution du vocabulaire a fait que certains circuits technologiquement différents ont le même nom.

Le terme même de circuit programmable est ambigu, la programmation d'un FPGA ne faisant pas appel aux mêmes opérations que celle d'un microprocesseur. Il serait plus juste de parler pour les PLD, CPLD et FPGA de circuits à architecture programmable ou encore de circuits à réseaux logiques programmables.

Ce domaine de l'électronique est aussi celui qui certainement a vu la plus forte évolution technologique ces dernières années :

- en moins de 15 ans la **densité d'intégration** à été multipliée par **200** (2000 à 20 000 portes en 85 pour 72 000 à 4 000 000 en 2000).
- en moins de 10 ans la **vitesse de fonctionnement** par **6** (40 MHz en 91 pour 240 MHz en 2000).
- la taille d'un transistor est passée **de 1,2 µm en 91à 0,18 µm** en 2000.
- les technologies de conception ont fortement évolué, tel constructeur initiateur d'un procédé l'abandonne pour un autre, alors que le concurrent le reprend à son compte.
- la tension d'alimentation est passée **de 5 V à 1,8 V** diminuant ainsi la consommation.

Aussi est-il très difficile de s'y retrouver et de donner des ordres de grandeurs qui puissent être comparés. Nous tenterons dans cet exposé une clarification des choses dont la volonté de simplification pourra être facilement prise en défaut.

Parallèlement à ces circuits, on trouvera les ASIC (Application Specific Integrated Circuits) qui sont des composants où le concepteur intervient au niveau du dessin de la pastille de silicium en fournissant des masques à un fondeur. On ne peut plus franchement parler de circuits programmables. Les temps de développement long ne justifient l'utilisation que pour des grandes séries.

Les PLD, CPLD et FPGA sont parfois considérés comme des ASIC par certains auteurs. Le tableau ci-après tente une classification possible des circuits numérique :

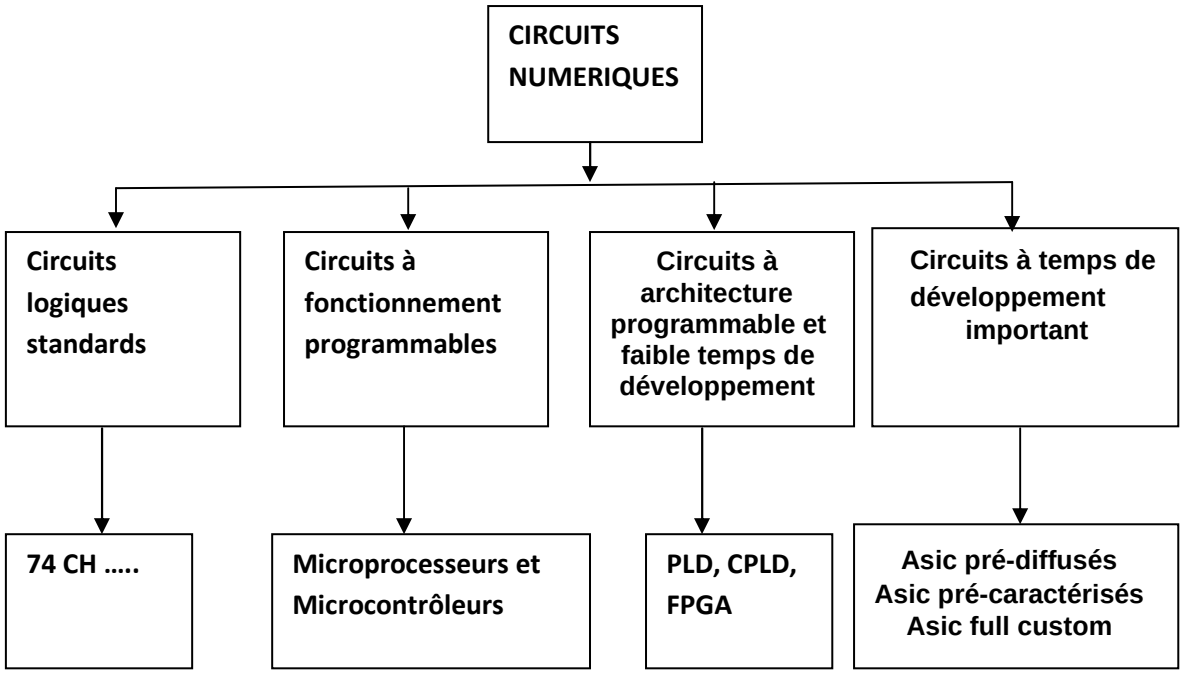


Figure I-1 : Le tableau classification possible des circuits numérique.

Nous nous intéresserons surtout aux circuits à architecture programmable à faible temps de développement. Le principe de base des circuits nous intéressant ici consiste à réaliser des connexions logiques programmables entre des structures présentant des fonctions de bases. Le premier problème va donc être d'établir ou non suivant la volonté de l'utilisateur, un contact électrique entre deux points. Aussi, nous intéresserons nous, avant de passer aux circuits proprement dits et à leur programmation, à ces technologies d'interconnexion. Mais avant toutes choses, rappelons Qu'est-ce qu'un circuit programmable [32].

I.2. Les circuits programmables

Un circuit programmable est un assemblage d'opérateurs logiques combinatoires et de bascules dans lequel la fonction réalisée n'est pas fixée lors de la fabrication. Il contient potentiellement la possibilité de réaliser toute une classe de fonctions, plus ou moins large suivant son architecture. La programmation du circuit consiste à définir une fonction parmi toutes celles qui sont potentiellement réalisables.

Comme dans toute réalisation en logique câblée, une fonction logique est définie par les interconnexions entre des opérateurs combinatoires et des bascules (synchrones, cela va presque sans dire), et par les équations des opérateurs combinatoires. Ce qui est programmable dans un circuit concerne donc les interconnexions et les opérateurs combinatoires. Les bascules sont le plus souvent de simples bascules D, ou des bascules configurables en bascules D ou T.

La réalisation d'opérateurs combinatoires utilise des opérateurs génériques, c'est à eux que nous allons nous intéresser dans la suite [34].

I.3. Les opérateurs combinatoires génériques

Les opérateurs combinatoires génériques qui interviennent dans les circuits programmables proviennent soit des mémoires (réseaux logiques) soit des fonctions standard (multiplexeurs et ou exclusif) [34].

La base d'une fonction logique, est toujours une fonction combinatoire. Pour obtenir une fonction séquentielle, il suffira ensuite de réinjecter les sorties sur les entrées, ce qui donnera alors un système asynchrone. Si on souhaite un système synchrone, on intercalera avant les sorties, une série de bascules à front, dont l'horloge commune synchronisera toutes les données et évitera bien des aléas de fonctionnement. Pour coder une fonction combinatoire, trois solutions sont classiquement utilisées [32].

I.3.1. Sommes de produits, produits de somme et matrice PLA (Programmable Logic Array)

La plupart des applications n'exigent pas une telle complexité et on peut se contenter d'une matrice ET programmable et d'une matrice OU figée. De même, il est peu probable d'utiliser tous les termes produits et on peut alors limiter le nombre d'entrées de la fonction OU. C'est le principe utilisé par les circuits programmable, appelés au début PAL (Programmable Array Logic) comme l'indique la figure I-1, mais plus communément désigné aujourd'hui sous le terme PLD (Programmable Logic Device) [4]. Un Programmable Logic Array (PLA ou FPLA : Field Programmable Logic Array) offre le maximum de souplesse car les deux matrices des OU et ET sont programmables comme l'indique la figure I-2 représente une matrice PLA à 4 entrées et 4 sorties. N'importe quelle fonction peut être codée par une somme de produit, par un produit de somme ou un mélange des deux. On peut immédiatement en déduire une structure de circuits, appelé matrice PLA (Programmable Logic Array). Chacune des 4 entrées et son complémentaire arrive sur une des 24=16 portes ET à 2x4=8 entrées. Afin de simplifier la représentation, les 8 lignes ont été représentées par une seule, chaque croix représentant une connexion programmable (un fusible par exemple). Ce type de structure est utilisé dans certains circuits ASIC (Application Spécific Integrated Circuit) et demande une densité d'intégration importante : en effet pour n variables en entrées, il faut 2n fonctions ET à 2n entrées et au moins un OU à 2n entrées (il y a en effet 2n combinaisons possibles, chaque combinaison dépendant de l'entrée et de son complémentaire) [32].

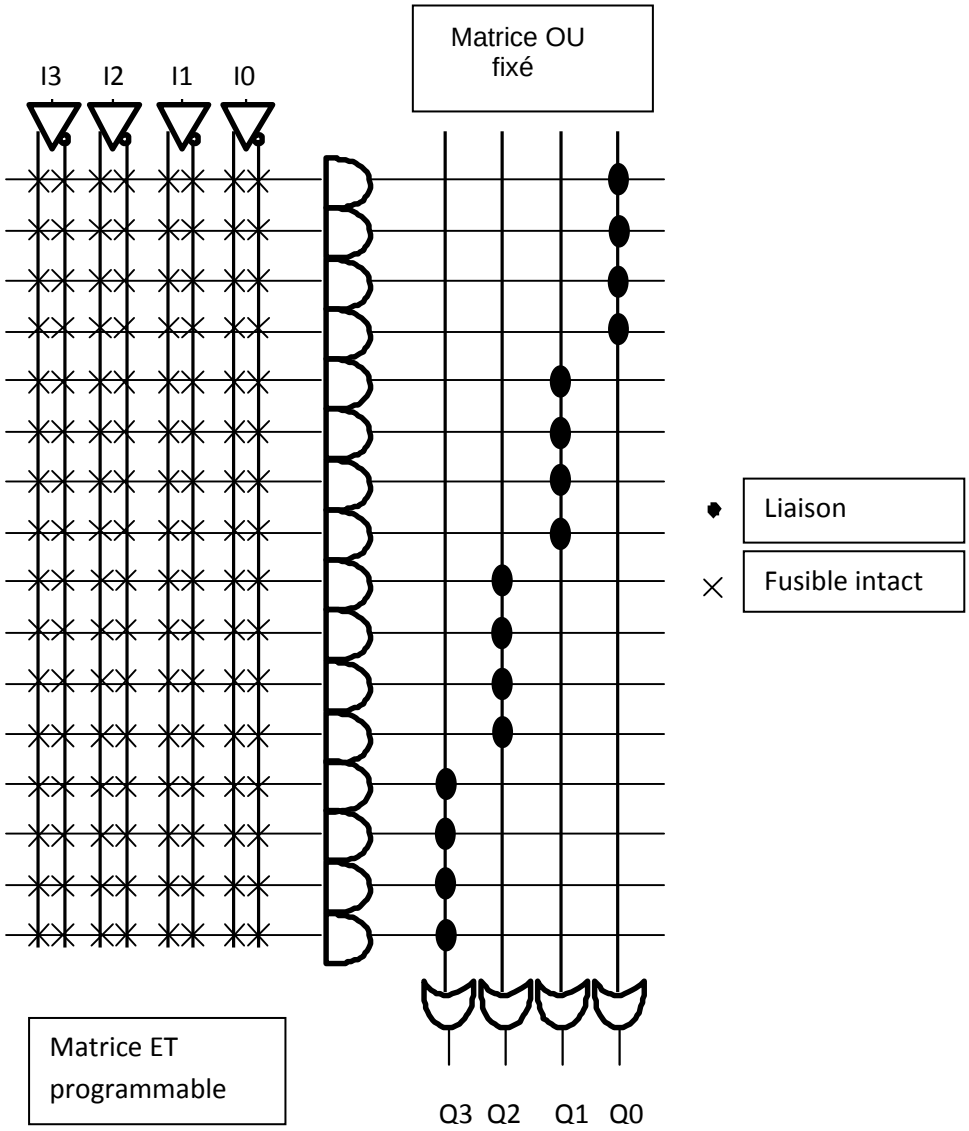


Figure I-2 : Propose le principe de réalisation des fonctions de la matrice ET PAL.

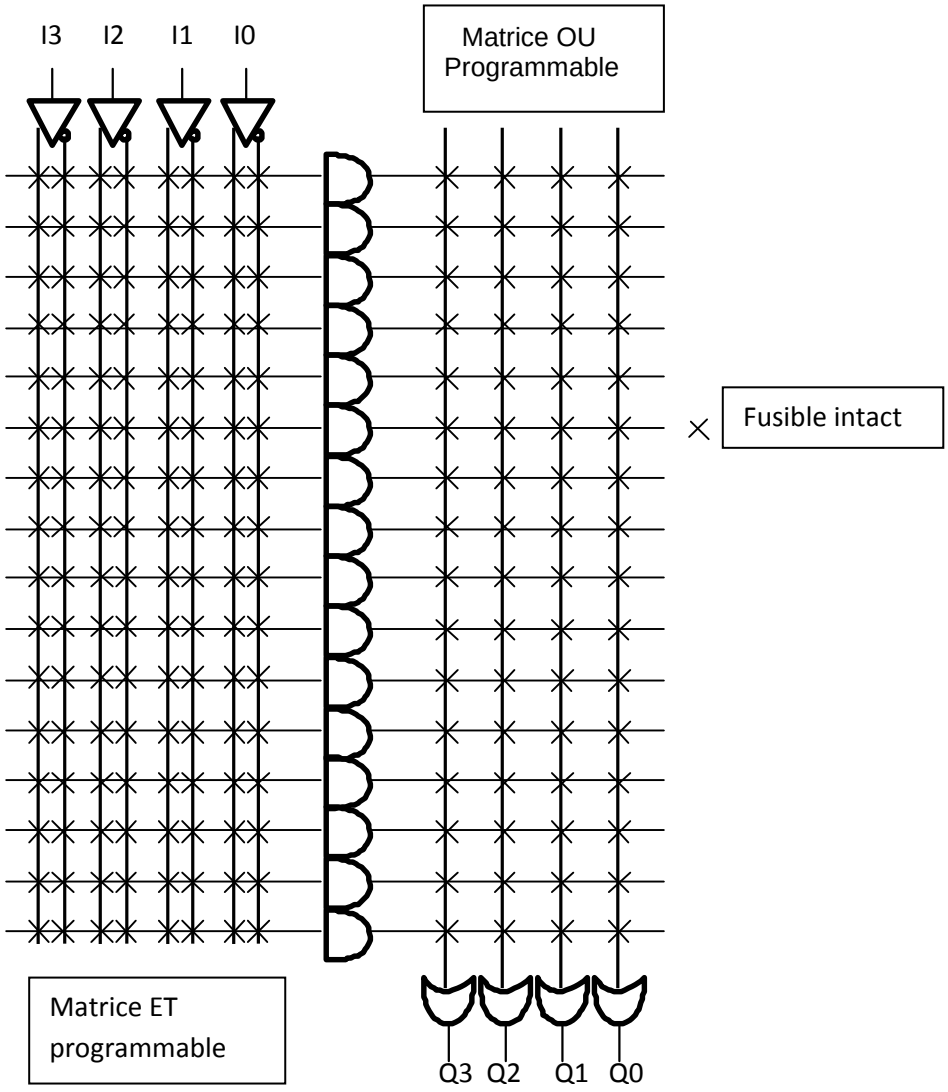


Figure I-3 : le principe de réalisation des fonctions de la matrice ET PLA.

Ces circuits constituent la famille de SPLDs (Simple Programmable Logic Device). Ce type de circuits n'a cessé de se sophistiquer au fil du temps. Certains canaux d'entrée/sortie peuvent être bidirectionnels, les deux portes "3 états" en commandant le sens pouvant être programmable. Par ailleurs, des sorties peuvent être injectées dans le réseau de portes ET, offrant ainsi la possibilité d'utiliser des variables intermédiaires. On peut également trouver un réseau de bascules de type D avec une horloge commune.

L'entrée d'une bascule peut être programmée au niveau des réseaux de portes ET et OU. Sa sortie peut être accessible à l'extérieur et utilisée comme variable interne injectée dans le réseau de connexions programmables. Il est alors possible de programmer des fonctions séquentielles. La figure I.3 illustre quelques unes de ces possibilités. On y trouve une bascule

D dont la sortie est injectée dans le réseau programmable, matérialisé par les petits carrés. La sortie doit être validée par le signal OE (Output Enable).

L'évolution a conduit à des composants plus complexes. Ce sont d'abord les CPLDs (Complex Programmable Logic Device), puis les FPGA (Field Programmable Gate Array). Ce sont de relativement de gros ensembles de cellules logiques programmables (comparable à ce que nous venons de présenter) qui peuvent être connectées de différentes matières.

Un CPLD est composé d'un certain nombre de PALs, ou macrocellule. La figure I-4 donne un exemple de ce que peut être une macrocellule. Ces macrocellules sont regroupées en blocs logiques, qui peuvent être reliés via une matrice de connexion programmable (fig. I-5). Les macrocellules d'un même bloc sont généralement interconnectées [33].

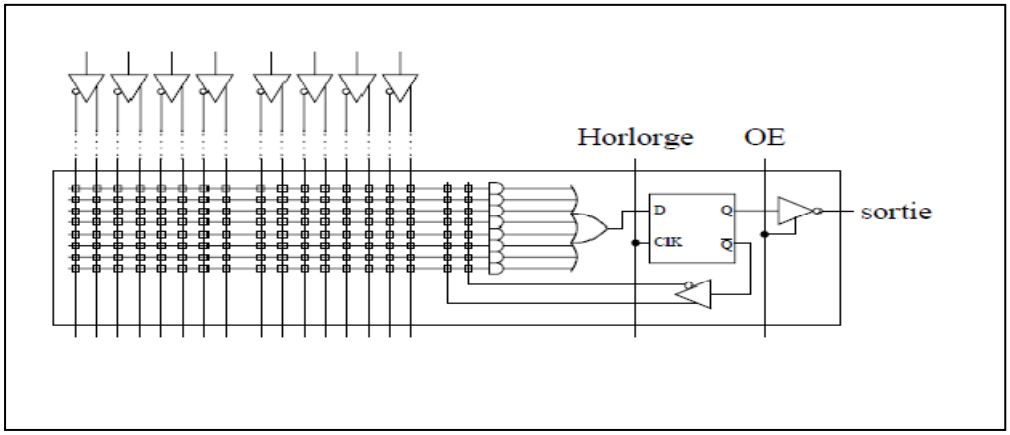


Figure I-4 : Macrocellule [35].

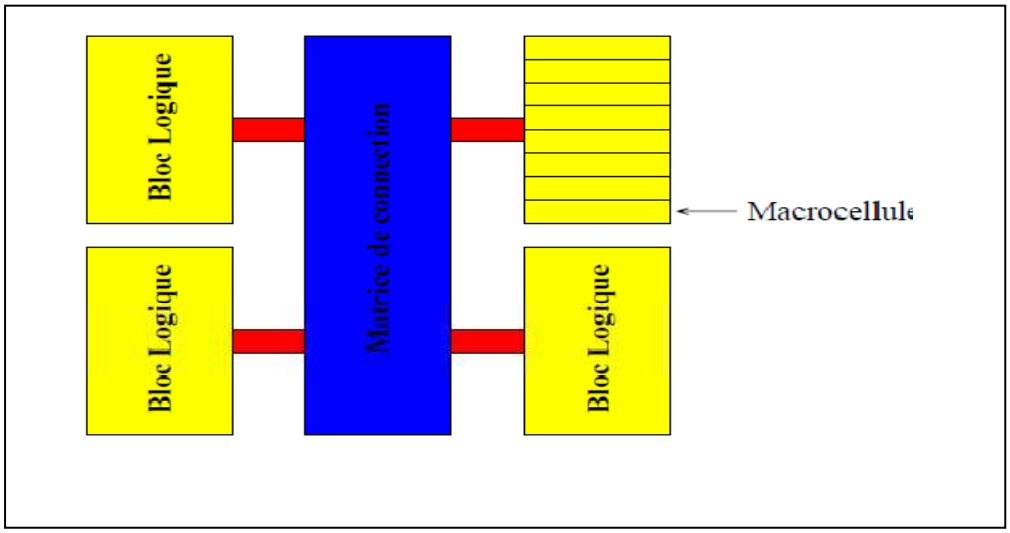


Figure I-5 : Matrice de connexion programmable [35].

I.3.2. Mémoires

Une fonction combinatoire associe à chacune de ces combinaisons d'entrée une valeur en sortie décrite par sa table de vérité. C'est le principe de la mémoire où pour chaque adresse en entrée, on associe une valeur en sortie, sur un ou plusieurs bits. La structure physique des mémoires fait appelle à une matrice PLA dont la matrice ET est figée et sert de décodeur d'adresse et dont la matrice OU est programmée en fonction de la sortie désirée (figure I-6).

Lorsqu'une adresse est présentée, par exemple I_1 I_2 I_3 I_4 = 1111, la porte ET concernée passe au NL1 (celle du bas dans l'exemple) et suivant les fusibles laissés intacts sur la matrice OU, on a un mot différent en O_0 O_1 O_2 O_3 (0000 si tous les fusibles sont "grillés" par exemple). Ce principe utilisé pour les mémoires, l'est aussi dans les CPLD (Complex Programmable Logic Device), mais surtout dans les FPGA (Field Programmable Gate Array) sous le nom de LUT (Lock UpTable) [32].

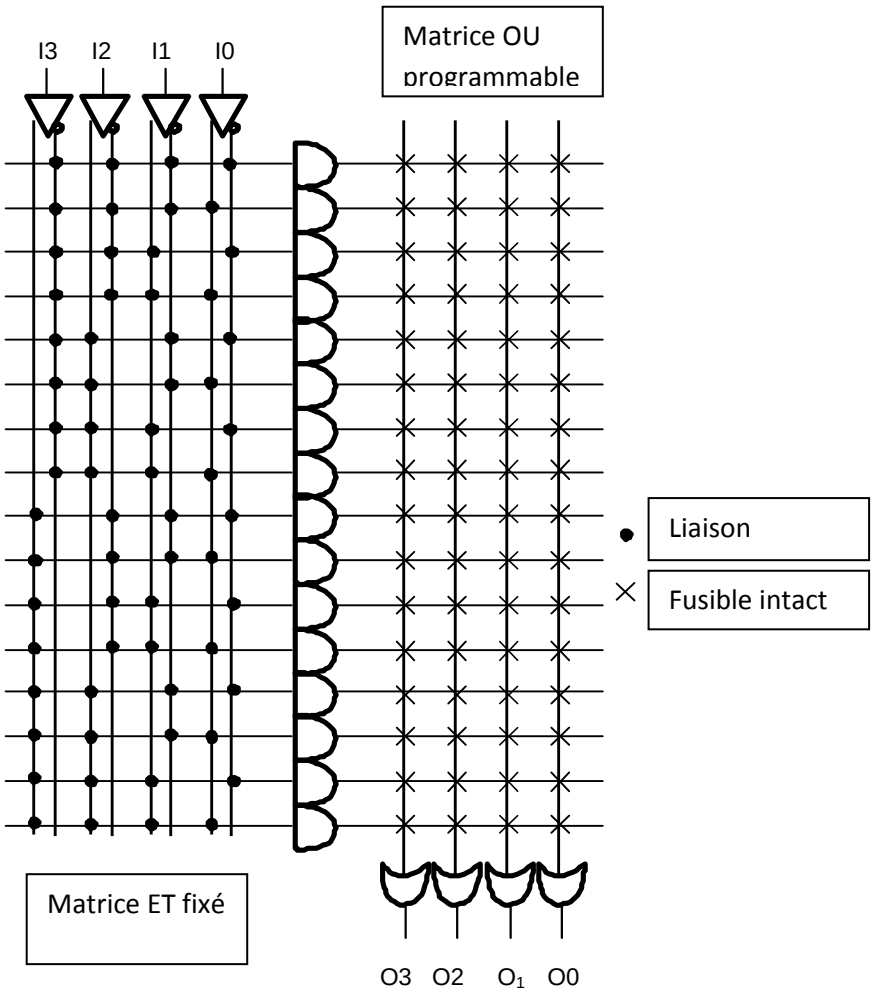


Figure I-6 : Représentation symbolique simplifiée d'une PROM de 16 mots de 4 bits.

I.3.3. Multiplexeur

Le multiplexeur permet également de coder une fonction combinatoire, comme le montre la figure suivante (figure I-7).

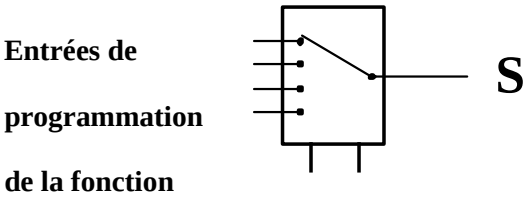


Figure I-7 : Représentation symbolique de Multiplixeur

A chaque valeur des entrées E_0 et E_1 est associé un niveau logique défini dans la table de vérité pour la sortie S. Ce niveau logique est imposé sur l'entrée de programmation correspondante. Ce principe est utilisé dans les FPGA.

Un multiplexeur est un aiguillage d'informations. Dans sa forme la plus simple, il comporte deux entrées de données, une sortie et une entrée de sélection, conformément au symbole de la figure I-8.

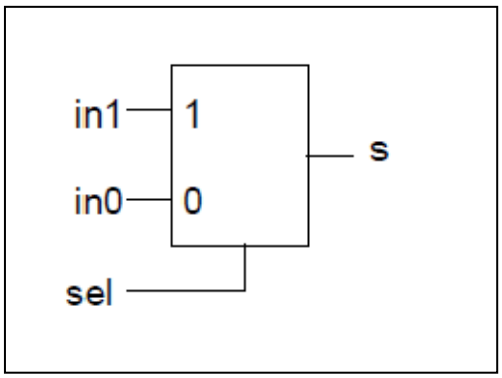


Figure I-8 : Un multiplexeur élémentaire [35].

Le fonctionnement de cet opérateur se décrit très simplement sous une forme algorithmique :

Si sel = '0' s ← in0 ;

Si non (sel = '1') s ← in1 ;

Certains constructeurs notent sur le symbole, comme nous l'avons fait, la valeur de l'entrée de sélection en regard de l'entrée correspondante.

La première utilisation des multiplexeurs dans les circuits programmables est, évidemment, de créer des chemins de données. La programmation consiste alors à fixer des valeurs aux entrées de sélection.
Une autre utilisation de la même fonction consiste à remarquer qu'un multiplexeur est, en soi, un opérateur générique. Reprenant l'exemple précédent du *ou exclusif*, on peut le décrire sous forme algorithmique :
si e1= '0' e1 $\oplus$ e2 $\leftarrow$ e2 ;

si non (e1 = '1') e1 $\oplus$ e2 $\leftarrow$ e2 ;

D'où une réalisation possible de l'opérateur ou exclusif au moyen d'un multiplexeur dans le schéma de la figure I-9.

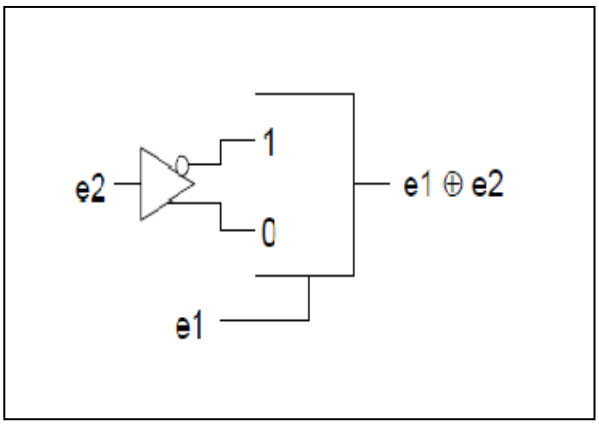


Figure I-9 : Ou exclusif réalisé par un multiplexeur [35].

L'exemple précédent peut être généralisé sans peine : un multiplexeur à n entrées de sélection, soit 2n entrées de données, permet de réaliser n'importe quelle fonction combinatoire de n + 1 entrées, pourvu que l'une, au moins, de ces entrées existe sous forme directe et sous forme complémentée.

Dans un circuit programmable dont les « briques » de base sont des multiplexeurs (c'est le cas de beaucoup de FPGAs) la programmation consiste à fixer des chemins de données, c'est à dire à établir des interconnexions entre des cellules de calcul et des signaux d'entrée et de

sortie. Cette opération de création d'interconnexions entre des cellules génériques s'appelle le *routage* d'un circuit ; l'affectation des cellules à des fonctions souhaitées par l'utilisateur s'appelle le *placement* [35].

I.3.4. Ou exclusif

L'opérateur élémentaire ou exclusif, ou somme modulo 2, dont nous avons rappelé l'expression algébrique précédemment, est disponible en tant que tel dans certains circuits.

Cet opérateur intervient naturellement dans de nombreuses fonctions combinatoires reliées de près ou de loin à l'arithmétique : additions et soustractions, contrôles d'erreurs, cryptages en tout genre, etc. Or ces fonctions se prêtent mal à une représentation en somme de produits, car elles ne conduisent à aucune minimisation de leurs équations. Un simple générateur de parité sur 8 bits, qui rajoute un bit de parité à un octet de données, ne nécessite pas moins de 128 (28-1) produits, quand il est « mis à plat », pour être réalisé au moyen d'une couche de ETs et d'une couche de OUs. De nombreuses familles de circuits programmables disposent, en plus des PLAs, d'opérateurs ou exclusifs pour faciliter la réalisation de ces fonctions arithmétiques.

Une autre application de l'opérateur ou exclusif est la programmation de la polarité d'une expression. Quand on calcule une fonction combinatoire quelconque sous forme disjonctive, il peut arriver qu'il soit plus économique, en nombre de produits nécessaires, de calculer le complément de la fonction et de complémenter le résultat obtenu. Par exemple, si (cba)2 représente l'écriture en base 2 d'un nombre N, compris entre 0 et 7, on peut exprimer par une équation logique que N est premier avec 3 :

$$prem3 = c * \bar{b} + \bar{b} * a + c * a + \bar{c} * b * \bar{a}$$

On peut également remarquer que si N est premier avec 3 c'est qu'il n'est pas multiple de 3, soit :

$$prem3 = \overline{mul3} = \overline{\bar{c} * \bar{b} * \bar{a} + \bar{c} * b * a + c * b * \bar{a}}$$

La deuxième forme, apparemment plus complexe, nécessite un produit de moins que la première pour sa réalisation. Dans des circuits où le nombre de produits disponibles est limité,

cela peut présenter un avantage. Un opérateur ou exclusif permet de passer, par programmation, d'une expression à son complément, comme l'indique la figure I-10. Comme cet opérateur peut être réalisé avec un multiplexeur, l'une ou l'autre de ces formes peut se trouver dans les notices ! [35].

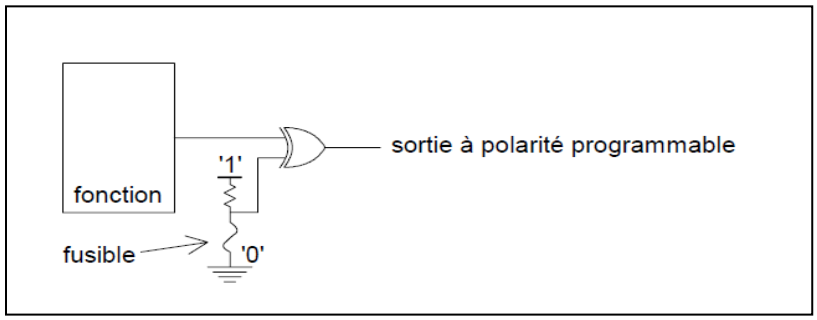


Figure I-10 : Polarité programmée par un ou exclusif [34].

I.3.5. Les Bascules

Qui dit logique dit logique séquentielle. Les circuits programmables actuels offrent tous la possibilité de créer des fonctions séquentielles, synchrones dans leur immense majorité. La brique de base de toute fonction séquentielle est la bascule, cellule mémoire élémentaire susceptible de changer d'état quand survient un front actif de son signal d'horloge.

Bascule D, T ou J-K ? La première est toujours présente. Comme certaines fonctions se réalisent plus simplement avec la seconde, les compteurs par exemple, de nombreux circuits permettent, toujours par programmation, de choisir entre bascule D et bascule T, voire entre l'un des trois types de base. La figure I-11 rappelle, par un diagramme de transitions, le fonctionnement de ces trois types de bascules.

Le programmeur n'a, en réalité, que rarement à se préoccuper de ce genre de choix, les optimiseurs déterminent automatiquement le type de bascule le mieux adapté à l'application [35].

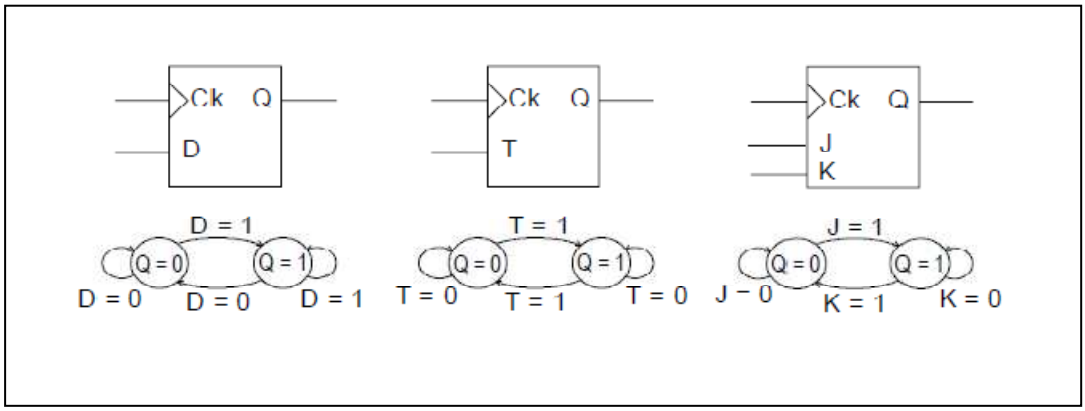


Figure I-11 : Les trois types fondamentaux de bascules [34].

I.4. Technologie d'interconnexions

Premier critère de choix d'un circuit programmable, la technologie utilisée pour matérialiser les interconnexions détermine les aspects électriques de la programmation : maintien (ou non) de la fonction programmée en l'absence d'alimentation, possibilité (ou non) de modifier la fonction programmée, nécessité (ou non) d'utiliser un appareil spécial (un programmateur, bien sûr) [36].

Comme nous venons de le voir, l'un des éléments clé des circuits étudié est la connexion programmable. Du choix d'une technologie dépendra essentiellement :

- la densité d'intégration
- la rapidité de fonctionnement une fois le composant programmé, fonction de la résistance à l'état passant et des capacités parasites
- la facilité de mise en œuvre (programmation sur site, reprogrammation etc.)
- la possibilité de maintien de l'information.

Passons en revue quelques technologies classiquement utilisées et leurs caractéristiques. Ces technologies sont ou pourraient être utilisées pour la réalisation de mémoires. Il faut cependant garder à l'esprit qu'hormis quelques cas particuliers (circuit reprogrammés en cours d'utilisation), le temps d'écriture reste secondaire, le circuit étant habituellement programmé une fois pour toutes avant utilisation [32].

I.4.1. Connexions programmable une seule fois (OTP : One Time Programming)

I.4.1.1. Cellules à fusible

La programmation consiste à détruire ou à garder intact des fusibles, de manière à supprimer ou à conserver des connections (figure I-12) :

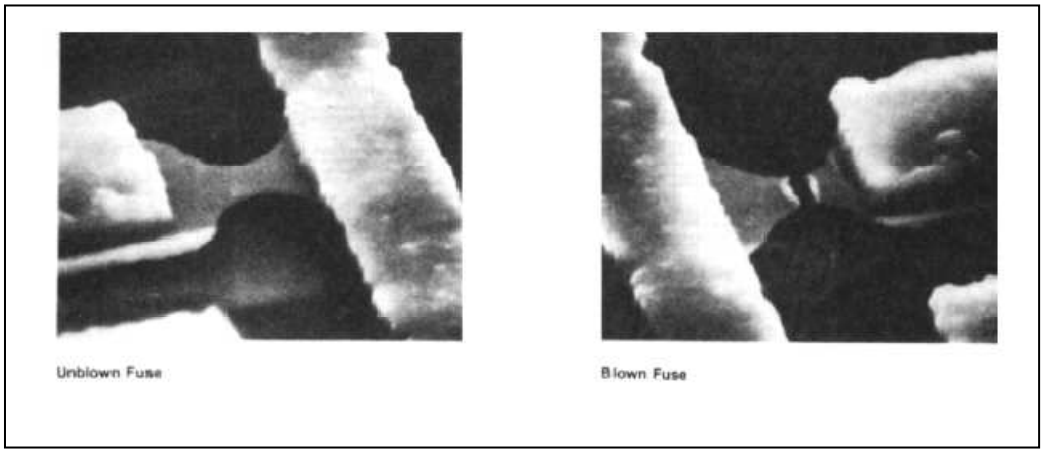


Figure I-12 **:** La programmation physique d'un fusible [37].

Remarque : la destruction d'un fusible entraîne la mise à 1 de l'entrée de la porte concernée [37].

Ce sont les premières à avoir été utilisées et elles ont aujourd'hui disparu au profit de technologies plus performantes. Leur principe consistait à détruire un fusible conducteur par passage d'un courant fourni par une tension supérieure à l'alimentation (12 à 25 V). Avec cette technique la programmation est irréversible [32].

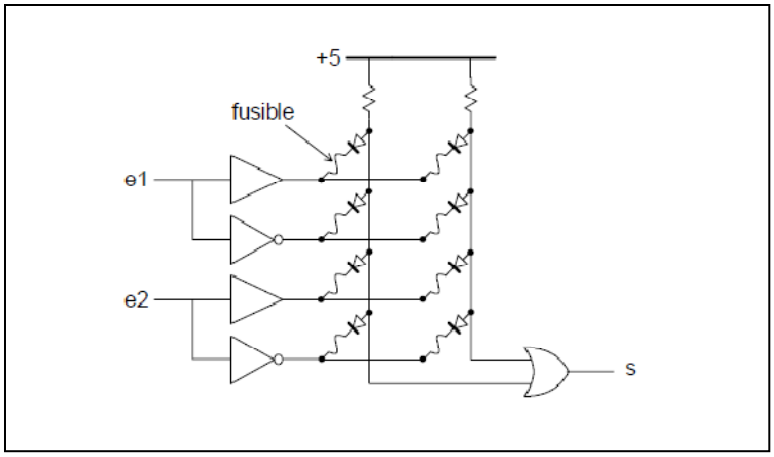


Figure I-13 : Pld élémentaire à fusibles [35].

La figure I-13 en illustre le principe, toutes les connexions sont établies à la fabrication. Lors de la programmation le circuit est placé dans un mode particulier par le programmateur, mode dans lequel des impulsions de courant sont aiguillées successivement vers les fusibles à détruire.

Pour programmer un circuit, il faut transférer dans le programmateur une table qui indique par un chiffre binaire l'état de chaque fusible (la table des fusibles). Cette table est généralement transférée entre le système de CAO et le programmateur sous forme d'un fichier au format normalisé le format JEDEC.

L'exemple ci-dessous est un extrait du fichier JEDEC, généré par un compilateur VHDL, qui implémente un compteur binaire 10 bits dans un circuit de type 22V10 :

```
C22V10*
QP24*                    Number of Pins*
QF5828*                  Number of Fuses*
F0*                      Note: Default fuse setting 0*
G0*                      Note: Security bit Unprogrammed*
NOTE DEVICE C22V10*
NOTE PACKAGE PAL22V10G-5PC*
NOTE PINS hor:1 oe:2 en:3 raz:4 compte_6:14 compte_8:15 compte_9:16
compte_3:17 *
NOTE PINS compte_1:18 compte_0:19 compte_2:20 compte_4:21 compte_7:22 *
NOTE PINS compte_5:23 *
NOTE NODES *
L00000
00000000000000000000000000000000000000000000
* Node hor[1] => BANK : 1 *

L00044
11110111111111111111111111111111111111111111
11011111011010101110111011101111111111111111
11101111110110111111111111111111111111111111

11101111111110111111111111011111111111111111
11101111111110011111111111111111111111111111
11101111111110111111111011111111111111111111
11101111111110111101111111111111111111111111
11101111101110111111111111111111111111111111
00000000000000000000000000000000000000000000
* Node compte_5[23] => OE : 1 ,LOGIC : 8 *
L00440
11110111111111111111111111111111111111111111
...
C72EB*                   Note: Fuse Checksum*
9604
```

Figure I-14 : un extrait du fichier JEDEC [34].

Pour chacun des 5828 fusibles de ce circuit, un '0' indique un fusible intact, un '1' un fusible programmé.

L'examen du début de la table précédente met en évidence un défaut majeur de cette technologie : la programmation détruit plus de fusibles qu'elle n'en conserve, et de loin. Cela se traduit par une mauvaise utilisation du silicium, un temps de programmation important (quelques secondes) et des contraintes thermiques sévères imposées au circuit lors de l'opération. Cette technologie n'est donc pas généralisable à des circuits dépassant quelques centaines de portes équivalentes.

Le lecteur averti aura peut-être remarqué, à la lecture de l'en-tête du fichier JEDEC, qu'en réalité le circuit précédent ne contient aucun fusible. Il s'agit en vérité d'un circuit CMOS à grille flottante, mais l'ancienne terminologie est restée [34].

I.4.1.2. Cellules à antifusible

L'inverse d'un fusible est un anti-fusible. Le principe est, à l'échelle microscopique, celui de la soudure électrique par points.

En appliquant une tension importante (16 V pendant 1 ms) à un isolant entre deux zones de semi-conducteur fortement dopées, ce dernier diffuse dans l'isolant et le rend conducteur. Cette technologie très en vogue permet une haute densité d'intégration.

Hormis le non reprogrammabilité, c'est la meilleure technologie (vitesse et surtout densité d'intégration).

Dans la première (fig. I-15) on fait fondre un diélectrique en forçant un courant important (environ 5 mA) dans une zone de très faibles dimensions située à l'intersection de deux électrodes l'une en silicium polycristallin et l'autre réalisée par un implant fortement dopé n+. Ce diélectrique est constitué de trois couches alternées d'oxyde et de nitrure de silicium : SiO2 - Si3N4 - SiO2. La présence du nitrure permet de réduire l'épaisseur du diélectrique. Lors de la fusion du diélectrique il y a diffusion de dopants, ce qui diminue la résistance du contact réalisé.

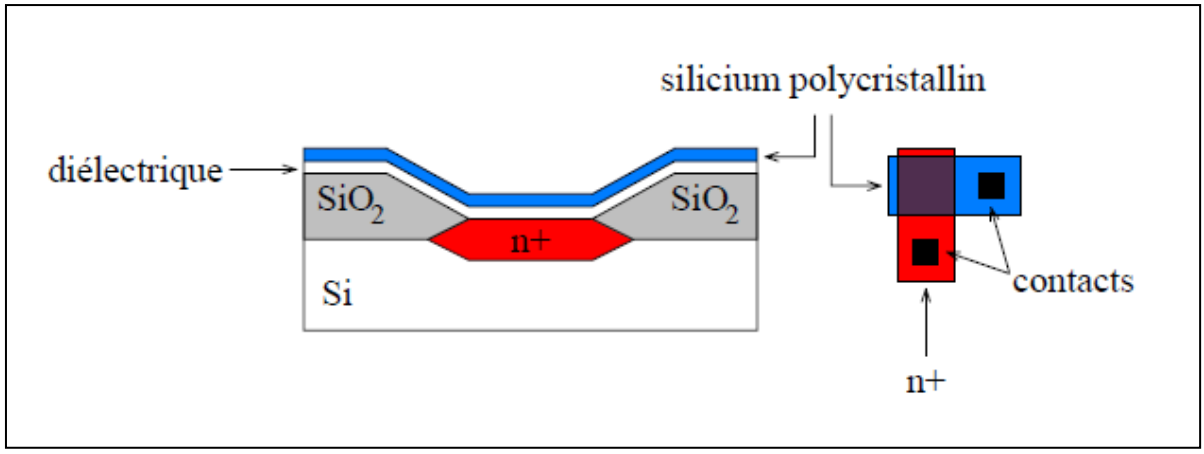


Figure I-15 : Première technologie de programmation cellules a antifusible [33].

La seconde technologie (fig. I-16) permet de réduire cette résistance, car le contact est réalisé par un alliage de tungstène, titane et silicium. Le diélectrique est cette fois du silicium amorphe entre deux électrodes métalliques (m1 et m2). Cette technique requiert moins de place que la première, mais elle est assez délicate à mettre en œuvre.

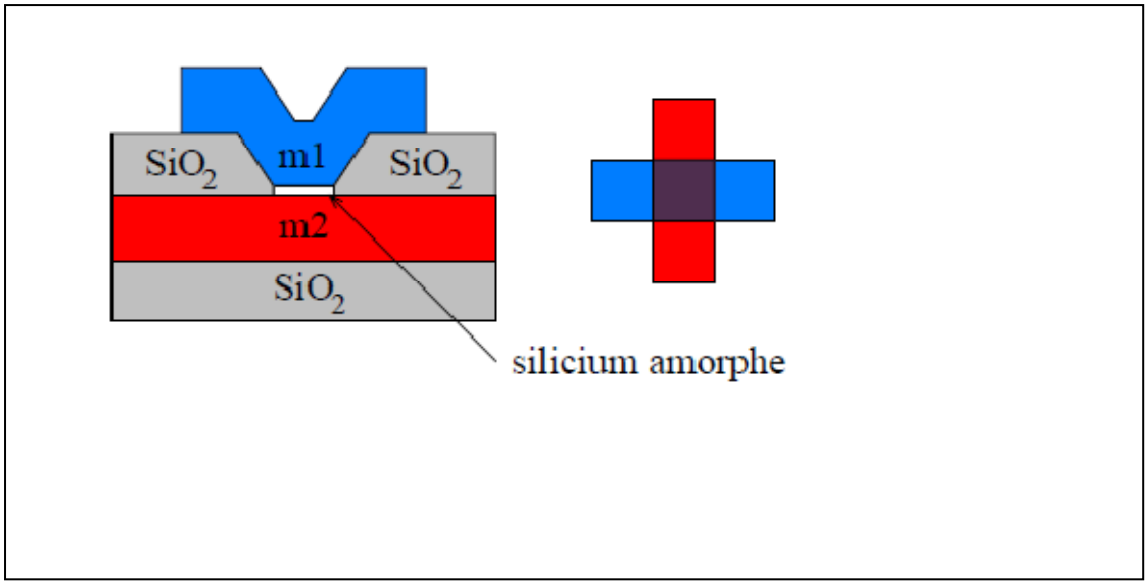


Figure I-16 : Seconde technologie de programmation cellules à antifusible [33].

Dans toutes les techniques précédentes la programmation irréversible, il est impossible de revenir en arrière [32,33,36].

I.4.2. Cellules reprogrammables

I.4.2.1. Cellule à transistor MOS à grille flottante et EPROM (Erasable Programmable Read Only Mémory)

L'apparition du transistor MOS à grille flottante a permis de rendre le composant bloqué ou passant sans application permanente d'une tension de commande. Le principe consiste à piéger ou non (à l'aide d'une tension supérieure à la tension habituelle d'alimentation) des électrons dans la grille. L'extraction éventuelle des électrons piégés permet le retour à l'état initial.

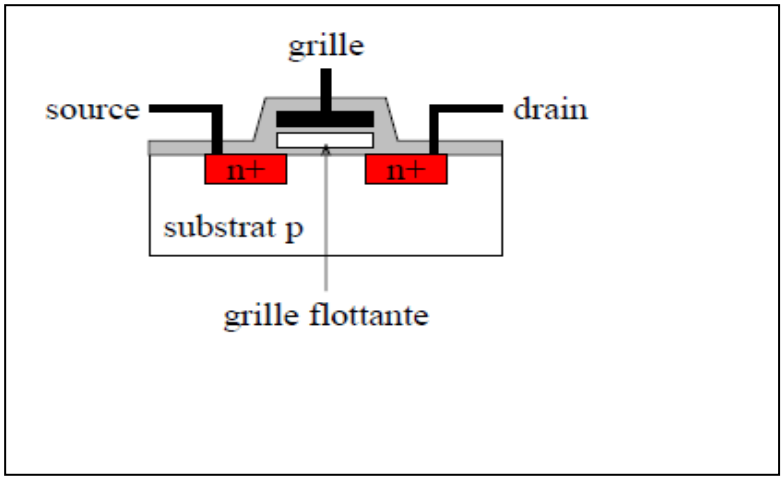


Figure I-17 : La structure des transistors Mos à grille flottante [33].

On utilise pour cela des transistors à effet de champ à structure MOS (Métal-Oxyde- Semi-conducteur) avec une grille supplémentaire flottante. Dans un transistor MOS classique la grille est utilisée pour induire un canal entre la source et le drain.

Dans l'exemple de la figure I-17 une tension positive de la grille va attirer des électrons et repousser des trous. Par ailleurs une partie des électrons attirés se recombinent avec des trous.

Une inversion de population se crée induisant un canal n entre les implants de la source et du drain. Le transistor devient passant. Pour programmer un tel nœud on fait alors circuler un courant intense entre la source et le drain. Certains électrons acquièrent une énergie leur permettant d'atteindre la grille flottante. Ils y sont alors piégés. Lorsque la charge piégée est suffisante elle masque le champ électrique induit par la grille et le transistor est bloqué. Le courant de fuite étant très faible cette charge peut se conserver très longtemps.

La programmation est réversible, il suffit de décharger cette grille flottante. Une première solution consiste à exposer le circuit à un rayonnement ultraviolet pendant quelques dizaines

de minutes. Ces circuits sont équipés d'une fenêtre en quartz et encapsulés dans un boîtier en céramique pour résister à l'échauffement. Cette mise en œuvre augmente le prix des composants, mais cela n'est nécessaire que pour le développement. Pour la production le même circuit existe sans fenêtre et en boîtier plastique. Cette technique correspond aux mémoires mortes effaçables :

EPROM (Erasable Programmable Read Only Memory). Il est également possible de décharger la grille flottante par effet tunnel en appliquant des tensions suffisamment élevées entre la grille, la source et le drain. Cet effacement est plus rapide et n'impose pas de retirer le circuit du système dans lequel il est installé. Cela correspond à la famille des mémoires mortes programmables et effaçables électriquement : EEPROM (Electrically Erasable Programmable Read Only Memory).

La différence entre les EEPROM et les mémoires vives réside dans la vitesse d'écriture. Le cycle d'écriture d'une EEPROM est environ 1000 fois plus long que celui d'une RAM. Par Contre rappelons que les temps d'accès en lecture des mémoires RAM, ROM, PROM, EPROM et EEPROM sont comparables.

Un composant hybride qui associe dans un même boîtier une RAM et une EEPROM de même capacité : la mémoire NOVRAM (NOn Volatile RAM). L'EEPROM permet de réaliser en moins de 10 ms une sauvegarde globale de la RAM. Cela permet une sauvegarde du contenu de la mémoire en cas de coupure d'alimentation électrique.

- **Flash EPROM**

L'utilisation de deux transistors par cellule uniquement (5 pour l'EEPROM) et une structure verticale permettent une densité d'intégration importante (25 µm2 par cellules en CMOS 0,6 µm) trois à quatre fois plus importante que l'EEPROM, mais quand même 10 fois moins que la technologie à antifusible. Le nombre de cycle d'écriture (104 à 106) est également plus grand que pour l'EEPROM car l'épaisseur de l'isolant est plus importante.

Par contre, la simplicité de la cellule élémentaire n'autorise pas une reprogrammation sélective (éventuellement par secteur), ce qui n'est pas gênant pour le type de circuits qui nous intéresse. La tension de programmation et d'effacement est de 12 V, avec un temps de programmation de quelques dizaines de µs pour un temps d'effacement de quelques millisecondes.

Un des inconvénients des cellules flash et EEPROM de nécessiter une alimentation supplémentaire pour la programmation et l'effacement est pallié par les constructeurs en intégrant dans le circuit un système à pompe de charge fournissant cette alimentation. Le composant peut alors être programmé directement sur la carte où il est utilisé. On parle alors de composants ISP : In Situ Programmation ou encore suivant les sources, In System Programmation (fig. I-18) [32,33,34].

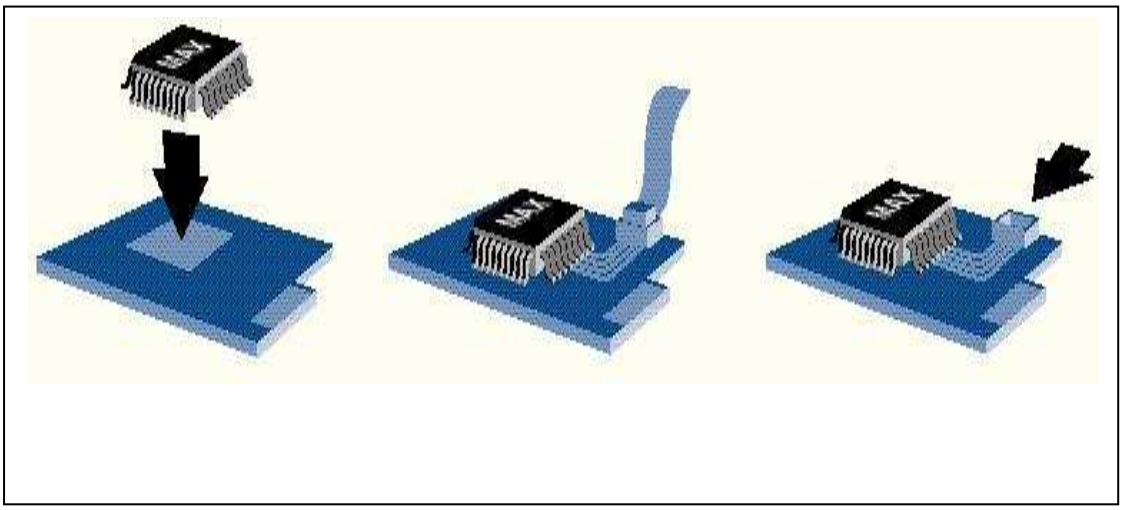


Figure I-18 : Le système de programmation ISP (In Sito Programmation) [32].

I.4.2.2. Cellules SRAM à transistors MOS classique

Dans les circuits précédents, la programmation de l'état des interrupteurs, conservée en l'absence de tension d'alimentation, fait appel à un mode de fonctionnement électrique particulier. Dans les technologies à mémoire statique, l'état de chaque interrupteur est commandé par une cellule mémoire classique à quatre transistors (plus un transistor de programmation), dont le schéma de principe est celui de la figure I-19.

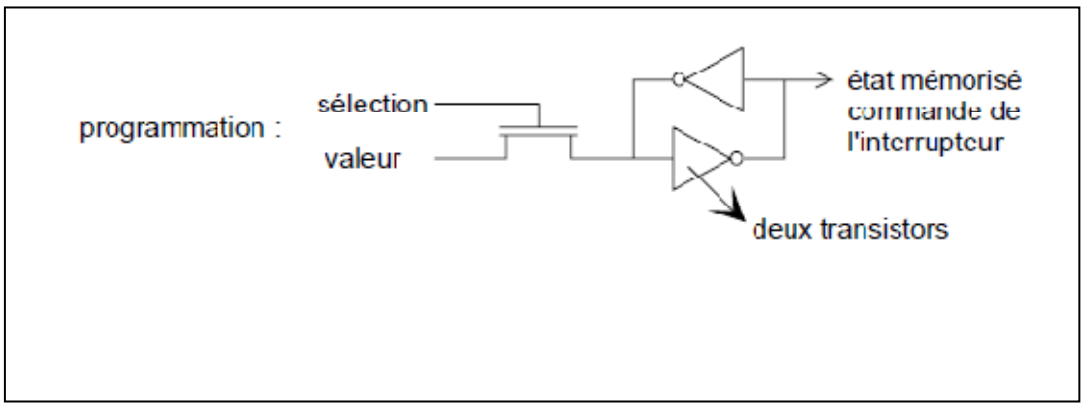


Figure I-19 : Cellule SRAM [34].

Ce principe est classiquement choisi pour les FPGA.

Le fait d'utiliser une mémoire de type RAM (donc volatile) impose la recharge de la configuration à chaque mise sous tension : une PROM série mémorise généralement les données. Ce qui peut paraître un inconvénient devient un avantage si on considère l'aspect évolutif du système qui peut s'adapter à un environnement extérieur changeant et modifier sa configuration en fonction des besoins. On pourra d'autre part facilement intégrer de la mémoire RAM dans le circuit. Le choix d'une cellule SRAM (Static Random Access Memory) à 6 transistors permet de bénéficier d'un accès sélectif et rapide (quelques ns) en cours d'utilisation.

La taille d'une cellule n'est que deux fois plus forte (50 µm2 par cellule) qu'avec une flash EEPROM. Cette technologie, utilisée pour les autres circuits VLSI (contrairement aux EEPROM et Flash EPROM et leurs transistors à grille flottante) permet de bénéficier directement des progrès importants réalisés dans ce domaine [32,34].

I.5. Architectures utilisées

I.5.1. PLD (Programmable Logic Device)

Comme nous l'avons vu, d'abord appelés PAL lors de sa sortie, ce circuit utilise le principe de la matrice PLA à réseau ET programmable. Bien que pas très anciens pour les dernières générations, les PLD ne sont presque plus utilisés pour une nouvelle conception. L'un de leur avantage qu'était la rapidité a disparu, les efforts de recherche des constructeurs portant plutôt sur les circuits à plus forte densité d'intégration que sont les CPLD et les FPGA.

Le fait que les PLD soient à la base de la conception des CPLD, très en vogue aujourd'hui, justifie cependant leur étude.

Initialement bipolaire, les cellules de connexions sont aujourd'hui réalisées en technologie MOS à grille flottante. La structure de base comprend un circuit PLA dont seule la matrice ET est programmable.

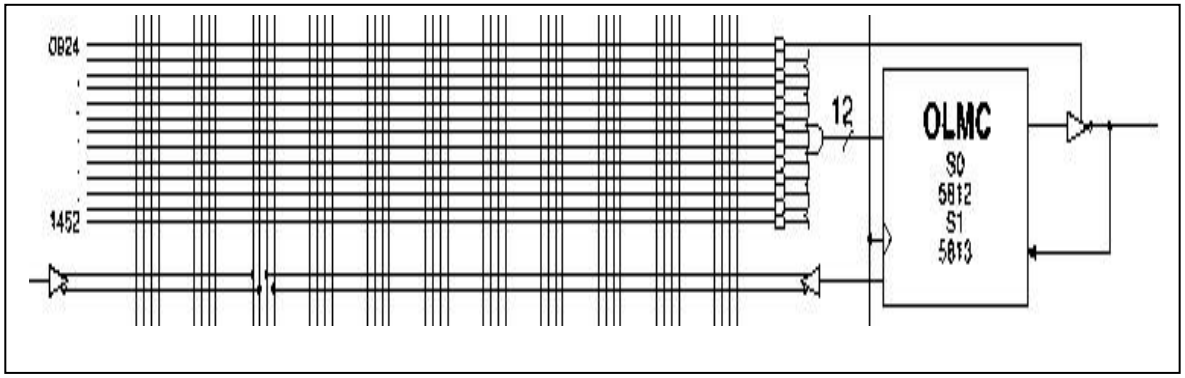


Figure I-20 : La structure générale d'un PLD [32].

La partie nommée OLMC (Output Logic MacroCell, dénomination Lattice) sur la figure I-20 peut être :

- combinatoire, une simple connexion relie alors la sortie du OU à l'entrée du buffer de sortie, dont la sortie est réinjectée sur le réseau programmable ;
- séquentielle, le bloc OLMC étant alors une simple bascule D ;
- versatile, il est alors possible par programmation de choisir entre les deux configurations précédentes.

Les PLD de dernière génération utilisent des OLMC versatiles, dont on donne ci-après la structure (fig. I-21) :

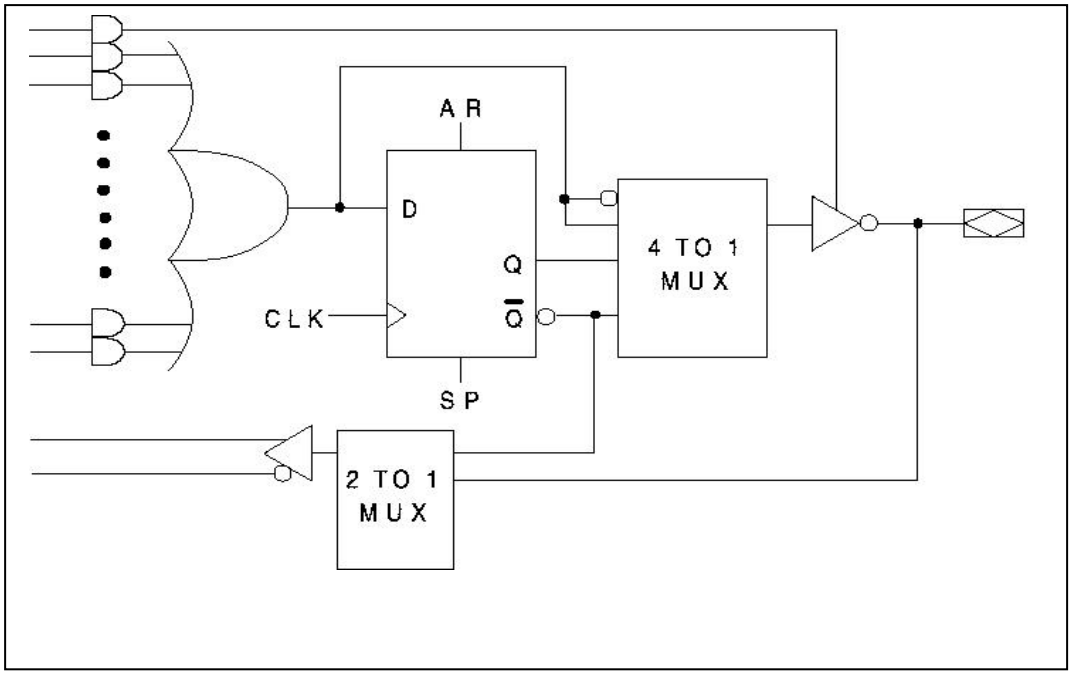


Figure I-21 : Structure interne d'une OLMC [32].

Le multiplexeur 4 vers 1 permet de mettre en circuit ou non la bascule D, en inversant ou pas les signaux. Le multiplexeur 2 vers 1 permet de réinjecter soit la sortie, soit l'entrée du buffer de sortie vers le réseau programmable [32].

I.5.2. CPLD (Complex Programmable Logic Device)

La nécessité de placer de plus en plus de fonctions dans un même circuit a conduit tout naturellement à intégrer plusieurs PLD (blocs logiques) sur une même pastille, reliée entre eux par une matrice centrale. Sur la figure suivante chaque bloc LAB (Logic Array Block) de 16 macrocellules est l'équivalent d'un PLD à 16 OLMC (fig. I-22). Ils sont reliés entre eux par une matrice d'interconnexion (PIA pour Programmable Interconnect Array).

Un seul point de connexion relie entre eux les blocs logiques. Les temps de propagation d'un bloc à l'autre sont donc constants et prédictibles.

La phase de placement des différentes fonctions au sein des macrocellules n'est donc pas critique sur un CPLD, l'outil de synthèse regroupant au maximum les entrées sorties utilisant des ressources communes.

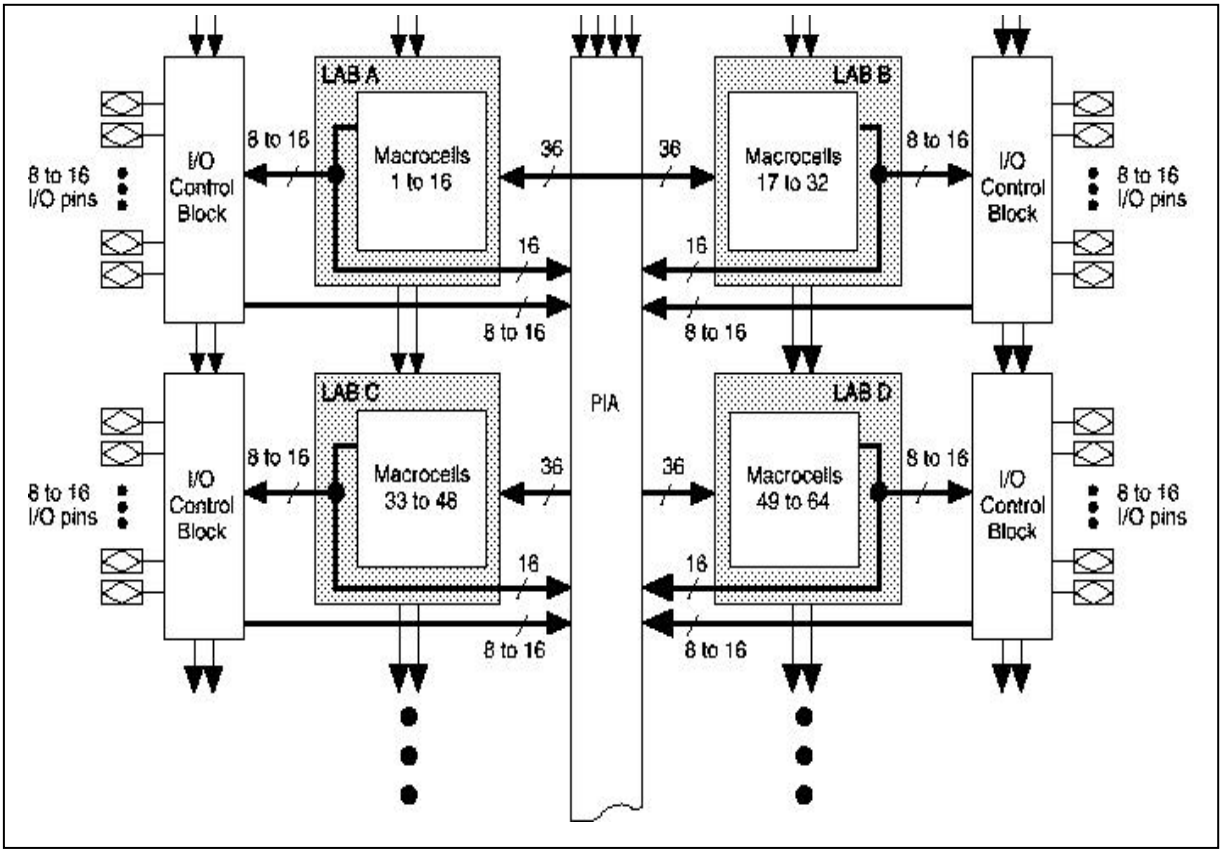


Figure I-22 : Architecture interne d'un CPLD [32].

L'établissement des liaisons (routage) entre les différentes macrocellules est encore moins critique :

Un seul point de connexion -cause du retard- relie les LAB entre eux. Le temps de propagation des signaux est parfaitement prédictible avant que le routage ne soit fait. Ce dernier n'influence donc pas les performances du circuit programmé.

Dans l'outil de synthèse, la partie s'occupant du placement et du routage est appelée le "fitter" (to fit : placer, garnir).

La technologie de connexion utilisée est généralement l'EEPROM (proche de celle des PLD) ou EEPROM flash [32].

I.5.3. FPGA (Field Programmable Gate Array)

Les blocs logiques sont plus nombreux et plus simple que pour les CPLD, mais cette fois les interconnexions entre les blocs logiques ne sont pas centralisées (fig. I-23).

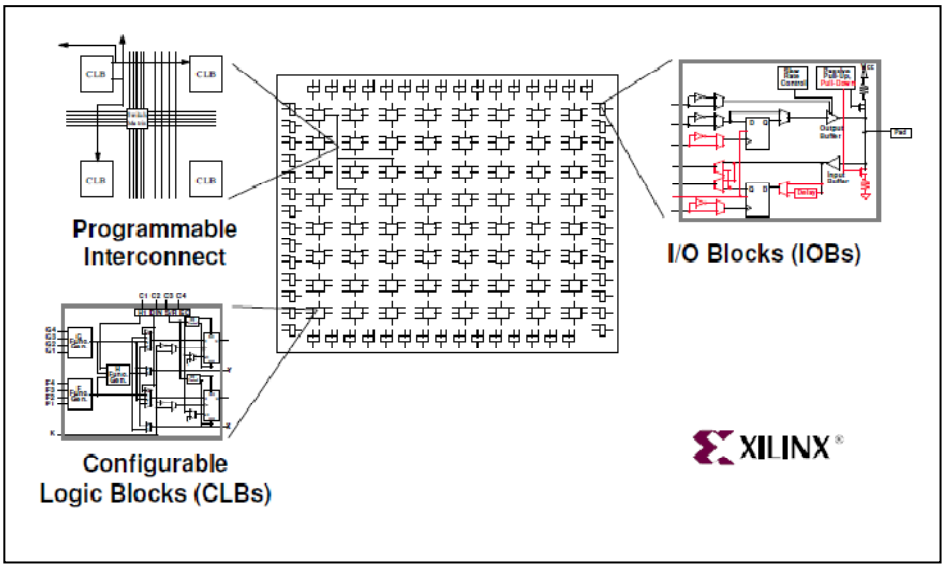


Figure I-23 : schéma d'une FPGA de constructeur XILINX [38].

Ces composants permettent une forte densité d'intégration. La petitesse des blocs logiques autorise une meilleure utilisation des ressources du composant (au prix d'un routage délicat) Il devient alors possible d'implanter dans le circuit des fonctions aussi complexes qu'un microcontrôleur. Ces fonctions sont fournies sous forme de programme par le constructeur du composant et appelées "megafunction" ou "megacore". Le terme générique classiquement utilisé pour les désigner est "propriété intellectuelle" ou IP (Intelectual Property).

Les FPGA utilisent généralement les technologies SRAM ou antifusible [32,38].

I.5.4. ASIC (Application Spécific Intégrated Circuit)

Si les composants précédents pouvaient être développés avec un simple ordinateur, ceux que nous abordons maintenant nécessitent l'intervention d'un fondeur qui produira le circuit demandé à partir des masques fournis par son client. Ici encore, le terme programmable n'est pas des plus judicieux, les connexions entre les éléments étant dessinées sur les masques.

Les temps et coûts de productions sont importants. On distingue trois types d'ASIC classé par ordre croissant de configurabilité.

I.5.4.1. Les prédiffusés (gate arrays)

Ils contiennent une nébuleuse de transistors ou de portes à interconnecter avec les problèmes de routage et de délais que cela comporte.

I.5.4.2. Les précactérisés (standard cell)

On utilise cette fois des bibliothèques de cellules standards à placer sur le semi-conducteur.

I.5.4.3. Les "fulls customs"

Ils sont entièrement définissables par le client. Ces circuits conduisent à la réalisation de tous les composants VLSI comme les microprocesseurs [32].

I.6. Conclusion

Dans le monde des circuits numériques les chiffres évoluent très vite, beaucoup plus vite que les concepts. Cette impression de mouvement permanent est accentuée par les effets d'annonce des fabricants et par l'usage systématique de la publicité comparative, très en vogue dans ce domaine.
Il semble que doivent se maintenir trois grandes familles :

− Les PLDs et CPLDs en technologie FLASH, utilisant une architecture somme de produits. La tendance est à la généralisation de la programmation in-situ, rendant inutiles les programmateurs sophistiqués. Réservés à des fonctions simples ou moyennement complexes, ces circuits sont rapides (jusqu'à environ 200 MHz) et leurs caractéristiques temporelles sont pratiquement indépendantes de la fonction réalisée. Les valeurs de fréquence maximum de fonctionnement de la notice sont directement applicables.

− Les FPGAs à SRAM, utilisant une architecture cellulaire. Proposés pratiquement par tous les fabricants, ils couvrent une gamme extrêmement large de produits, tant en densités qu'en

vitesses. Reprogrammables indéfiniment, ils sont devenus reconfigurables rapidement (200 ns par cellule), en totalité ou partiellement.

− Les FPGAs à antifusibles, utilisant une architecture cellulaire à granularité fine. Ces circuits stendent à remplacer une bonne partie des ASICs prédiffusés. Programmables une fois, ils présentent l'avantage d'une très grande routabilité, d'où une bonne occupation de la surface

du circuit. Leur configuration est absolument immuable et disponible sans aucun délai après la mise sous tension ; c'est un avantage parfois incontournable [36].

CHAPITRE II

Technologie et environnement de développement des FPGAs

II.1. INTRODUCTION

Les FPGA, sigle anglais qui signifie « Field Programmable Gates Arrays » traduit en français par réseau de portes programmables, sont des circuits intégrés reprogrammables. Ils offrent la possibilité de réaliser des fonctions numériques plus ou moins complexes, tout comme leurs homologues figés : les ASIC [40].

Les FPGA sont des circuits numériques matériels configurables dédiés à l'électronique numériques [41].

A l'état initial ils ne peuvent rien faire mais disposent d'une importante quantité (dépendant de la technologie utilisée) de ressources matérielles opérationnelles dont on peut configurer la fonction. Ces ressources sont, principalement, des blocs élémentaires logiques (pour réaliser des fonctions booléennes), des mémoires RAM, des opérateurs arithmétiques (qui travaillent en virgule fixe), des ressources de routage interne et des entrées/sorties. Ces ressources configurables sont reliées par un réseau dense de lignes de routage et de lignes de transport des horloges. Ces lignes de routage sont aussi configurables. En plus de ces ressources, un FPGA est composé d'une mémoire interne de configuration. Chaque point de cette mémoire correspond à la configuration d'un élément d'une des ressources opérationnelles. Cette mémoire est, dans la plupart des cas, réalisée avec une des trois technologies suivantes : ANTIFUSIBLE (la plus ancienne, configurable une seule fois), FLASH (non-volatile) ou SRAM (volatile, la plus utilisée, représente plus de 80 % du marché). Pour réaliser une application avec un FPGA il faut décrire le circuit électronique à réaliser avec un langage de description matérielle comme le VHDL (Very High Speed Integrated Circuit Hardware Description Langage). Puis il faut synthétiser cette description en circuit électronique. Cette étape et les suivantes peuvent se faire avec des logiciels gratuits fournies par le fabricant de circuit. Enfin après une étape de placement et routage qui prend en compte l'architecture du FPGA, un fichier de configuration appelé bitstream est généré. Celui-ci permet de spécifier au FPGA lors de la configuration la position des points de la mémoire de configuration [41].

La technologie FPGA (Field-Programmable Gate Array) continue de gagner du terrain : selon les prévisions, le marché mondial du FPGA devrait passer d'1,9 milliard de dollars en 2005 à 2,75 milliards d'ici 2010 [1]. Depuis leur invention par Xilinx en 1984, les FPGA sont partis d'un simple rôle d'« interfaçage d'appoint » pour arriver à véritablement remplacer les ASIC (circuits intégrés à application spécifique) et les processeurs personnalisés dans des applications de contrôle et de traitement de signaux. Afin d'expliquer ce succès, cet chapitre

propose une introduction à la technologie FPGA et met en évidence quelques-uns des nombreux avantages que les FPGA sont aujourd'hui les seuls à offrir [43].

II.2. Description de la composent FPGA

Un FPGA est un circuit en silicium reprogrammable. À l'aide de blocs logiques préconstruits et de ressources de routage programmables, vous pouvez configurer ce circuit afin de mettre en œuvre des fonctionnalités matérielles personnalisées, sans avoir jamais besoin d'utiliser une maquette ou un fer à souder. Il vous suffit de développer des tâches de traitement numérique par logiciel et de les compiler sous forme de fichier de configuration ou de flux de bits contenant des informations sur la manière dont les composants doivent être reliés. En outre, les FPGA sont totalement reconfigurables et peuvent adopter instantanément une nouvelle « personnalité » si vous recompilez une nouvelle configuration de circuits. Jusqu'à présent, seuls des ingénieurs particulièrement expérimentés en matière de conception de matériel numérique pouvaient utiliser la technologie FPGA.

Si les FPGA rencontrent un tel succès dans tous les secteurs, c'est parce qu'ils réunissent le meilleur des ASIC et des systèmes basés processeur. Ainsi, ils offrent un cadencement par matériel qui leur assure vitesse et fiabilité, mais sont plus rentables que les ASIC personnalisés. Les circuits reprogrammables jouissent également de la même souplesse d'exécution logicielle qu'un système basé processeur, mais ils ne sont pas limités par le nombre de cœurs de traitement disponibles. Contrairement aux processeurs, les FPGA sont vraiment parallèles par nature, de sorte que plusieurs opérations de traitement différentes ne se trouvent pas en concurrence pour l'utilisation des ressources. Chaque tâche de traitement indépendante est affectée à une section spécifique du circuit, et peut donc s'exécuter en toute autonomie sans dépendre aucunement des autres blocs logiques. En conséquence, vous pouvez accroître le volume de traitement effectué sans que les performances d'une partie de l'application n'en soient affectées pour autant [43,38.46].

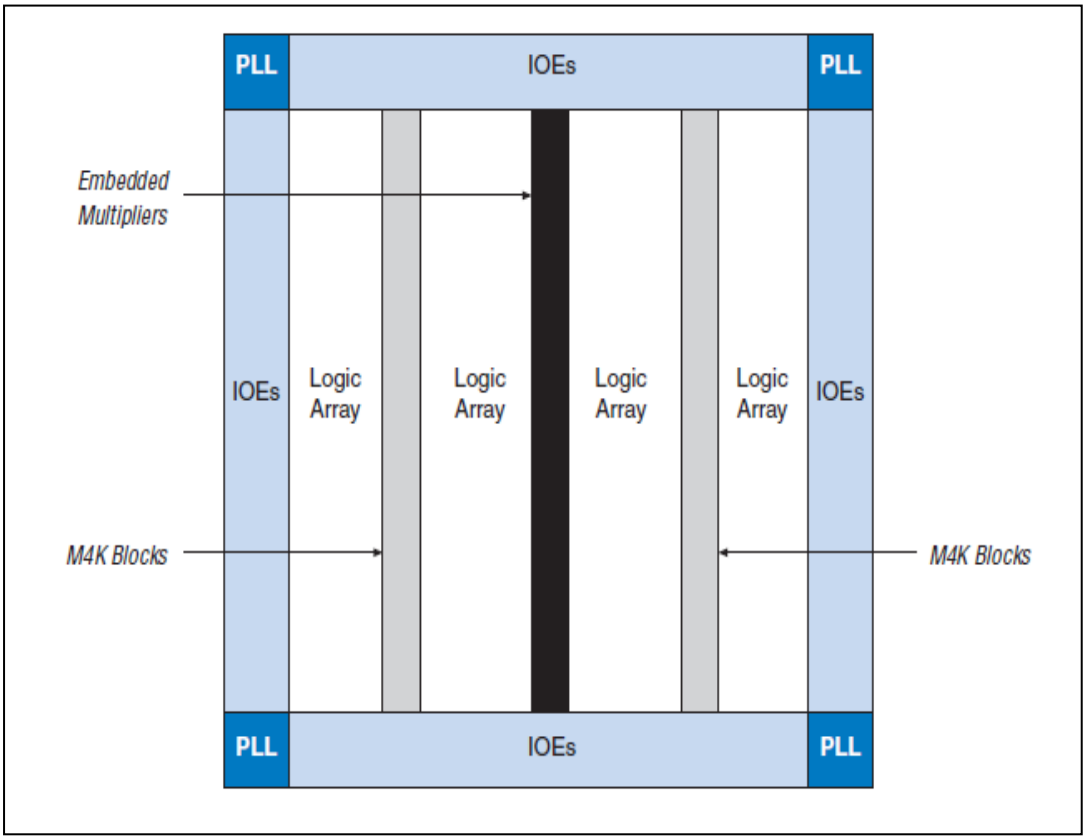


Figure II–1.a : Architecture interne du FPGA fabriqué ALTERA (Cyclone II EP2C20) [55].

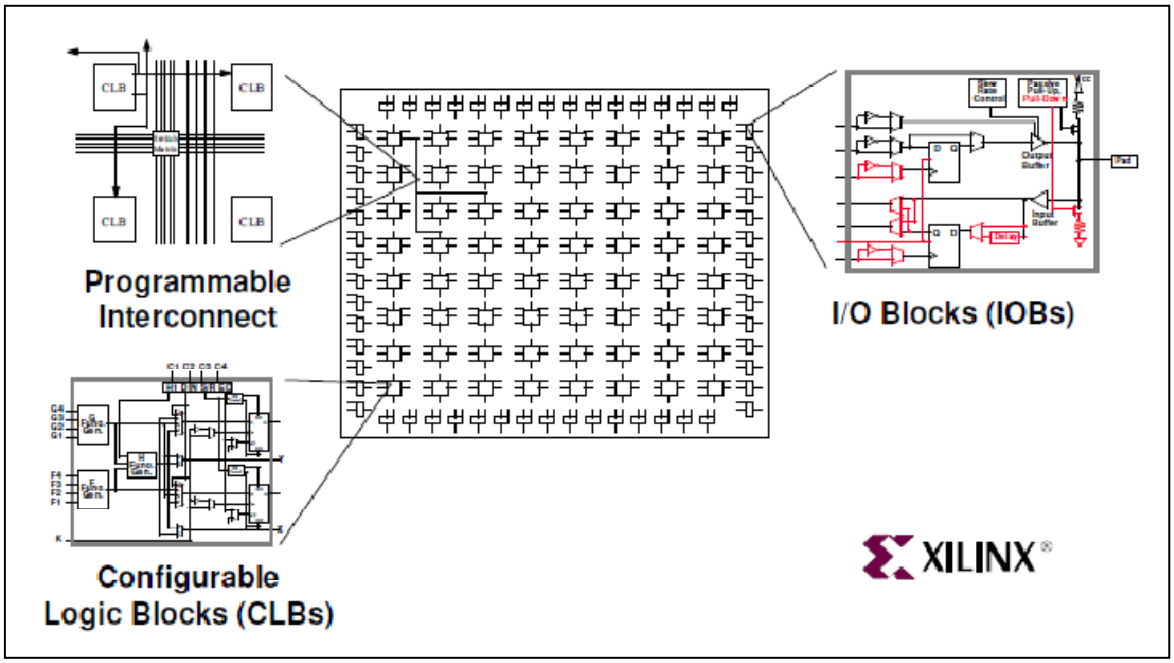


Figure II-1.b : Architecture interne du FPGA fabriqué Xilinx : XC4000 [38].

II.3. Les cinq principaux atouts de la technologie FPGA

1. Performances
2. Temps de mise sur le marché
3. Coût
4. Fiabilité
5. Maintenance à long terme

II.3 .1. Performances

Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. BDTI, une importante société d'analyse et de « *benchmarking* », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application.

II.3. 2. Temps de mise sur le marché

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, vous pouvez tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé [41]. Par la suite, vous pourrez apporter les éventuelles modifications nécessaires à votre FPGA, en quelques heures au lieu de quelques semaines. Le matériel « sur étagère » actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cœurs de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux.

II.3. 3. Coût

Les coûts d'ingénierie non récurrents (NRE) des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC se justifie largement pour les OEM, par exemple, qui peuvent livrer des circuits par milliers chaque année. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la reconception d'un ASIC.

II.3. 4. Fiabilité

Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche.

II.3. 5. Maintenance à long terme

Comme nous l'avons vu, les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la reconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, vous pouvez y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit [43].

II.4. Fabricants

Ayant présenté les caractéristiques globales des FPGAS, nous allons rapidement passer en revue les produits proposés par les différents fabricants, à savoir, dans l'ordre alphabétique, Actel, Altera, Atmel, Lattice Semiconductor, QuickLogic et Xilinx .

- **Actel**

Les principaux circuits d'Actel sont basés sur des anti-fusibles, les rendant non volatiles, rapides, et très sûrs sur le plan de la propriété intellectuelle. De plus, les anti-fusibles sont insérés entre les couches de métal, afin d'économiser la surface de silicium. Toutefois, la taille des ces circuits n'est pas exceptionnelle, la série Axcelerator contenant au plus 10'752 cellules composées d'un registre, 21'504 cellules combinatoires (quelques multiplexeurs et portes logiques), et 294'912 bits de mémoire dédiés, pour un total de 2 millions de portes équivalentes8.

Afin de proposer une alternative à leurs précédents produits tout en maintenant la non-volatilité, Actel a récemment introduit la série ProASIC Plus, basée sur une technologie FLASH. Elle est également sûre, les bits de configuration étant cryptés, et va jusqu'à une densité de 1 million de portes équivalentes.

- **Altera**

A l'heure de la rédaction de ces lignes, Altera, le deuxième plus gros fabricant, se concentre sur deux séries de base à technologie SRAM, Cyclone et Stratix. L'optique de la série Cyclone est de disposer d'un FPGA à bas prix. Bien que les éléments de base (LE, pour Logic Element) des deux séries soient semblables, la version Cyclone a été optimisée, et requiert 30% de moins d'espace. Il est principalement composé d'une LUT à 4 entrées, d'une bascule, et d'un système de propagation de retenue, afin d'accélérer les opérations arithmétiques. Le plus imposant des Cyclones contient 20'060 LEs et 288Kbits de RAM, pour un total de 1 million de portes équivalentes.

La série Stratix est nettement plus imposante, pouvant contenir jusqu'à 79'040 LEs, 7'427Kbits de RAM et 22 blocs DSP (88 multiplicateurs 1818, 178 de 99 ou 22 de 3636). En ajoutant jusqu'à 20 transcievers sériel à 3.125 Gbps et moyennant une baisse de densité (41'250 LEs et 3'423Kbits de RAM), nous obtenons la série Stratix GX. Enfin, la

dernière née des séries est la Stratix II. Elle est deux fois plus dense que la première, notamment grâce à une redéfinition de son élément de base.

Le LE a été remplacé par l'ALM (Adaptive Logic Module), qui contient deux bascules, deux reports de retenue, et un bloc combinatoire composé principalement de deux 4-LUTs et quatre 3-LUTs. A titre de comparaison, le plus gros Stratix II correspond à 179'400 LEs, 9Mbits de RAM et 384 multiplicateurs 1818.

Sur le plan des processeurs, la série Excalibur propose un FPGA de type APEX20KE (avec un maximum de 38'400 LEs, soit 1.7 millions de portes équivalentes), auquel a été ajouté un processeur ARM922T tournant à 200MHz. Il semblerait toutefois qu'il n'ait pas obtenu le succès escompté, et c'est plutôt sur son processeur Nios qu'Altera mise. Il s'agit d'un processeur soft, qui peut être inséré dans n'importe quel design, puis programmé sur le FPGA cible. Sa version 32 bits n'occupe que 1400 LEs, et permet donc d'en placer plusieurs sur un FPGA.

Finalement, Altera propose la solution HardCopy, qui offre à l'utilisateur la possibilité de transformer un design pour FPGA Stratix ou Apex en un ASIC. Basé sur le concept d'ASIC à réseau structuré, le circuit est prédiffusé avec les mêmes composants que ceux du FPGA, et seul l'apposition de deux couches de métal supplémentaires est nécessaire à la réalisation physique du design. L'avantage de cette approche est que la consommation est plus faible et la rapidité plus grande en comparaison de l'implémentation sur FPGA, et que le temps de réalisation est moins élevé que pour un ASIC standard.

- **Atmel**

Atmel, fabricant de semi-conducteurs, propose deux architectures de FPGA, la série AT6000 et la série AT40, toutes deux de technologie SRAM. Le bloc de base de la première est composé de deux multiplexeurs à quatre entrées, et de portes logiques simples, ainsi que d'une bascule, alors que celui de l'AT40, plus conventionnel, est composé de deux 3-LUT, d'une porte ET et d'une bascule. Bien que d'architecture relativement simple, l'originalité de ces deux circuits réside dans le routage. En effet, outre un réseau de routage longue distance simple, chaque cellule est directement reliée à ces 8 voisines, alors que chez la plupart des autres fabricants les liaisons directes ne sont que 4. Cette spécificité permet entre autre d'implémenter efficacement des multiplications de matrice. Concernant la taille, nous pouvons noter que ces circuits sont

plutôt petits en comparaison de leur concurrents, la série AT6000 possédant au maximum 75'000 portes équivalentes et la série AT40 1 million, dont 18Kbits de RAM.

Finalement, la série FPSLIC propose un système composé d'un microcontrôleur 8-bit AVR associé à un tableau reconfigurable identique à celui de l'AT40. Ce contrôleur a priori plus simple que l'ARM d'Altera ou le PowerPC de Xilinx, a toutefois la capacité de reprogrammer le FPGA dynamiquement, avantage certain quant à de potentiels systèmes adaptatifs. Notons également que la série Secure FPSLIC offre une EEPROM de 1Mbits, permettant une programmation immédiate au démarrage, sans la nécessité d'une mémoire externe stockant la configuration du FPGA.

- **Lattice**

Les circuits Lattice peuvent être regroupés en 3 familles de FPGA, ORCA, ispXPGA, ECP, et une famille FPSC. La première, ORCA, consiste en trois séries, à savoir les séries 2, 3 et 4.

La série 2 est composée d'éléments de base appelés PFU (Programmable Functional Unit) contenant quatre 4-LUT et quatre bascules. Ces PFUs peuvent servir à implémenter des blocs de mémoire RAM et ROM, synchrone, asynchrone ou dual-port, et offrent un maximum de 99'400 portes équivalentes. Les PFUs de la série 3 et des suivantes ont une taille doublée par rapport à la série.

La série 3 possède également une interface processeur facilitant la configuration et propose des circuits allant jusqu'à 340K portes équivalentes. Enfin, la série 4 offre encore plus de complexité avec des blocs de mémoire additionnels pour un total maximal de 148Kbits, pouvant être utilisés comme RAM, ROM ou multiplicateur, et un maximum de 899K portes équivalentes. Une interface processeur y est également proposée, qui sera exploitée dans les circuits ECP.

Alors que la famille ORCA fut récupérée lors de l'achat de Lucent Technologies, la famille ispXPGA fut entièrement développée par Lattice Semiconductors. Son élément de base est composé de quatre 4-LUT et huit bascules, pour l'implémentation efficace de pipelines. Chaque PFU peut y être utilisé pour la réalisation de 6-LUT, d'une fonction logique jusqu'à 20 entrées, d'un multiplexeur à huit entrées, d'un bloc de 61 bits de RAM, ou d'un registre à décalage de 8 bits. Jusqu'à 246Kbits de RAM sont en outre disponibles, et le nombre maximal de portes équivalentes proposé est de 1'250K. La

grande particularité de cette famille concerne sa programmation basée sur des cellules de mémoire ECMOS, offrant la non-volatilité à ces circuits. De ce fait aucune mémoire externe n'est nécessaire, et le circuit est automatiquement configuré lors du démarrage.

Suivant la série 4, Lattice propose des Field Programmable System Chip (FPSC) composés d'un tableau reconfigurable identique à celui trouvé dans la série ORCA4 ainsi que d'éléments réalisés en technologie ASIC, pouvant être des interfaces 10Gbits/s, ou backplane transceivers.

Finalement, deux familles, LatticeECP, pour EConomy Plus, et LatticeECPDSP, offrent une solution à bas coûts. Leurs éléments de base sont relativement semblables à ceux de la série ORCA4, mais elles offrent plus de modules utiles à l'implémentation d'applications DSP, tels que des multiplicateurs (jusqu'à quarante 1818), et des blocs de mémoire. La famille ECP-DSP propose en plus jusqu'à 10 blocs DSP permettant chacun l'implémentation de huit multiplicateurs 9 bits ou quatre Multiply-Accumulate sur 9 bits, par exemple. Le plus grand de ces circuits possède 40 multiplicateurs, 5120 PFUs et 645Kbits de RAM.

- **QuickLogic**

Tous les circuits QuickLogic sont de types anti-fusibles (la technologie ViaLink leur permet d'intercaler le fusible entre deux couches de métal, afin de sauver de l'espace sur le silicium), basés sur un même élément de base, avec une légère différence pour la série Eclipse. Cet élément comprend 2 portes ET à 6 entrées, 4 portes ET à 2 entrées, 6 multiplexeurs à 2 entrées, et une bascule. Cette architecture est particulière en comparaison des concurrents. En effet, elle semble moins intuitive que l'approche LUT, mais permet de créer des fonctions à grand nombre de variables d'entrées comme plusieurs fonctions à moins de variables, grâce à ses 5 sorties. Dans la série Eclipse, la principale différence réside en la présence de deux bascules au lieu d'une. Dans tous les circuits, le réseau de routage est composé de lignes et colonnes connectés par des switchboxs.

La série pASIC consiste en un simple tableau d'éléments logiques et du routage correspondant, pour un maximum de 1584 cellules logiques, soit 75'000 portes équivalentes.

Augmentée de blocs de RAM, la série QuickRAM propose jusqu'à 25'344 bits, pour un total de 176'608 portes équivalentes. La série QuickPCI, toujours sur la même base, contient quant à elle un contrôleur PCI, alors que la série QuickMIPS, de loin la plus complexe, possède, outre 1152 cellules, 82'944 bits de RAM et 18 ECUs, un processeur MIPS 32 bits, un contrôleur PCI, 2 UARTs, etc. La série Eclipse, quant à elle, pour un maximum de 662'208 portes équivalentes, peut contenir jusqu'à 82'900 bits de RAM et 18 blocs ECU (Embedded Computational Units), chacun étant composé d'un multiplicateur 8X8, d'un additionneur 16 bits et d'un registre.

- **Xilinx**

A l'heure actuelle, Xilinx, le premier fabricant de FPGAs, propose principalement deux familles, Virtex et Spartan, toutes deux de type SRAM, basées sur une architecture LUT. La différence entre les deux familles est minime, et tient principalement du nombre d'éléments proposés ainsi que du type de process utilisé, les Spartan étant positionnées bas coût en comparaison des Virtex. L'élément de base, le CLB (Configurable Logic Block), est composé de deux Slices, eux-mêmes comprenant deux 4-LUT et deux bascules. Les versions Virtex-II, Virtex-4 et Spartan-3 diffèrent toutefois, le CLB y contenant quatre Slices. Les deux familles contiennent des blocs de RAM, pouvant être utilisés en single ou dual port.

Alors que les séries Spartan-3, Virtex-II et Virtex-4 sont presque identiques en terme d'architecture, la série Virtex-II Pro introduit un ou deux processeurs de type PowerPC. Contrairement à l'échec de la solution d'Altera faisant intervenir un ARM, Xilinx a su imposer son produit, et la série Virtex-II Pro se trouve très bien positionnée dans la gamme de ses produits.

Le plus imposant de la série Spartan-3 propose 5M de portes équivalentes, dont 104 multiplicateurs de 18x18 bits et 1'872 K bits de RAM. En comparaison, les plus gros Virtex sont le XC4VLX200, de la famille Virtex-4LX et le XC4VFX140, un Virtex-FX. Le premier contient 178'176 éléments logiques, un élément logique étant une LUT à 4 entrées et une bascule. A ceci s'ajoutent 6'048 Kbits de RAM configurables, ainsi que 96 multiplicateurs 18x18 bits, pour un total de 15M de portes équivalentes (chiffre calculé). Le deuxième, le XC4VFX140, est composé de 126'336 éléments logiques, de 9'936 KBits de RAM, 192 multiplicateurs, et 2 processeurs de type PowerPC.

Nous pouvons finalement noter que Xilinx, à l'instar d'Altera, propose une solution appelée EasyPath, permettant la réalisation en ASIC d'un design implémenté sur un Virtex-II, Virtex-II Pro ou un Spartan-3 [46].

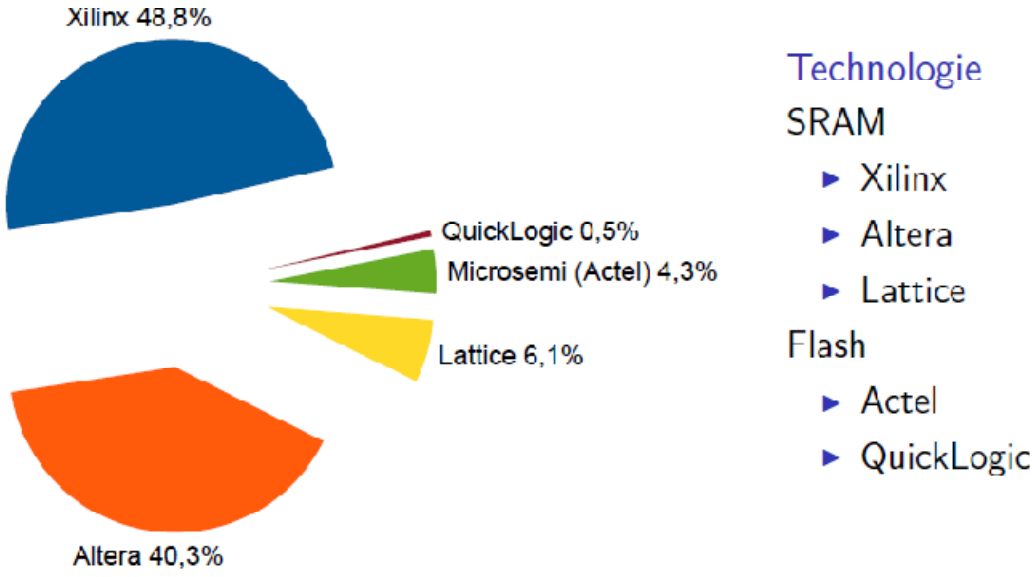


Figure II-2 : Parts de marché en termes de CA en 2010 [56].

Le tableau II-1 de Comparaison des caractéristiques des différentes FPGAs résume les principales caractéristiques des FPGAs disponibles sur le marché. Nous pouvons observer une complexification constante de ces circuits qui se voient ajouter de plus en plus de nouvelles caractéristiques. Il devient en effet quasiment impossible d'en trouver un ne possédant pas de blocs de mémoire RAM, et la présence de multiplicateurs devient presque systématique.

Sur le plan des structures de routage, la tendance est à l'optimisation. Les technologies utilisées devenant de plus en plus petites, le délai des portes se réduit d'autant, alors que celui des fils ne diminue pas. Dès lors une nouvelle attention est donnée à cette partie du circuit, de manière à assurer un fonctionnement correct à haute fréquence.

Concernant la complexité du routage, nous pouvons noter que les réseaux de routage des gros FPGAs ne se bornent pas à une simple grille de switchboxs, mais qu'ils font intervenir plusieurs niveaux. La série Startix d'Altera, par exemple, est composés de Logic Array Blocs (LABs) contenant 10 Logic Elements (LEs), chaque LE étant une LUT, une bascule et quelques portes logiques. Le LAB y a une connectique interne, et une connectique externe

pour des chemins de longue distance entre les LABs. Altera voit ce réseau comme une structure à plusieurs dimensions, l'ensemble du réseau étant vu comme une grille de sous-réseaux interconnectés, eux-mêmes étant une grille de sous-réseaux, etc [46].

Vendeur	Famille	Technologie	Portes Equivalentes	Nb Flip-flops	Mémoire Bloc (Kbits)	Multiplicateur 18×18	Processeur Processeur
Actel	Axcelerator	anti-fusible	2M	21'504	294	-	-
Actel	ProASIC Plus	FLASH	1M	56'320	198	-	-
Altera	Cyclone	SRAM	1M	20'060	295	-	-
Altera	Stratix	SRAM	4M (calcul)	79'040	7'427	176	-
Altera	Stratix II	SRAM	9M (calcul)	143'520	9'383	384	-
Altera	Excalibur	SRAM	1.7M	38'400	256	-	1 ARM922T
Atmel	AT40	SRAM	50K (Usable gates)	3'048	18.4	-	-
Atmel	AT6000	SRAM	30K (Usable gates)	6'400	-	-	-
Atmel	FPSLIC	SRAM	50K (déduit)	2'962	18.4	-	8-bit AVR
Lattice	ORCA4	SRAM	899K	18'216	148	-	-
Lattice	ispXPGA	E^2CMOS	1.25M	30'700	414	-	-
Lattice	ECP	SRAM	1M (calcul)	46'080	645	40	-
QuickLogic	Eclipse II	anti-fusible	320K	4'002	55	-	-
QuickLogic	pASIC	anti-fusible	75K	2'692	-	-	-
QuickLogic	QuickRAM	anti-fusible	176K (90K Usable PLD gates)	2'692	25	-	-
QuickLogic	QuickMIPS	anti-fusible	457K (115K Usable PLD gates)	4'032	83	18	MIPS32 4Kc
Xilinx	Virtex-II	SRAM	8M	93'184	3'024	168	-
Xilinx	Virtex-II Pro	SRAM	8M	88'192	7'992	444	2 PowerPC
Xilinx	Virtex-4LX	SRAM	15M (calcul)	178'176	6'048	96	-
Xilinx	Virtex-4FX	SRAM	11M (calcul)	126'336	9'936	192	2 PowerPC
Xilinx	Spartan-3	SRAM	5M	66'560	1'971	104	-

Tableau II-1 : Comparaison des caractéristiques des différentes FPGAs [46].

II.5. Structure interne de FPGA

II.5.1. Architectures des FPGA

L'architecture d'un FPGA est principalement décrite par la topologie des ressources de routages et des éléments logiques configurables de base. Il existe deux architectures classiques, l'architecture îlot de calcul (initialement utilisée dans les composants Xilinx) et l'architecture hiérarchique (initialement utilisée dans les composants Altera). Cependant, une tendance apparait avec les dernières générations de circuits, les architectures sont principalement de style îlots de calculs avec une légère hiérarchique (un ou deux niveaux de cluster hiérarchique). Les sections suivantes donnent quelques détails sur ces architectures. Dans le passé, d'autres architectures ont étaient utilisées. On peut citer l'architecture de routage logarithmique utilisée par Xilinx pour son circuit XC6000. Malheureusement, les outils de placement routage n'étaient pas adaptés à cette architecture pour permettre au concepteur d'en tirer pleinement partie.

II.5.1.1. Architecture îlot de calcul

L'architecture la plus communément utilisée pour réaliser ces circuits est de type *îlot de calcul*. Dans ce cas les ressources configurables sont disposées sous formes de matrice, comme on peut le voir sur la figure II-3. Des lignes de routage sont disposées horizontalement et verticalement autour des ressources configurables. Des blocs de connexion relient les ressources configurables aux lignes de connexion. Des matrices de connexion relient les lignes de routage horizontales et verticales.

L'utilisation de matrices de connexions configurables est indispensable pour assurer la connectivité des modules, mais les matrices de connexions configurables dégradent les caractéristiques des signaux, diminuent les performances (fréquence de fonctionnement et consommation de puissance) et nécessitent des outils de placement-routage efficaces. Sur la figure 4, on peut voir en gras des liaisons point à point entre deux éléments configurables. Ces liaisons utilisent : les ports d'entrées/sorties des éléments configurables, les connexions configurables qui permettent la connexion des éléments configurables au réseau de routage, les lignes de routages et les matrices de connexions configurables. Autant d'éléments parcourus qui dégradent les performances du circuit mais qui permettent une flexibilité importante.

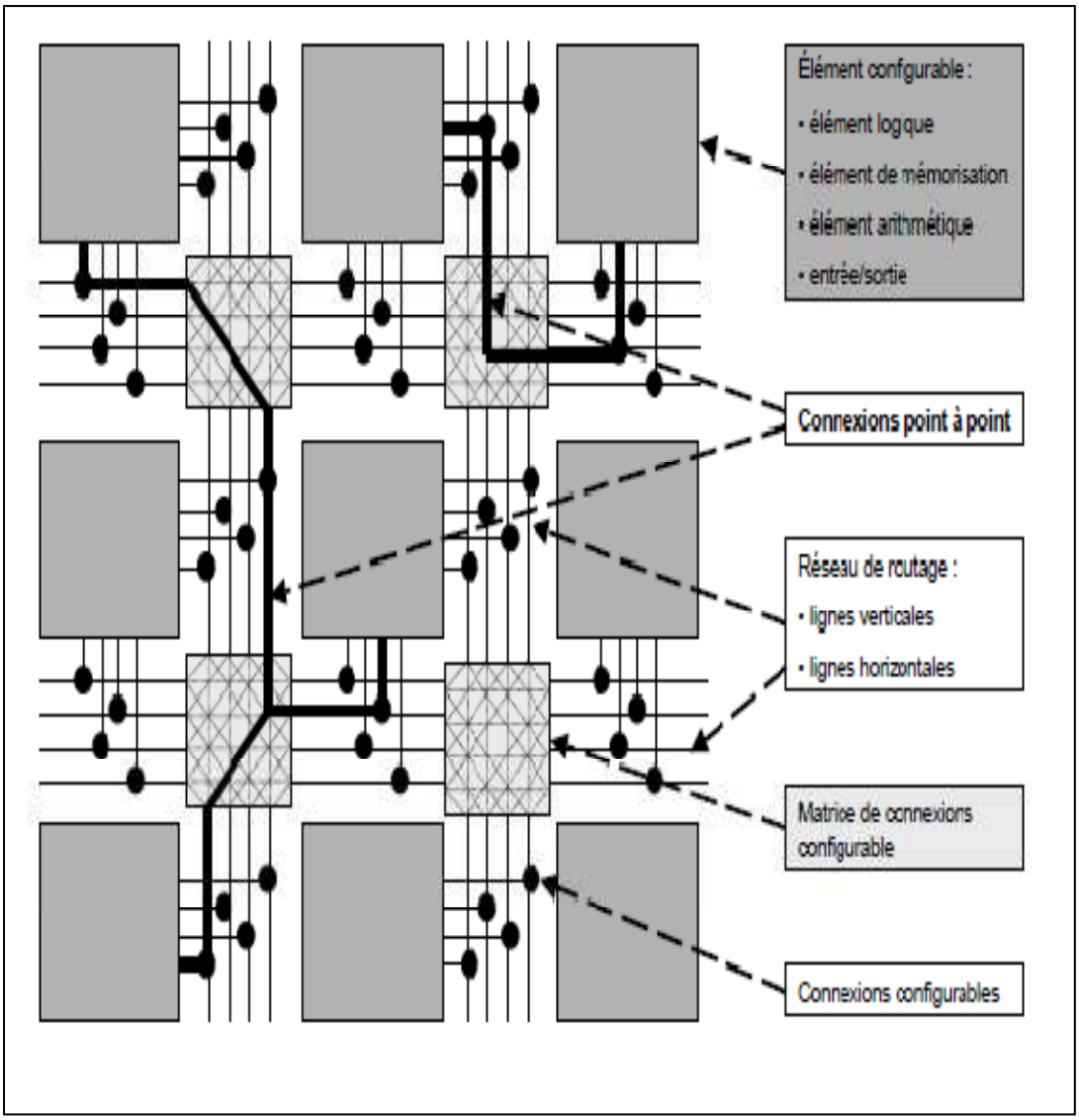


Figure II- 3 : Architecture îlot de calcul, typique des FPGA actuels [42].

II.5.1.2. Architecture hiérarchique

L'architecture hiérarchique couramment utilisée pour les circuits FPGA est constituée de quatre ou trois niveaux. A chacun de ces niveaux des ressources de routage sont disponibles pour communiquer entre les éléments propres du circuit. La figure 5 schématise une architecture hiérarchique à quatre niveaux en utilisant un exemple d'architecture couramment rencontrée dans les FPGA ACTEL. Au niveau le plus haut de la hiérarchique, le circuit est constitué de tuiles agencées matricielle ment. Les tuiles sont constituées de clusters logiques et de bancs de mémoires. Enfin, les clusters logiques regroupent les éléments logiques (et/ou arithmétiques) configurables. Ce style d'architecture peut être très efficace énergétiquement

car elle permet de localiser les communications intenses et limite l'utilisation de longues lignes de routage. Cependant, il est nécessaire pour les algorithmes de placement et routage de prendre en compte les caractéristiques Spécifiques de ces architectures.

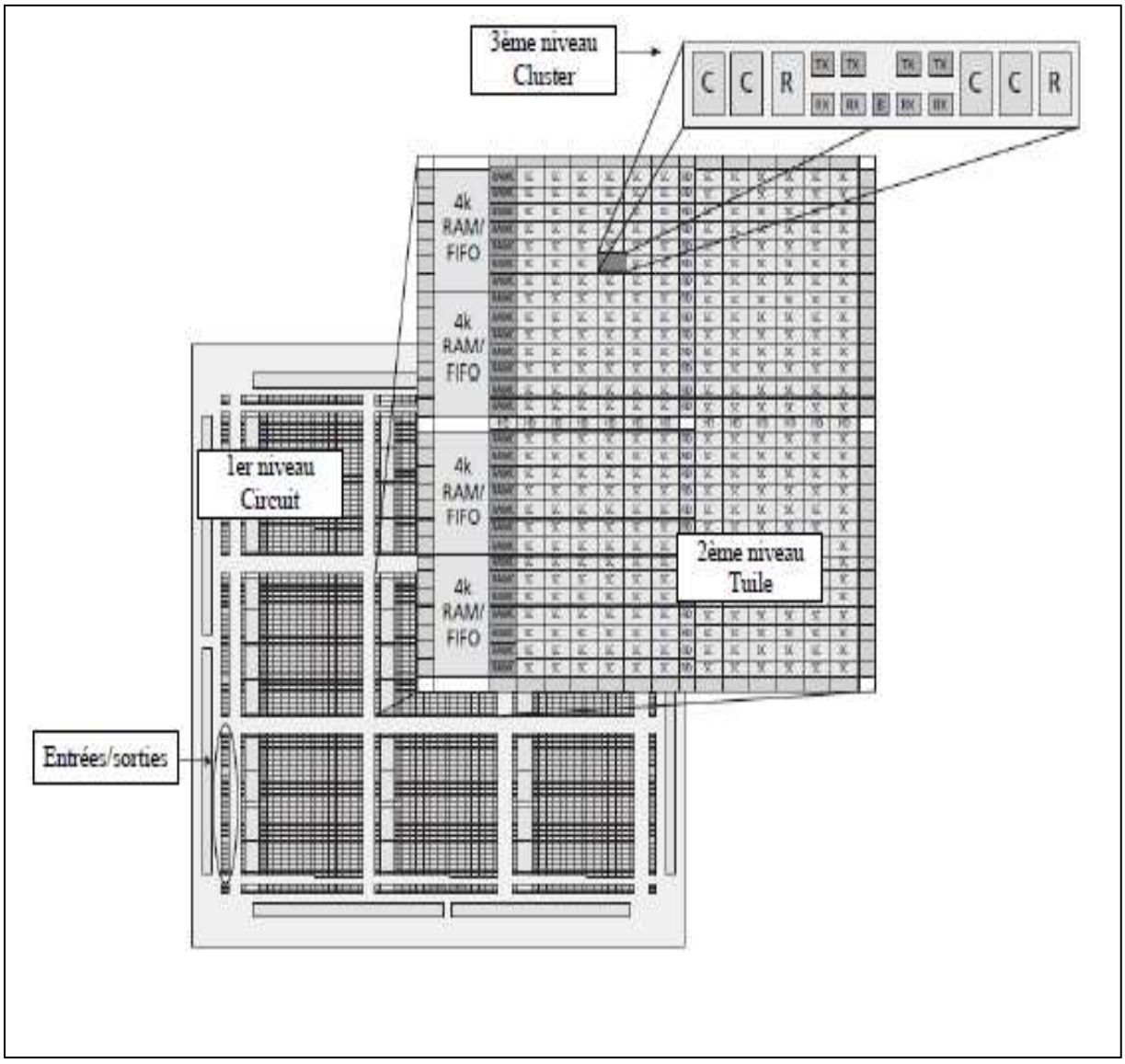


Figure II-4 : Exemple d'Architecture hiérarchique à quatre niveaux (circuit, tuiles, clusters, éléments configurables) que l'on rencontre fréquemment dans les circuits ACTEL [42].

II.5.1.3. Architecture de type mer de portes

Elle est composée hiérarchiquement et le routage est de type logarithmique. Ce type de topologie fut utilisé par Xilinx pour sa série 6000. Mais ces composants n'ont pas eu de succès, commercialement parlant, par manque d'outils de CAO capables de les exploiter correctement. Peut être, reverrons-nous un jour des architectures de ce type ressortir sur le marché [41].

II.5.1.4. Architecture Spacetime

La société Tabula vient, de son côté, de concrétiser pour la première fois son architecture Spacetime avec sa famille de réseaux logiques programmables (3PLD) baptisée Abax. Spacetime revendique l'appellation 3D en prenant le temps comme troisième dimension pour optimiser l'utilisation des ressources sur la puce et ainsi accroître la densité de logique disponible. La société donne du volume à la matrice matérielle en la reconfigurant très rapidement : 1,6 milliard de fois par seconde (1,6 GHz), soit environ un million de fois plus rapidement que les FPGA actuels qui chargent la configuration à partir d'une mémoire externe. Pour faciliter la gestion de cette structure temporelle et permettre une reconfiguration

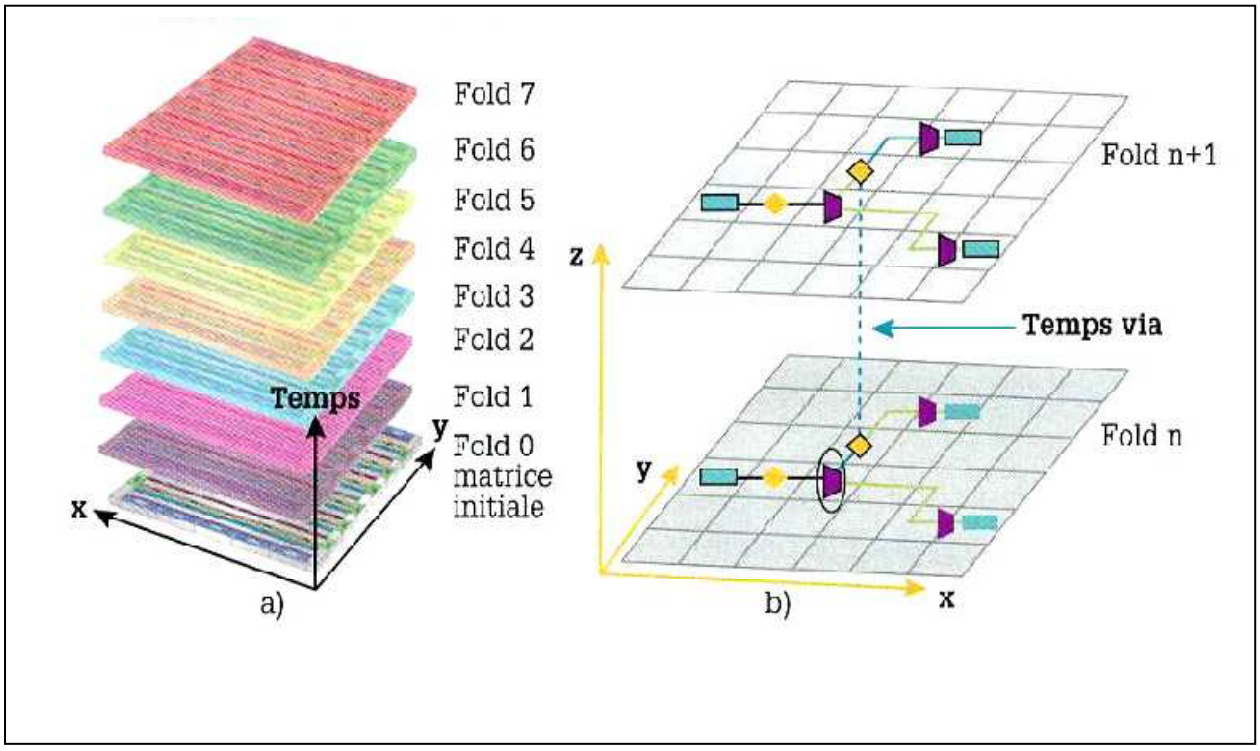


Figure II-5 : Architecture Spacetime (société Tabula) [45].

ultra rapide de la logique, la succession de configurations est stockée localement sur la puce. Par rapport aux solutions conventionnelles, la société annonce pour ses FPGA fabriqués dans le procédé 40 nm de TSMC, des gains respectifs de 2,5 pour la densité, de 2 pour la capacité mémoire et de 3,7 pour les performances DSP. Tabula précise également que cette reconfiguration est transparente pour l'utilisateur, celui-ci pourra donc suivre une méthodologie de conception classique. Dans l'architecture Spacetime, la matrice matérielle est découpée en plusieurs zones de blocs logiques basés sur des LUT, alternant avec des bancs de mémoire Ram et des multiplexeurs pour distribuer les signaux entre ces diverses entités. Suivant le même principe que celui de la logique programmable, l'axe des temps donne un effet de volume à la mémoire, multipliant sa capacité initiale ainsi que le nombre de ports. Lorsqu'en 2003, Steve Teig, président et CTO de Tabula, mit au point le concept de cette architecture Spacetime, le premier travail de son équipe fut d'étudier la faisabilité d'outils de CAO de type classique pour « animer » cette architecture. Les premiers membres de la famille Abax (A1EC02,-03,-04,-06) embarquent entre 220 000 et 630 000 LUT par FPGA. Ils possèdent tous 5,5 Mo de Ram, 920 E/S parallèles, 44 PLL ainsi que 48 SerDes opérant entre 55 Mbits/ s et 6,5 Gbits/ s. Le FPGA A1EC06 bénéficie en plus de 1 280 blocs MAC (multiplier/accumulate). L'échantillonnage de ces FPGA est prévu en juillet, pour une production en volume en octobre.

Quelle que soit l'architecture choisie, les éléments constitutifs d'un FPGA sont toujours à peu près les mêmes. Chaque fabricant ayant ses variantes par rapport à un autre. Nous pouvons citer un certain nombre de ces éléments [45].

II.5.2. Ressources fonctionnelles configurables

Dans la plupart des cas l'élément logique configurable de base des FPGA se compose d'une LUT (Look Up Table5) avec un nombre d'entrées allant de 4 à 8 pour les dernières générations, d'une chaîne de propagation rapide de la retenue et d'un registre de sortie afin d'assurer la synchronisation des signaux (très utile pour l'implémentation de calculs pipelinés). Ces éléments configurables peuvent être rassemblés en clusters hiérarchiques afin de favoriser une connectivité locale et rapide (cas des composants Altera). Les éléments logiques configurables appelé SLICE chez Xilinx. Entre deux générations successives de FPGA Xilinx, qu'un SLICE Virtex4 ne correspond pas à un SLICE Virtex6.

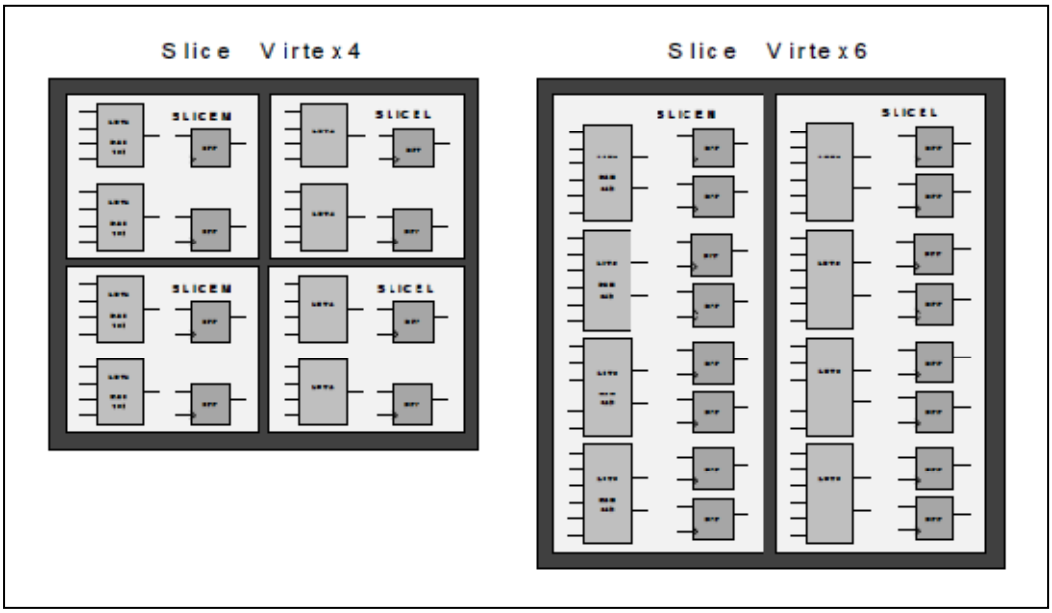


Figure II-6 : Eléments logiques configurables (circuits Xilinx Virtex4 et Virtex6) [42].

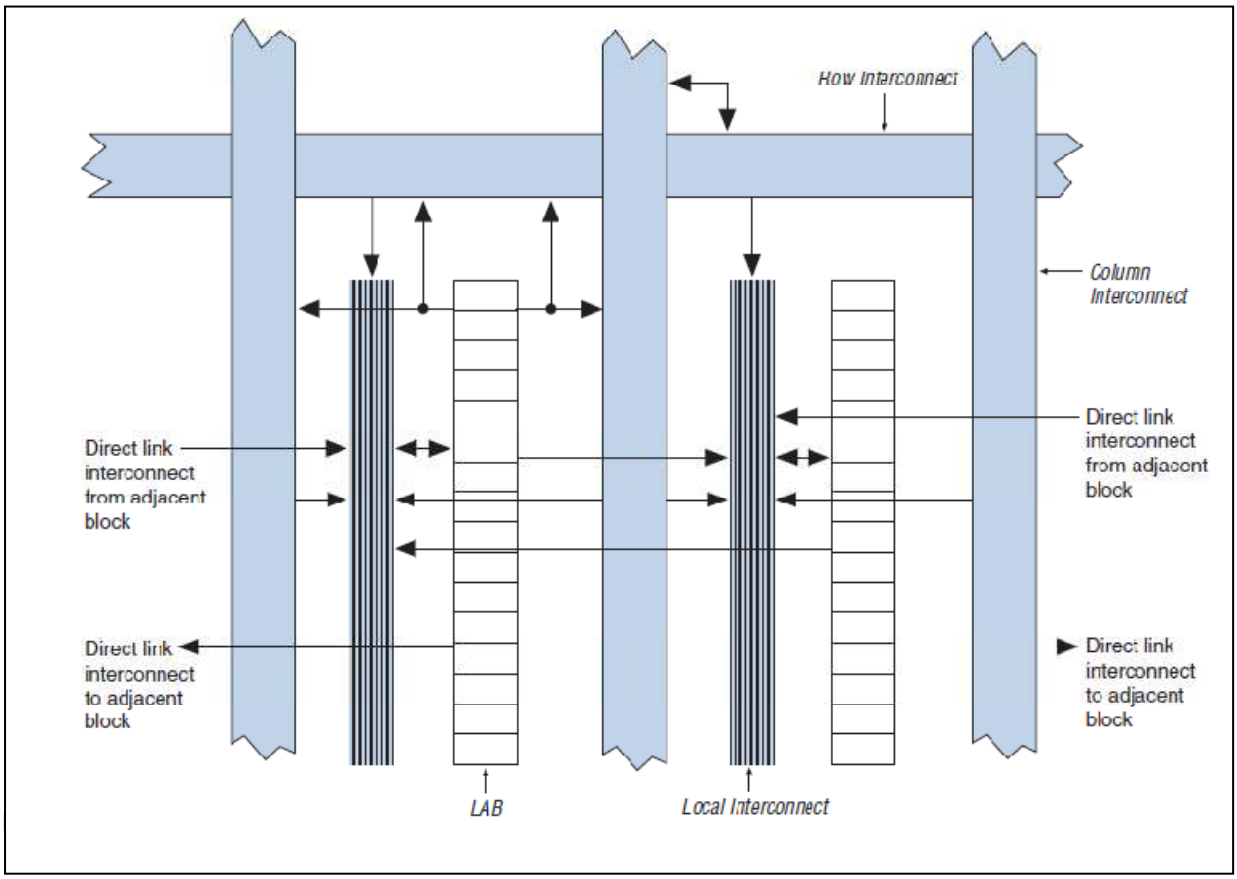


Figure II-7.a : Eléments logiques (LAB) configurables (Logic Blocks, Cyclone II) d'Altera [55].

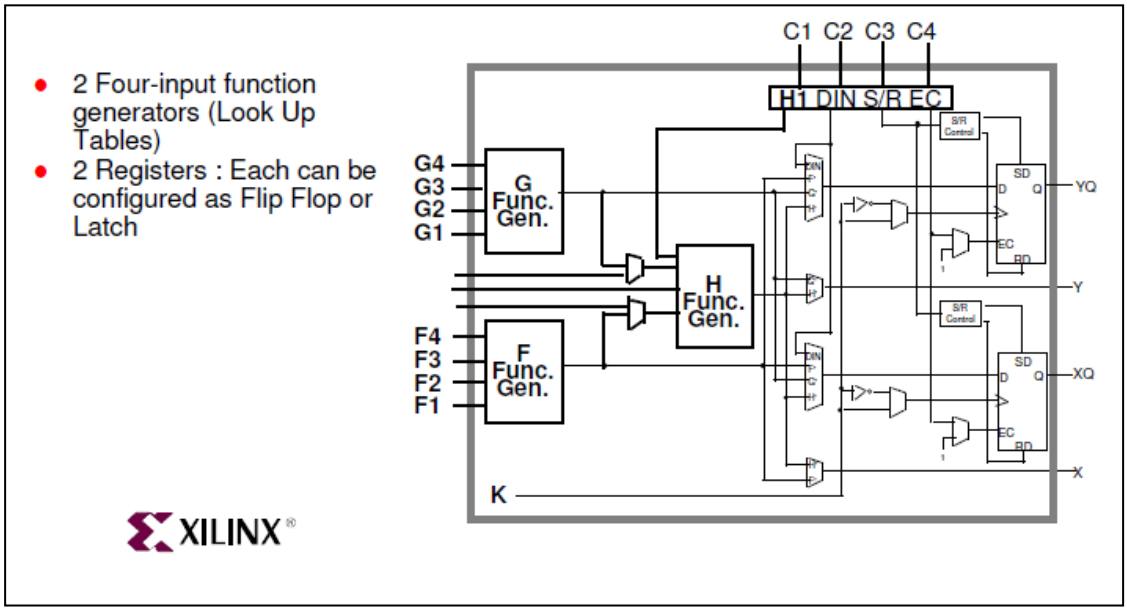


Figure II.7.b : Eléments logiques configurables (simplifiés) des circuits XC4000 Configurable (LAB) Logic Blocks de XILINX [38].

Remarque : Une LUT peut être considérée comme une petite mémoire RAM dans laquelle on mémorise la table de vérité d'une fonction logique. Une LUT à 4 entrées et 1 sortie, classiquement utilisée dans les FPGA, est donc équivalente à une RAM 16 bits.

Quelques rares circuits ne comportent pas de LUT et sont constitués de cellules configurables basées sur des multiplexeurs. Ce fut le cas du circuit Xilinx XC6000 qui n'est plus disponible. C'est le cas pour certains composants ACTEL comme les familles FPGA de technologie FLASH ProASIC-3 et Fusion ainsi que la famille de FPGA de technologie anti-fusible Axcelerator.

Rapidement, afin de réaliser complètement des applications modernes, les FPGA ont dû se doter d'éléments configurables de mémorisation (apparition en 1999 dans les composants Virtex et Apex de Xilinx et Altera). Sans ceux-ci la mémoire synthétisée doit être distribuée sur les LUT, ce qui laisse peu de place pour les traitements [41,38,8].

II.5.2.1. Les éléments de mémorisation

Les FPGAs ont changé et ne sont plus seulement utilisés pour des applications de type "glue logique", comme ce fût le cas à leur début, mais aussi pour des applications plus importantes qui demandent souvent des capacités de stockage (citons en exemple les applications du traitement d'images). La nécessité d'intégrer des blocs de mémoires directement dans l'architecture des FPGAs est vite devenue cruciale. De cette façon les temps d'accès à la mémoire sont diminués puisqu'il n'est plus nécessaire de communiquer avec des éléments extérieurs au circuit. Dans les exemples détaillés par la suite, nous verrons que ces blocs mémoires sont généralement de type RAM, et sont configurables au niveau de la largeur des mots mémorisés. Du fait de l'intégration de ces blocs, dans des architectures déjà existantes, un point crucial apparaît qui est le routage entre les parties mémoires et les ressources logiques. La caractéristique importante de ce routage est sa flexibilité. Si ce routage n'est pas assez flexible alors le circuit est difficilement routable, si il est trop flexible il y a surconsommation de surface (donc de silicium) [41,8].

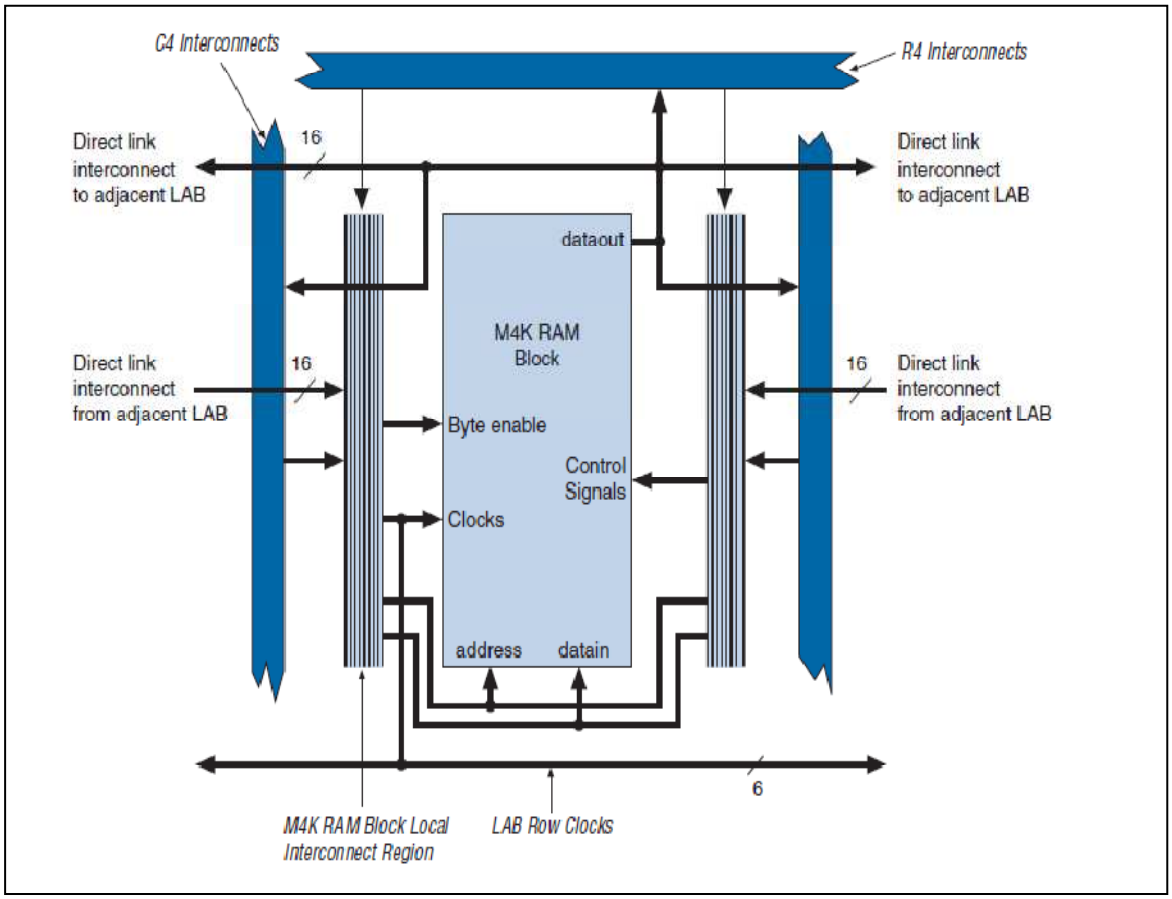


Figure II–8 : Éléments de mémorisation (M4K RAM) d'Altera [55].

II.5.2.2. Les éléments de routages

S'il y a des éléments plus importants que d'autres dans les FPGAs, ce sont les éléments de routages. En effet les ressources de routages représentent la plus grosse partie de silicium consommée sur la puce réalisant le circuit. Ces ressources sont composées de segments (de longueurs différentes) qui permettent de relier entre eux les autres éléments via des matrices de connexions. Le routage de ces ressources est un point critique du développement d'une application sur un FPGA, et les méthodes utilisées pour les ASICs ne sont plus tout à fait valables (du fait de la segmentation des ressources). Si ces éléments sont importants c'est parce qu'ils vont déterminer la vitesse et la densité logique du système. Par exemple les matrices de routage (programmable Switch) sont, physiquement, réalisées grâce à des transistors de cellules SRAMs, qui ont une résistance et une capacité, ce qui entraîne l'existence de constantes de temps [41,8,38].

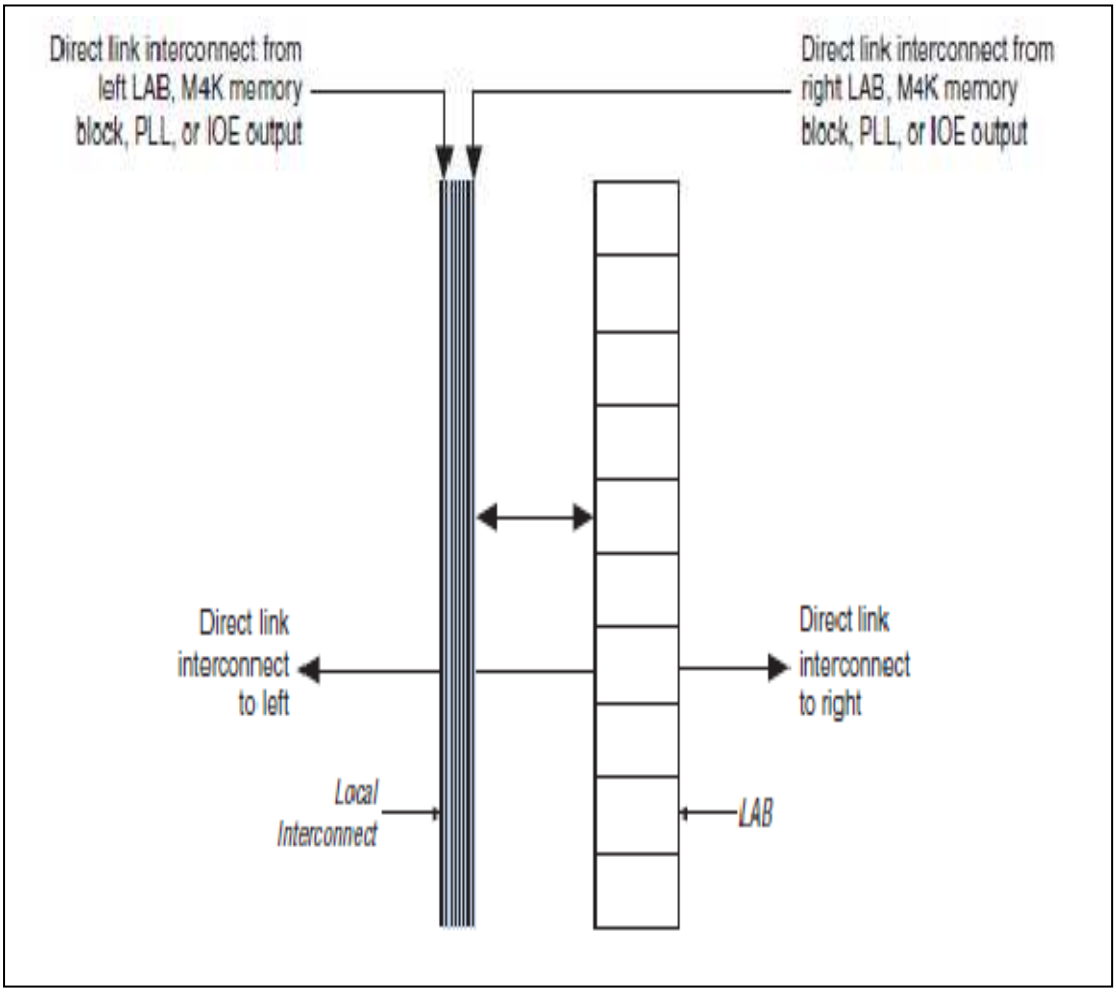


Figure II–9 : Élément de routages (Altera) [55].

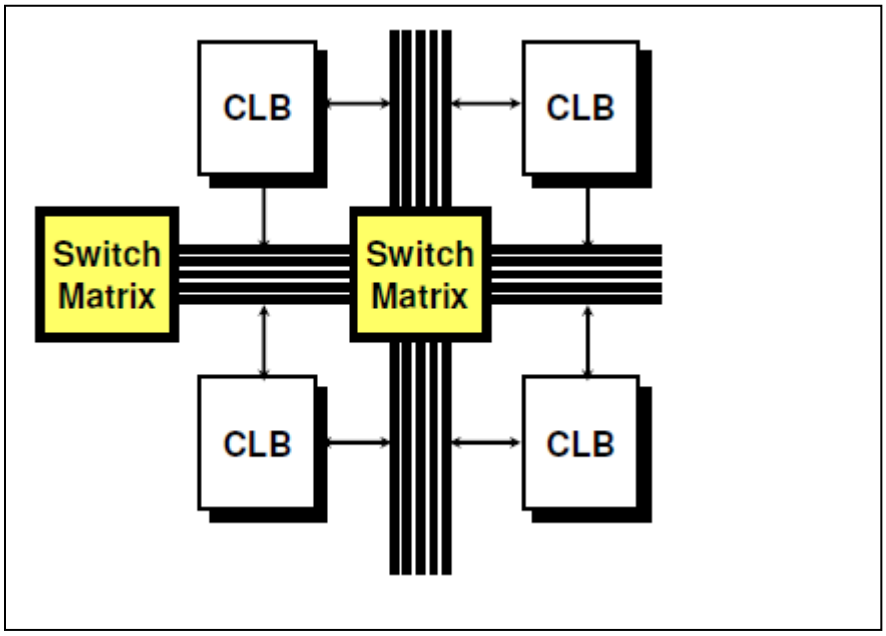


Figure II-10 : Élément de routages (Xilinx) [38].

II.5.2.3. Les éléments d'entrées sorties

Le composant ne vie pas seul sur une carte (ou très rarement), il appartient à un système d'ensemble pouvant contenir des parties micro programmées comme dans le cas du "Co-Design" (conception conjointe logiciel matériel). Le circuit doit donc avoir un lien avec son environnement, c'est le but des éléments d'entrées/sorties. Ceux-ci peuvent bénéficier de protections, de buffer ou d'autres éléments permettant la gestion des entrées et des sorties. En particulier il est à noter que les circuits actuels proposent différentes normes pour les niveaux d'entrées et de sorties (par exemple: LVTTL 2 - 54mA, PCI 5V, PCI 3.3V, HSTL…) qui par configuration peuvent êtres choisies afin de s'adapter à l'environnement [41,38,8].

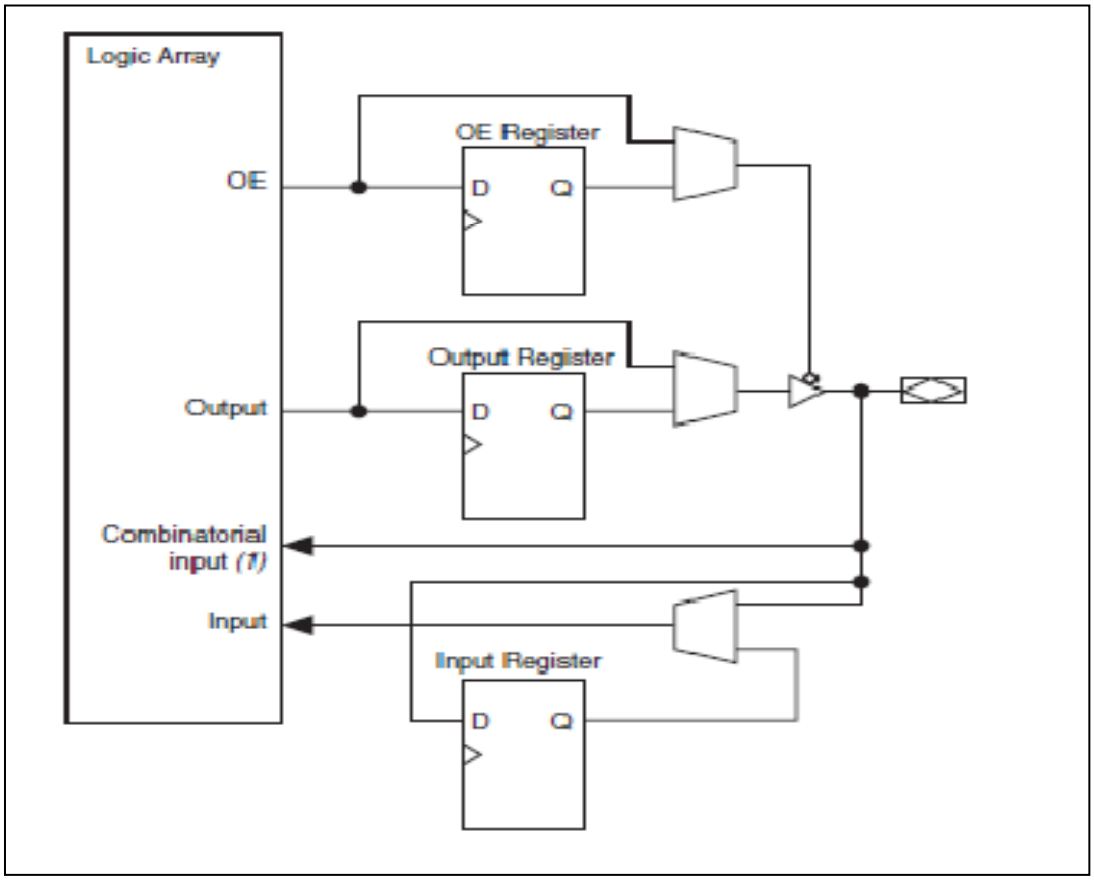


Figure II–11 : Élément d'entrées sorties (Altera) [55].

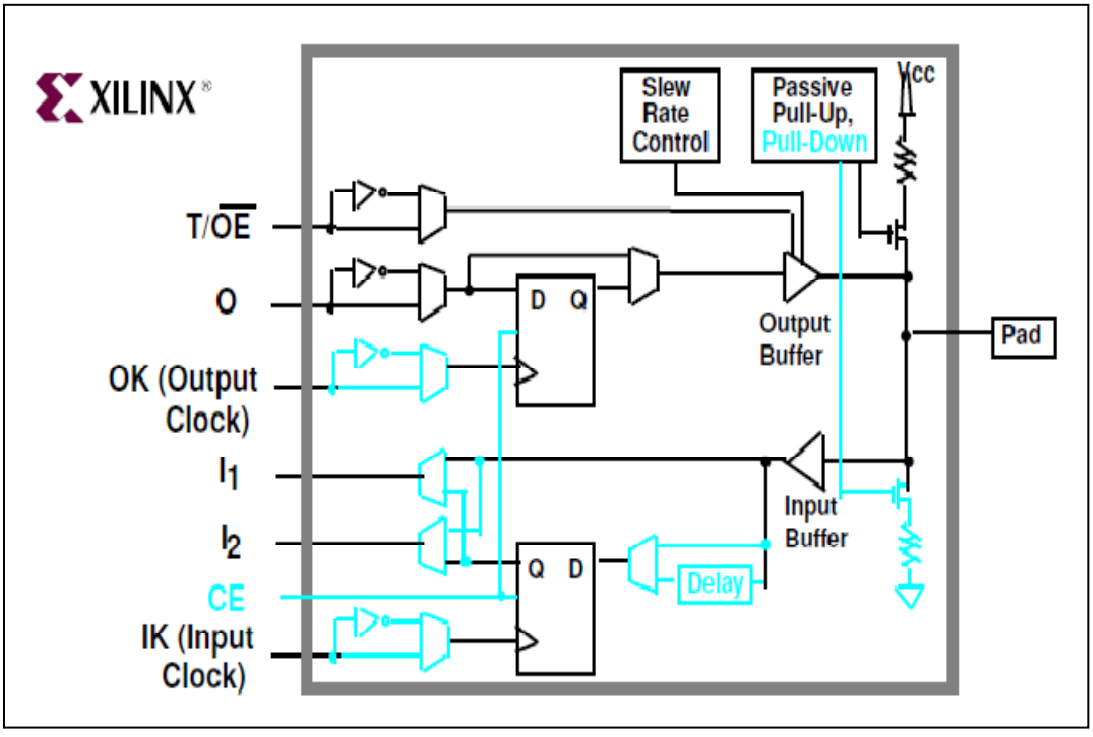


Figure II-12 : Élément d'entrées sorties (XC4000E-Xilinx) [38]**.**

II.5.2.4. Les éléments de contrôle et d'acheminement des horloges

Il paraît évident que dans tout système électronique relativement important, il faut disposer d'horloges et qu'elles sont souvent d'une importance capitale pour le bon fonctionnement du système (pensons aux systèmes de télécommunications par exemple). Alors, bien sur, les FPGAs sont prévus pour recevoir une ou plusieurs horloges. Des entrées peuvent être spécialement réservées à ce type de signaux, ainsi que des ressources de routages spécialement adaptées au transport d'horloges sur de longues distances (bufférisassions des lignes). Aussi, pour assurer d'avoir la même horloge dans tout le circuit (synchronisation des signaux) les circuits ont reçu des éléments d'asservissement des horloges (des PLLs ou des DLLs) qui permettent souvent de créer à partir d'une horloge d'autres horloges à des fréquences multiples de la fréquence de l'horloge incidente. Ces arbres d'horloge permettent donc de disposer sur le circuit de fonctions travaillant à des fréquences différentes. Afin de mieux percevoir tous les éléments et architectures précités nous allons présenter les composants Virtex de Xilinx et Apex d'Altera. Ces présentations sont succinctes et ne sont là que pour monter la diversité des architectures de FPGAs, ce qui nous permettra de mieux choisir la modélisation à apporter à ces dernières [41,47,48].

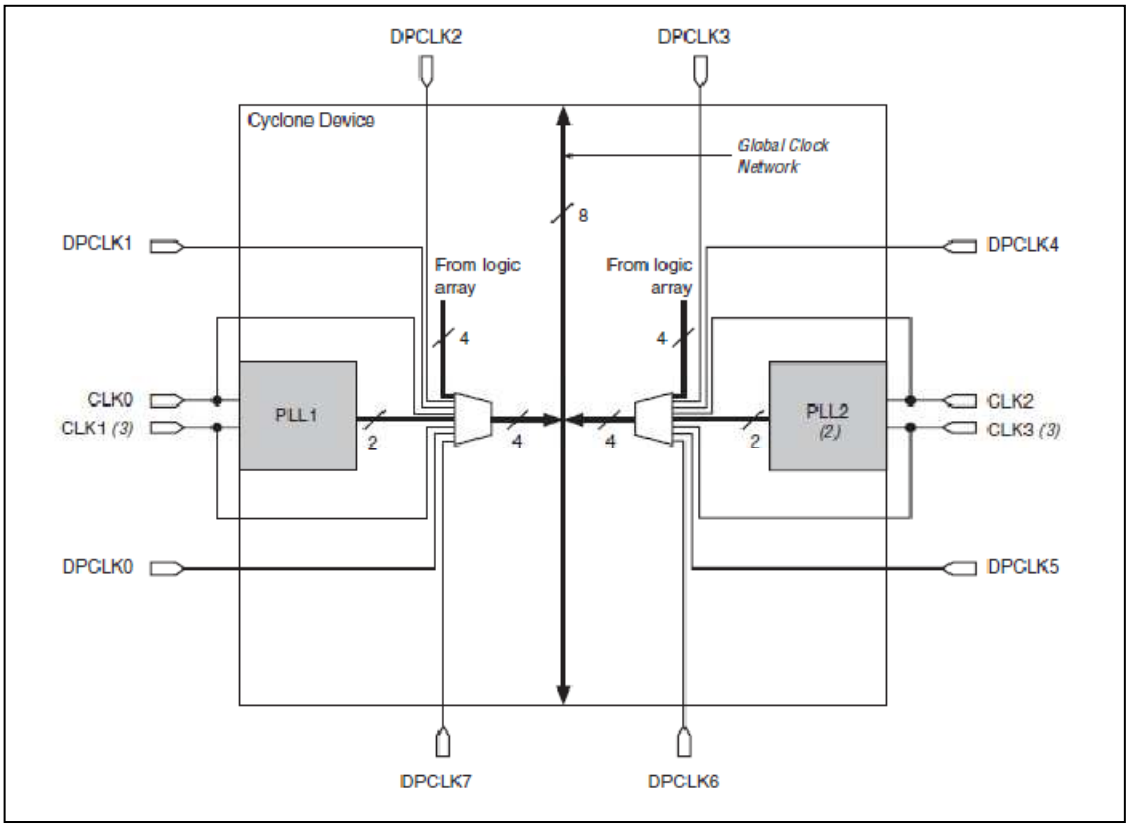


Figure II-13 : Élément de contrôle et d'acheminement des horloges (Cyclone II Altera) [55].

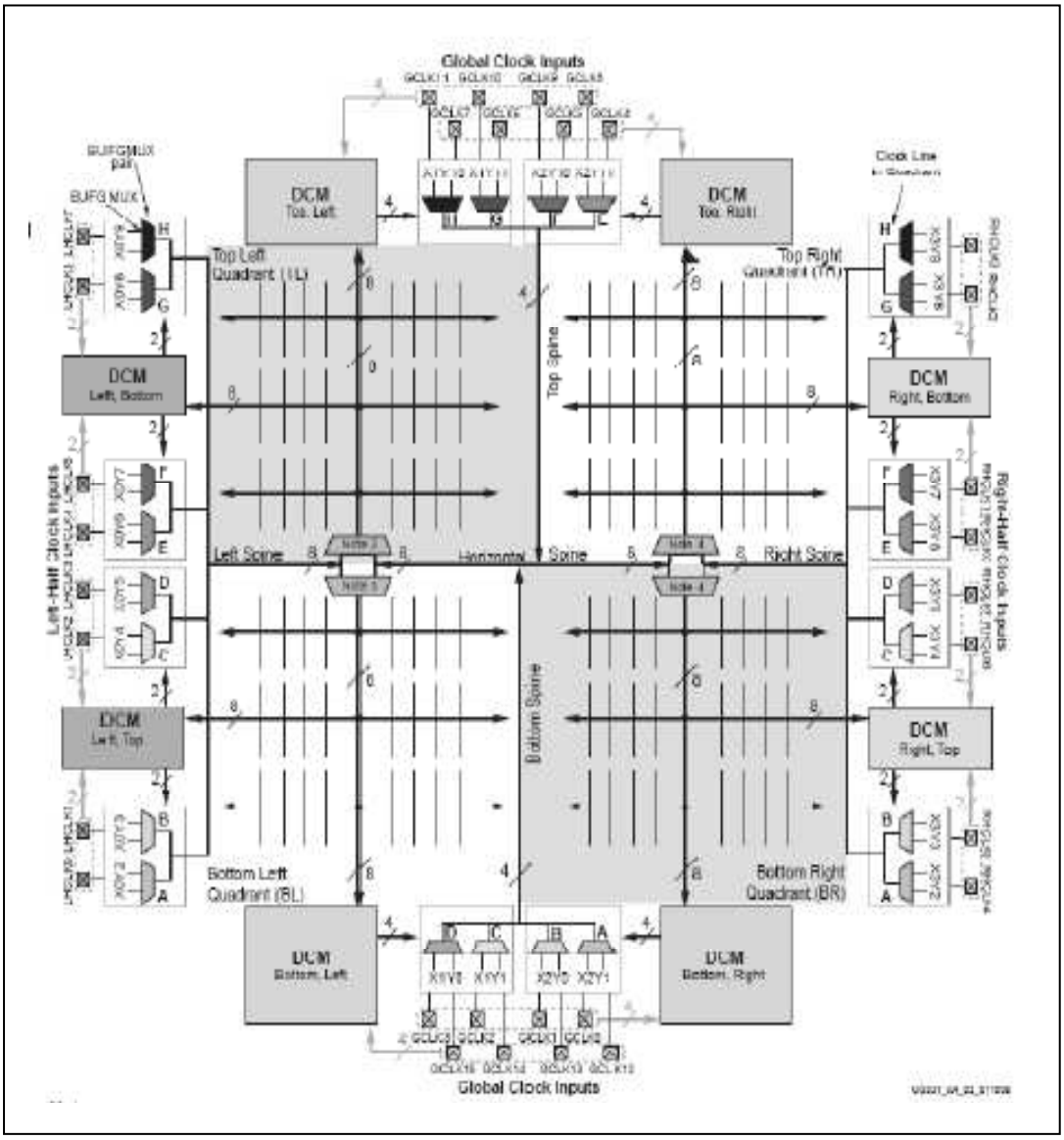


Figure II-14 : Élément de contrôle et d'acheminement des horloges (Xilinx Spartan) [38].

II.5.3. Ressources programmables embarqués

Depuis les années 2000, la densité d'intégration des circuits FPGA permet de regrouper, sur une même puce, une matrice d'éléments matériels (logiques, mémoires, opérateurs arithmétiques, entrées-sorties) configurables et un ou plusieurs systèmes à microprocesseurs. Ce type de circuit permet de profiter du parallélisme de calcul offert par l'architecture matérielle et du contrôle séquentiel efficace offert par le système programmable (microprocesseur). Aussi en tirant parti des propriétés respectives des systèmes programmables et des systèmes reconfigurables il est possible d'améliorer l'adéquation du

système global avec l'application développée. Dans ce cas l'utilisation de méthodes de conception conjointe logicielle/matérielle est indispensable et demande un effort important en développement d'outils.

Plusieurs architectures existent aujourd'hui, la figure II-15 illustre ces différentes possibilités. Dans certains circuits, la partie matérielle configurable et la partie programmable sont séparées par un bus spécifique. La partie programmable comprend le système à microprocesseur dans son ensemble : cœur de processeur, mémoires caches, périphériques, interface etc … Ce fut le cas du premier circuit commercial embarquant un cœur de processeur, le circuit Altera Excalibur qui regroupait une matrice FPGA APEX 20KE et un cœur de processeur ARM9 (32 bits) fonctionnement à 100MHz accompagné de deux fois 8kilo-octets de mémoire cache (instructions + données). Malheureusement, au moment de la sortie de ce composant, les outils n'étaient pas matures pour permettre une utilisation simple et efficace dans un contexte industriel.

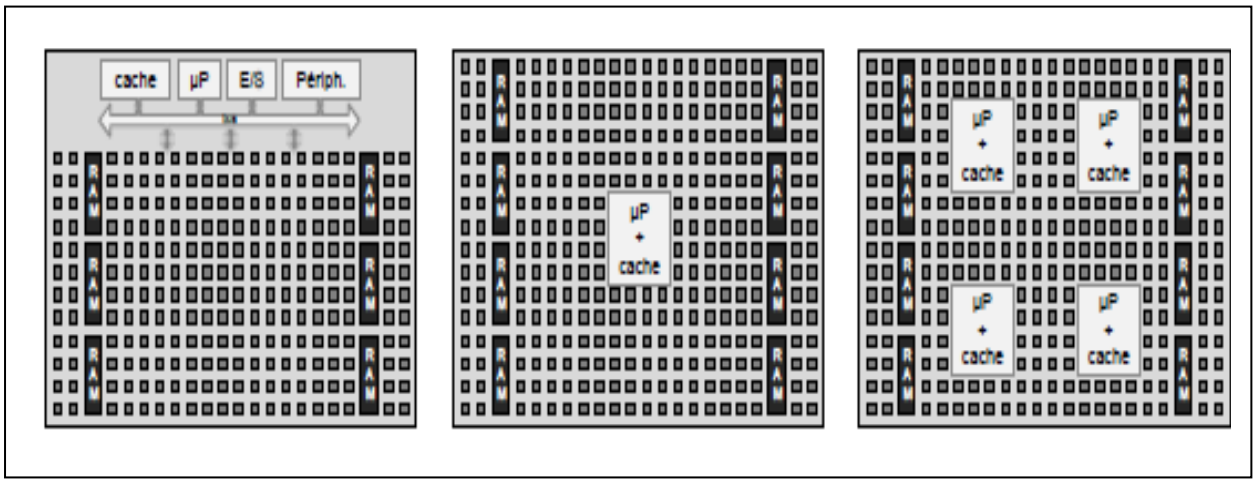


Figure II-15 : Trois architectures possibles de circuits mixtes FPGA microprocesseur(s) [42].

Certains circuits embarquent *profondément* au cœur de la matrice configurable le ou les processeurs. Ceux-ci ne sont pas nécessairement accompagnés de leur système, mais des outils logiciels permettent de configurer une partie de la logique pour constituer le système complet. Cela permet ainsi une plus grande flexibilité dans le choix du système programmable. C'est cette solution qui fut choisie par Xilinx pour son premier composant mixte Virtex-II Pro. Celui-ci était composé d'une matrice Virtex-II Pro et de un à quatre cœurs IBM PowerPC 405 (32 bits) cadencés à 400MHz disposant de deux fois 16 kilo-octets

de mémoire cache (instructions et données). Cette même architecture est toujours utilisée pour les composants Virtex de dernière génération.

Il faut noter qu'il est possible de réaliser un circuit mixte sans disposer physiquement d'un cœur de processeur embarqué dans le circuit FPGA. Dans ce cas l'utilisation d'un cœur synthétisable (dit *cœur soft*), fourni gratuitement par le fabricant de circuit, est une solution très efficace. Les cœurs synthétisables 32 bits Altera NIOS et Xilinx MicroBlaze sont par exemple très utilisés. Bien entendu, les performances de ces cœurs sont inférieures à celles des cœurs embarqués, mais ils permettent une plus grande flexibilité de configuration [41].

II.5.4. Ressources arithmétiques de gros grain

Nombreuses sont les applications qui nécessitent la synthèse d'opérateurs du type multiplieur, additionneur et multiplieur/accumulateur. S'il est possible, grâce aux chaînes de propagation rapide de la retenu de réaliser sur un petit nombre de LUT des additionneurs efficaces, ce n'est pas le cas pour des multiplieurs très coûteux en ressources. Les industriels ont donc choisi d'implanter de façon matérielle des multiplieurs reconfigurables (la reconfiguration intervient en particulier sur la taille des données à traiter) au sein même de la matrice de grain fin. En positionnant ces multiplieurs près des colonnes d'éléments mémoires et d'éléments reconfigurables de grain fin il est possible de synthétiser des opérateurs MAC (Multiplieur Accumulateur). Cette solution fut retenue par Xilinx pour les composants Virtex-II. Altera a choisi d'implanter dans les circuits de la famille Stratix des opérateurs câblés plus complexes pouvant directement être configurés en opérateurs MAC que l'on trouve également sur les nouvelles générations de FPGA Xilinx.

La figure II-16 donne le schéma (simplifié) d'un élément arithmétique configurable de dernière génération (DSP48E Slice) que l'on trouve dans les composants Xilinx Virtex-6. Ceux-ci comportent, entres autres, un multiplieur prenant en entrée un mot de 25 bits et un mot de 18 bits, une unité arithmétique (opération addition ou soustraction) et logique ainsi que des chaines de retour pour les calculs itératif (filtre IIR par exemple). En fonction du circuit choisi le nombre de DSP48E Slice est plus ou moins grand, il atteint 2016 éléments pour le plus gros des circuits Virtex-6 SXT qui est une version spécialisée pour le traitement numérique du signal [41].

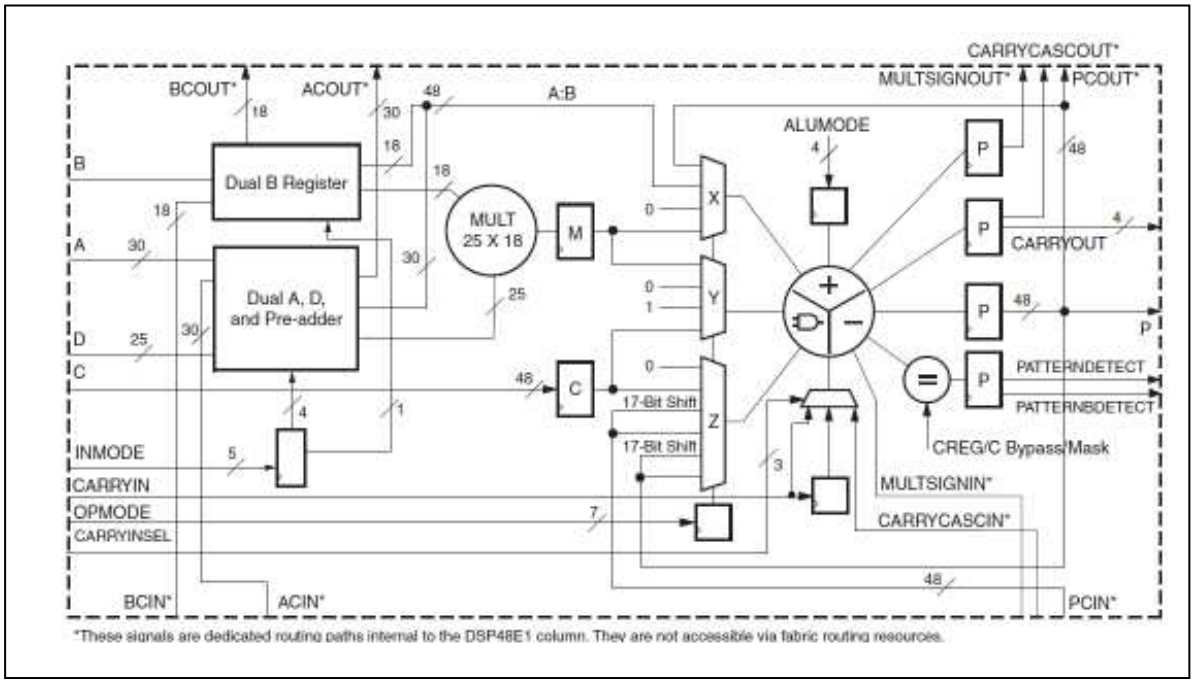


Figure II-16 : Elément arithmétique configurable du circuit Xilinx Virtex6 [42].

II.6. Le système Excalibur d'Altera

Excalibur est un SOPC (System On a Programmable Chip). Il comporte un cœur de processeur, de la logique et de la mémoire fournis sous forme d'IP qui seront placé sur un circuit de type FPGA de la famille APEX d'Altera. Excalibur propose deux types de cœur de processeur :

- Le NIOS est un processeur de type **firm** c'est à dire exclusivement dédié à la famille APEX d'Altera. C'est un processeur 32 ou 16 bits configurable. La performance maximale est de 50 Mips. L'intérêt de ce cœur est qu'il est gratuit, qu'il pourra suivre les évolutions technologiques d'Altera, la souplesse de la solution.
- Le ARM922T est un processeur RISC 32 bit de chez ARM. Dans sa version Excalibur, il est directement intégré à un circuit APEX. Cette version **Hard** permet de garantir des performances de 200 Mips tout en ayant optimisé la surface nécessaire.

On bénéficie aussi de tout le support et l'expérience existant autour de cette famille de processeurs embarqués les plus utilisés dans le monde industriel.

II.6.1. Architecture Nios

Le NIOS comprend un CPU et des périphériques. Il peut être configuré en fonction des besoins et performances visés. Il prendra alors de 2% à 20 % de la surface totale selon le type de circuit cible (Typiquement 12% pour un processeur 32 bits dans un circuit de 200000 portes EP20K200E) [6,7].

- **Le CPU**

Le processeur est entièrement synchrone, son architecture interne de type Berkeley est celle de la figure ci-après

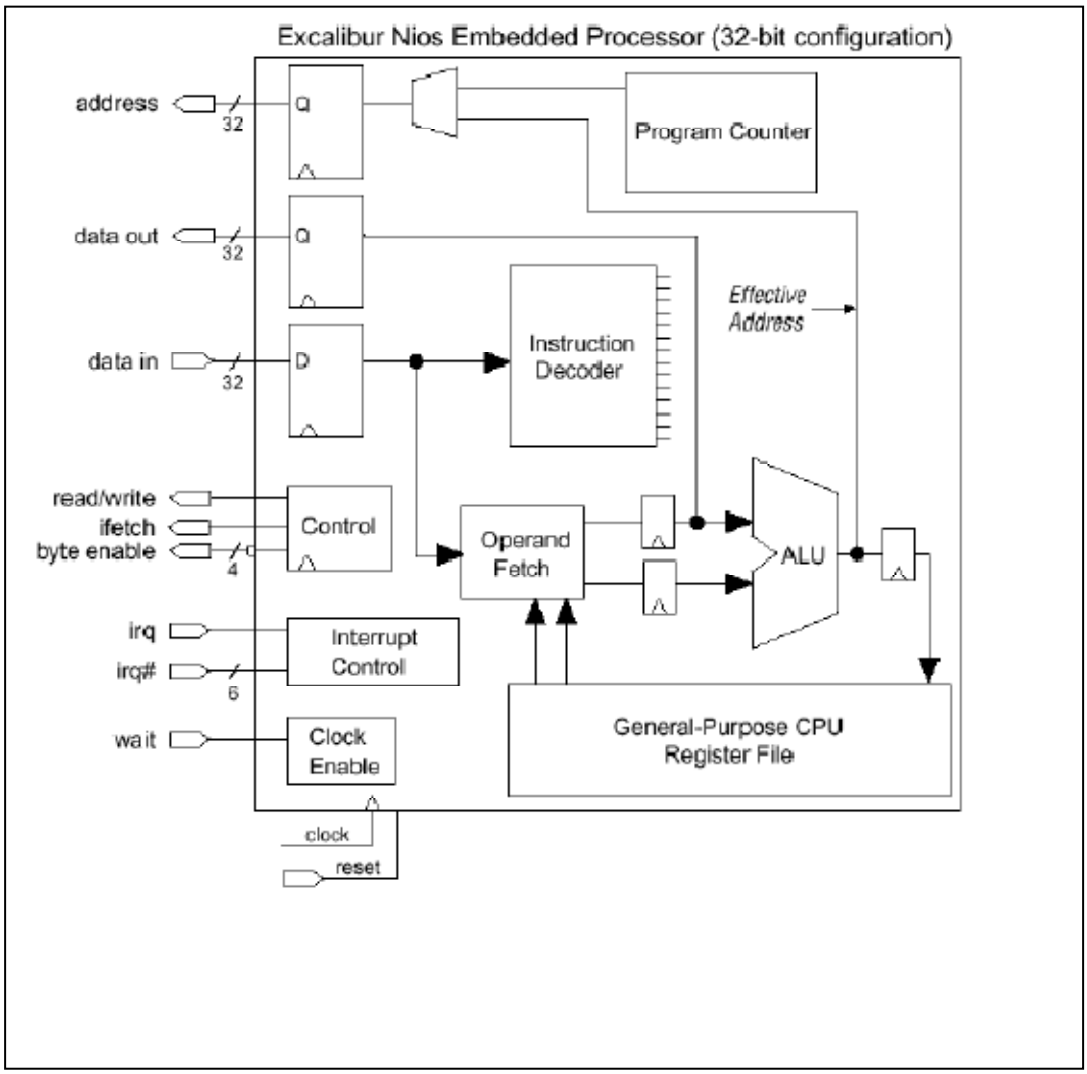


Figure II-17 : Architecture interne processeur de type Berkeley [6].

Ce que l'on peut configurer :

- La taille du bus de données 16 ou 32 bits
- Le bus adresse : de 10 à 33 bits
- La taille de la banque de registre (File Registre) : 128, 256 ou 512
- L'adresse du RESET
- La table d'adressage des vecteurs d'exceptions
- Le nombre de décalages en un seul cycle d'horloge : 1, 3, 7, 15 ou 31
- Une multiplication câblée ou non

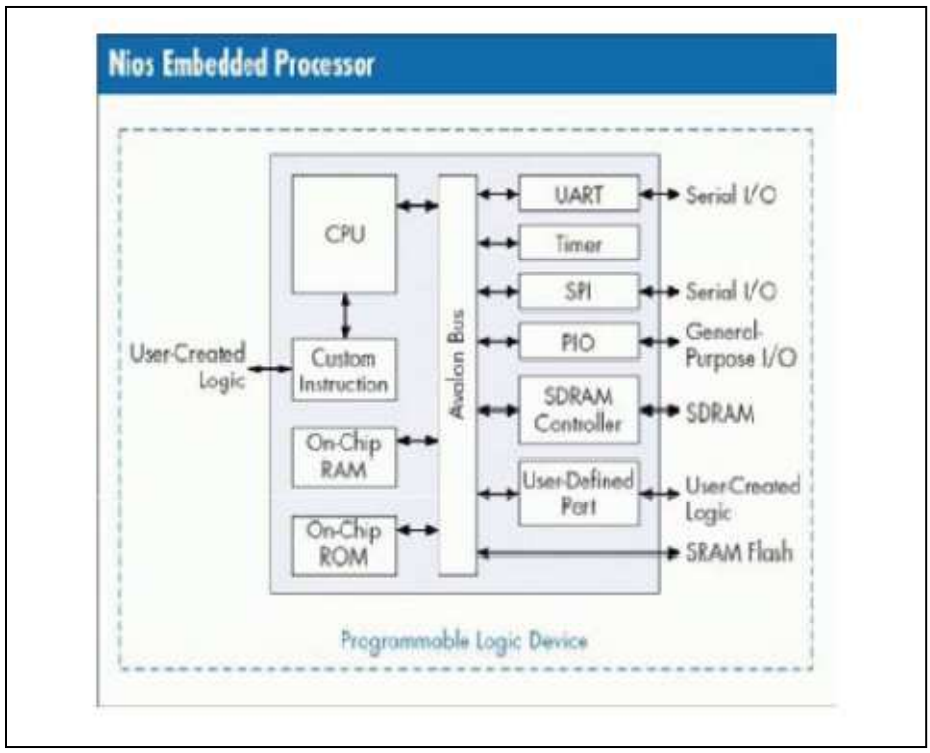


Figure II-18 : Processeur embarqué NIOS d'Altera [7].

II.6.2. Processeur NIOS II

Le processeur « softcore » NIOS II, d'Altera, est basé sur une architecture RISC. Ce processeur existe en trois versions : rapide NIOSII/f (fast), économique NIOSII/e et standard NIOSII/s. C'est un cœur de processeur configurable (taille des instructions, nombre de registres, ajout de périphériques, etc.) à volonté selon les besoins. Il possède un nombre paramétrable de différents périphériques (voir figure 4.13) à savoir : mémoire RAM/ROM,

UART pour gérer une ligne série, timer, PIO (Parallel Input Output), etc. Hormis ces périphériques implémentés par Altera, l'utilisateur a la possibilité d'en ajouter qu'il aura lui-même écrits. Ils sont vus par le NIOS comme de simples emplacements mémoires accédés autant en lecture qu'en écriture avec la possibilité de gestion d'interruptions. Ce processeur embarqué est optimisé pour la logique programmable d'Altera et l'intégration système sur un circuit programmable (SOPC). Avec l'outil de développement de système SOPC Builder, les concepteurs peuvent adapter le processeur NIOS et ses périphériques pour créer le système exact dont ils ont besoin.
L'architecture interne du processeur NIOS est conçue pour fournir plusieurs avantages tels que :

Implémentation efficace dans les composants FPGA d'Altera :

- Nombre minimal d'éléments logiques.
- Utilisation du minimum de mémoire.
- Fréquence d'horloge maximale.

Intégration de système sans effort :

- Interface mémoire simple.
- Placement de périphériques configurables standard.
- Outil SOPC Builder qui crée la logique d'interface du bus Avalon entre l'unité centrale de traitement, les périphériques et la mémoire.

Étant donné que le code source de ce processeur est non accessible, la seule méthodologie de conception qu'on peut appliquer est celle de réplication. Cette méthodologie est mieux expliquée dans le paragraphe suivant. L'adaptation de NIOS aux besoins de mppSoC est aussi détaillée [7,8].

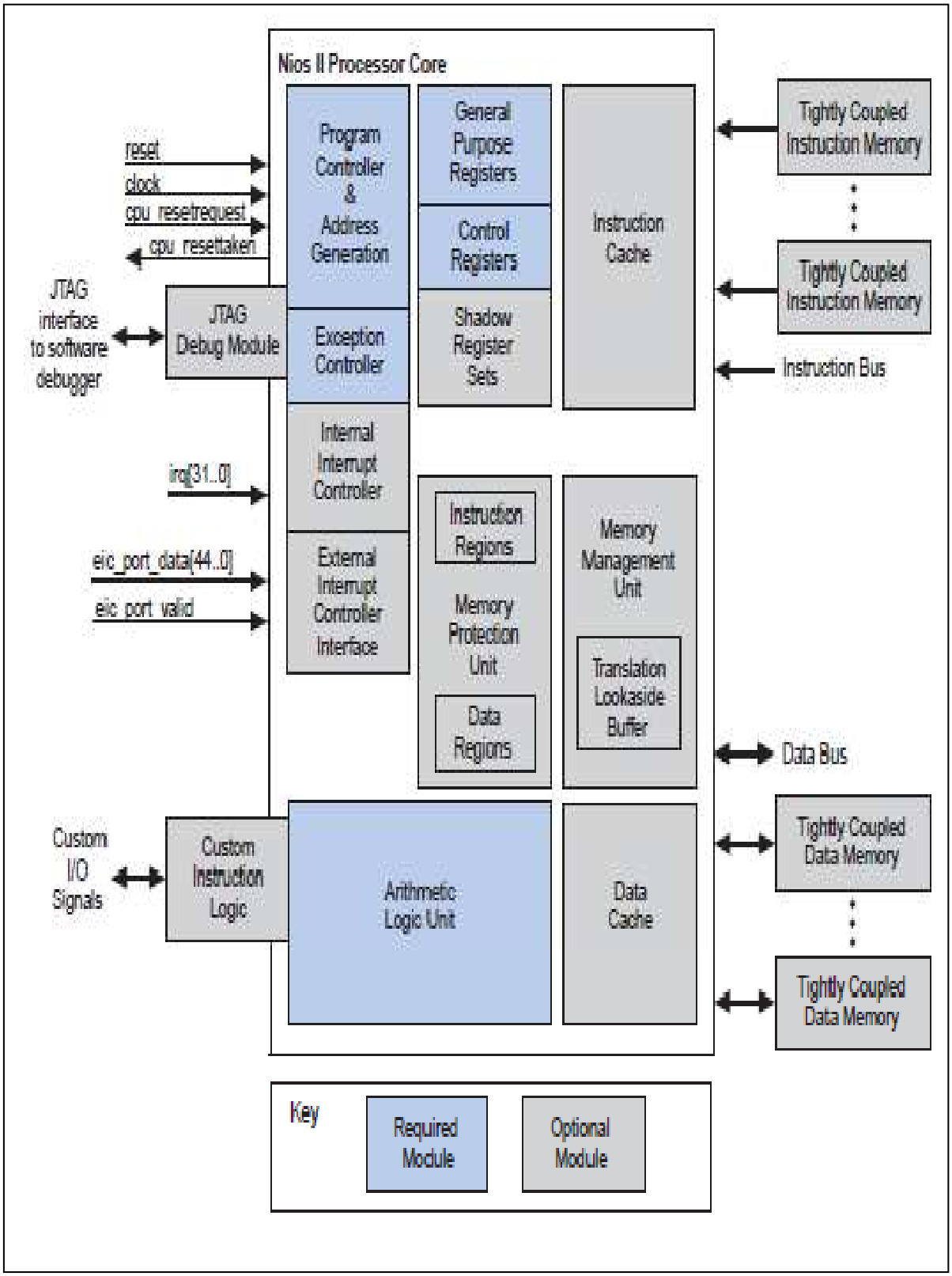


Figure II-19 : Schéma block du processeur NIOS II [55].

II.7. Exemples des Cartes (FPGA) de développement

II.7.1. Carte de développement et de formation DE2(Altera)

La carte DE2 (figure II-20), construite autour d'un FPGA ALTERA Cyclone II (35000 Logic Elements ou LEs), comporte de la mémoire FLASH, SRAM et SDRAM, et de nombreux périphériques d'affichage (LEDs, LCD, VGA, TV), sonores et de communication (Ethernet, USB, IrDA...) [8].

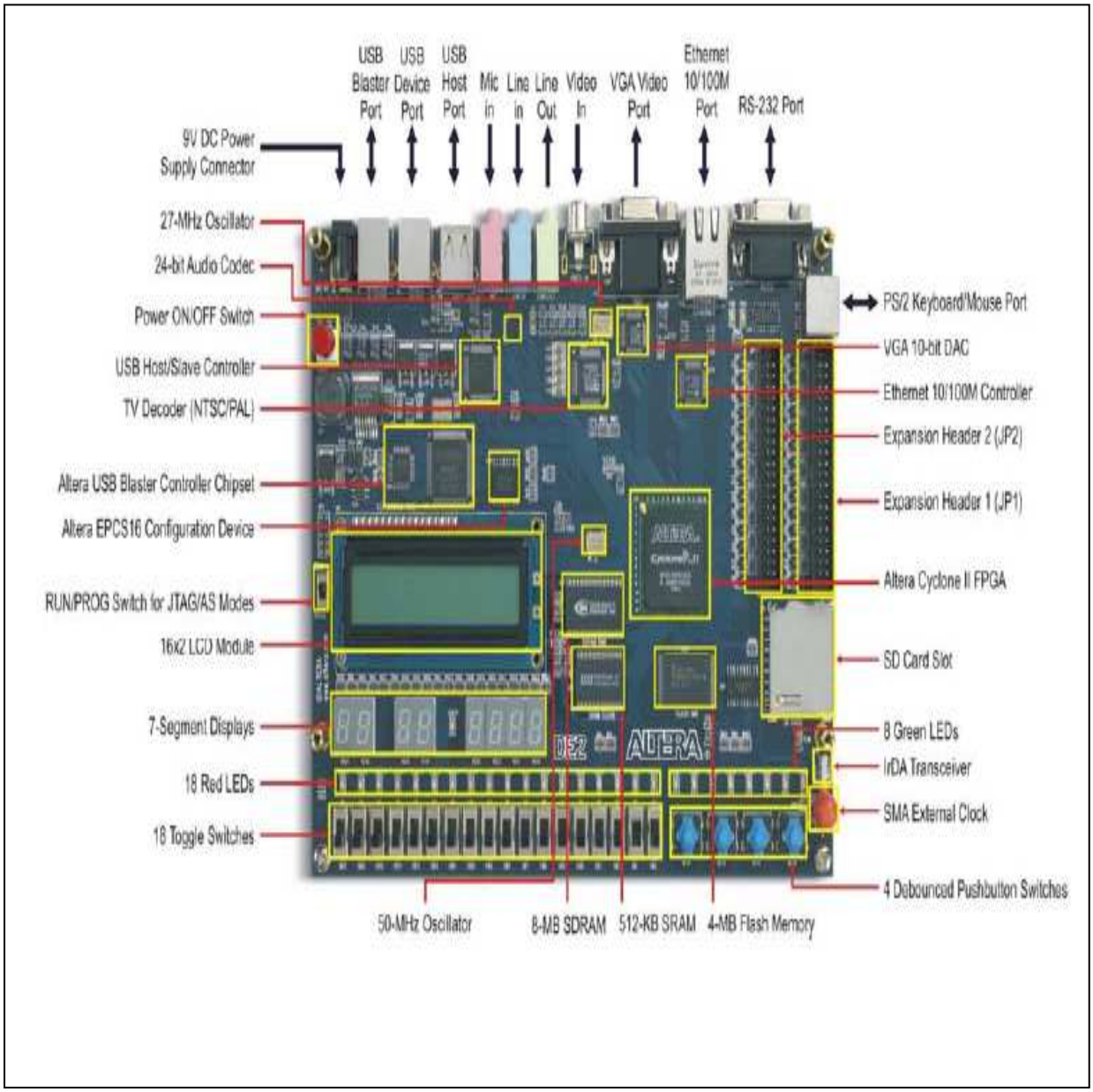


Figure II-20 : Carte de développement et de formation DE2(Altera) [8].

Caractéristiques FPGA EP2C35F672C6 Cyclone II :

- USB Blaster intégré pour configuration FPGA
- Ethernet 10/100, RS-232, port IR
- Sortie vidéo (CNA 10 bits VGA)
- Entrée vidéo (NTSC/PAL/Multi-format)
- USB 2.0 (type A et type B)
- Port PS/2 pour clavier ou souris
- Entrée ligne, sortie ligne, entrée micro (CODEC audio 24 bits)
- Embases d'extension (76 broches de signal) Mémoire :

o 8 Mo de SDRAM, 512 Ko de SRAM, 4 Mo de Flash

o Port de carte mémoire SD Commutateurs, DEL, affichages et horloges :

o 18 commutateurs à bascule

- 4 commutateurs à bouton-poussoir antiparasités
- 18 DEL rouges, 9 DEL vertes
- Huit affichages à 7 segments
- Écran LCD 16 x 2
- Oscillateurs 27 MHz et 50 MHz, entrée d'horloge SMA externe

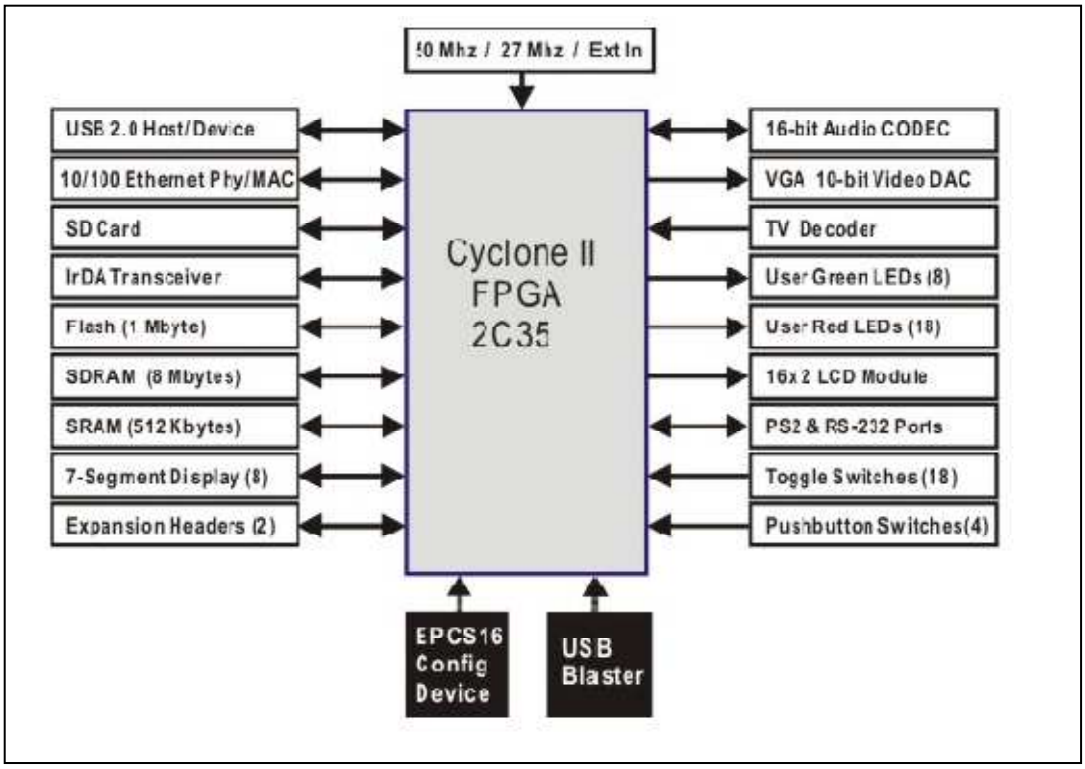


Figure II-21 : Synoptique de la carte DE2 [8].

II.7.2. Carte d'étude BASYS2 – Digilent (Xilinx)

Est une plateforme de conception et de mise en œuvre de circuits que quiconque peut utiliser pour acquérir de l'expérience en construction de circuits numériques réels. Elle repose sur le FPGA (réseau logique programmable) Spartan-E3 de Xilinx et un contrôleur USB AT90USB2 d'Atmel. La carte Basys2 fournit un matériel complet et prêt à l'emploi convenant à l'hébergement de circuits allant de dispositifs logiques de base à des contrôleurs complexes. Un large éventail de périphériques E/S intégrés et tous les circuits de support FPGA nécessaires sont inclus, de sorte à pouvoir créer d'innombrables dessins sans avoir recours à d'autres composants.

Quatre connecteurs d'extension standard permettent de développer les conceptions au-delà de la carte Basys2 en utilisant des plaques d'essais, des cartes de circuit conçues par les utilisateurs ou des Pmods. Les Pmods sont des modules d'E/S analogiques et numériques peu coûteux qui offrent la conversion A/N et N/A, des pilotes de moteur, des entrées de capteur et de nombreuses autres fonctionnalités. Les signaux sur les connecteurs à 6 broches sont protégés contre les dommages DES et les courts-circuits, assurant une longue durée de vie dans tout environnement. La carte Basys2 fonctionne sans heurts avec toutes les versions des outils Xilinx ISE, y compris le WebPack gratuit. Elle est livrée avec un câble USB qui fournit l'alimentation et une interface de programmation, de sorte qu'aucun autre bloc d'alimentation ou câble de programmation ne soit nécessaire [22].

- **Caractéristiques :**

- FPGA Spartan 3-E Xilinx, 100 K portes logiques.
- Le FPGA dispose de multiplicateurs 18 bits, de 72 Kbits de bloc de mémoire RAM double-port et d'une fréquence de fonctionnement de plus de 500 MHz.
- Port USB 2.0 pleine vitesse pour la configuration FPGA et les transferts de données (en utilisant le logiciel Adept 2.0).
- Mémoire ROM Flash de la plateforme XCF02 qui stocke les configurations FPGA indéfiniment.
- Fréquence de l'oscillateur réglable par l'utilisateur (25, 50 et 100 MHz), plus support pour un deuxième oscillateur.
- Trois régulateurs de tension intégrés (1,2 V, 2,5 V et 3,3 V) qui permettent l'utilisation de sources d'alimentation externes de 3,5 V à 5,5 V.

- 8 DEL, affichage à 7 segments et 4 chiffres, quatre boutons-poussoirs, 8 interrupteurs à glissière, port PS/2 et port VGA 8 bits.
- Quatre embases à 6 broches pour les E/S utilisateurs et cartes de circuit des accessoires PMOD de Digilent.
- Requiert le logiciel Adept 2.0 ou supérieur.

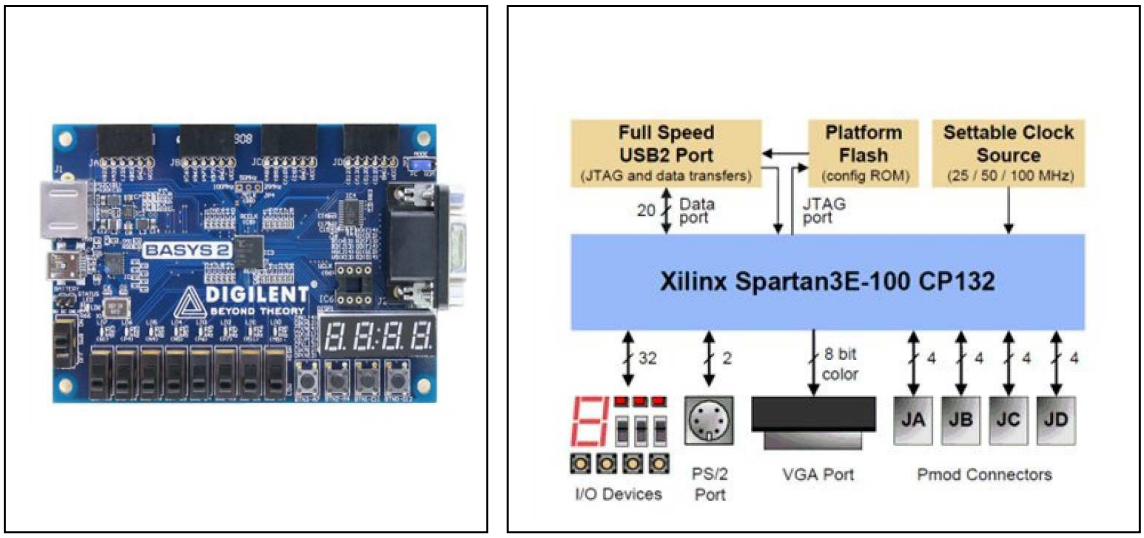
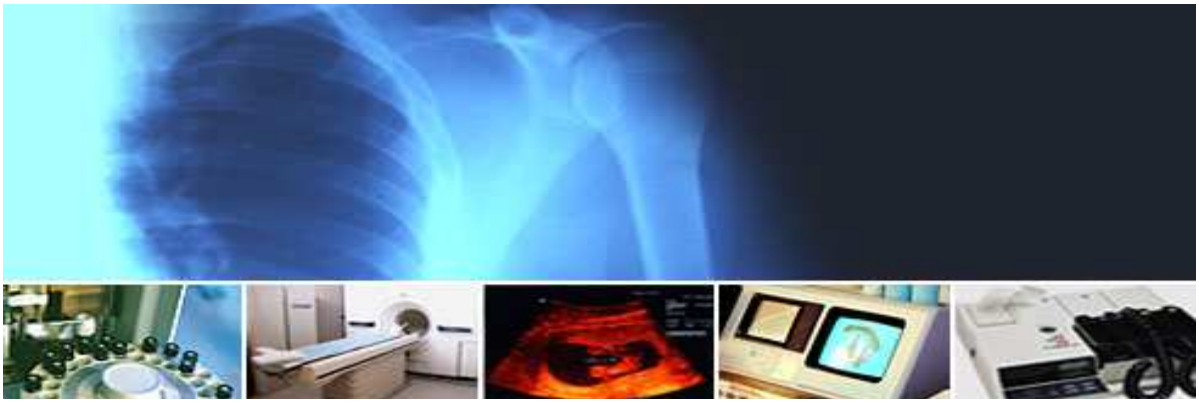


Figure II-22 : Carte d'étude BASYS2 – Digilent (Xilinx) avec le Synoptique [22].

II.8. Principales applications des FPGA

Ici, nous mettre en lumière certaines applications clés où la conception avec les FPGA peut être un avantage concurrentiel.

II.8.1. Applications Médical



La majorité des produits médicaux ont un certain type de semi-conducteurs en elle. Au fil des ans, le contenu des semi-conducteurs continue d'augmenter dans ces myriades de produits. Dispositifs logiques programmables (FPGA) continuent à voir un taux d'adoption plus élevé que d'autres semi-conducteurs. C'est parce que FPGA offrent une alternative viable et puissante à la fois dans le développement du matériel médical. FPGA éliminé les coûts initiaux NRE et les quantités minimales de commande associés aux circuits ASIC et les risques coûteux de multiples itérations de silicium par sa capacité reprogrammable.

Les FPGAs offrent la flexibilité et le conseil possibilités d'intégration de conception de différencier contre les fabricants de matériel médical concurrents. En outre, FPGAs peuvent être mis à niveau dans le domaine de l'évolution des normes ou exigences de changement. En outre, la possibilité de réutiliser une plate-forme matérielle commune permet aux concepteurs de créer des systèmes différenciés qui prennent en charge une variété d'ensemble de fonctionnalités avec un design de base, ce qui entraîne une réduction des coûts de fabrication. Qu'il s'agisse de concevoir une machine CT ou de l'équipement de surveillance des patients, la logique programmable est un chemin souple et à faible risque pour une conception réussie du système. Logique programmable vous permet d'offrir des économies de coûts optimaux tout en offrant à valeur ajoutée capacités de différenciation par rapport à d'autres fabricants d'équipements médicaux.

Altera fournit un produit complet et le portefeuille d'outils pour accélérer application médicale développement Altera ® SDK pour OpenCL ™ , des conceptions de référence industrielles, le Quartus II et outil d'intégration de systèmes Qsys, NiosII processeur intégré 32-bit , mémoire embarquée, un vaste ensemble de broches d'E / S, et un large éventail de la propriété intellectuelle standard et médical (IP) de Altera et nos partenaires.

La configurable Nios II processeur embarqué vous permet de créer facilement un système basé sur un microprocesseur qui est personnalisé pour répondre à vos exigences d'application. L'outil d'intégration de systèmes Qsys automatise la configuration IP et d'intégration. Altera offre plus de 200 cœurs d'IP que vous pouvez utiliser dans le commerce pour créer un système matériel en quelques minutes. Avec le logiciel de conception Quartus II, vous pouvez réduire la puissance, la vitesse d'optimiser et de réduire les temps de compilation du système dans votre conception de FPGA.

II.8.2. Application Militaire

Aéronautique et designers militaires font face à une longue liste de défis, y compris la nécessité pour la haute performance, une large plage de fonctionnement, et une long durée de vie du système , ainsi que les contraintes ajoutées de la taille limitée du système et les budgets de puissance. FPGA Altera permettent aux concepteurs de répondre aux exigences de l'industrie aérospatiale et les marchés militaires.

II.8.3. Applications Wireline

Avec le développement de nombreuses applications multimédia gourmandes en bande passante d'aujourd'hui, il ya une augmentation massive de la demande des clients pour plus de bande passante. Cela impose un fardeau énorme sur les fournisseurs de services de faire évoluer tous les segments de leurs réseaux filaires rapidement pour soutenir cette énorme demande de trafic en voix, vidéo et données. Les innovations sont constamment se déroulent dans l'accès, la transmission et les arènes d'équipement de réseau à l'ensemble, transporter et livrer le trafic triple-play sur des réseaux convergents multiservices. Ces technologies innovantes nécessitent une plate-forme flexible qui offre des solutions rapidement, tout en ayant une voie d'évolution constante, à faible coût pour la réussite de déploiement de production haut volume. Avec un portefeuille complet de haute performance, les FPGA à faible coût et une migration sans risque à faible coût HardCopy ® ASIC technologie, dispositifs logiques programmables d'Altera (PLD) offrent la bande passante de traitement et la flexibilité nécessaires pour la conception d'infrastructures filaires. Pour accélérer le délai de commercialisation, Altera fournit également un large éventail de la propriété intellectuelle (IP), des modèles de référence , et compléter les solutions des partenaires via Altera ® écosystème des services filaires. D'Altera 28 nm silicium direction , des outils de soutien et solutions permettent un niveau élevé d'intégration de système pour les plates-formes d'infrastructure de communication filaires.

II.8.4. Application Sans fil

La croissance explosive de l'Internet au cours de la dernière décennie a conduit à une demande croissante pour l'accès à Internet haute vitesse omniprésente. Accès à large bande sans fil mobile répond à ce besoin croissant en offrant un accès à tout moment, n'importe où Internet à la maison et le bureau. Il existe de nombreuses applications sans fil qui répondent largement aux besoins différents pour un accès continu, pratique et flexible pour la mise à jour des informations via le web.

Communication cellulaire mobile a évolué à partir du système analogique traditionnel à un système numérique 4G d'aujourd'hui basée sur la fréquence accès multiple par répartition orthogonale (OFDMA) et multiple-input multiple-output (MIMO). Pour les services de données à haute vitesse, + normes 3G ont été déployés pouvant atteindre significativement plus élevés des débits de pointe par rapport aux technologies 3G telles que W-CDMA et CDMA2000. Par exemple, les technologies d'accès à haut débit de paquets en liaison descendante (HSDPA) et l'accès par paquets en liaison montante haut débit (HSUPA) augmentent la liaison descendante et des débits de données en liaison montante du W-CDMA à 14,4 Mbps et 5,6 Mbps, respectivement. Réseaux de prochaine génération basée sur le 3 ème projet de partenariat de la génération (3GPP) standard de l'évolution à long terme (LTE) sont actuellement prototypé et préparés pour des essais sur le terrain.La figure II-23. montre le taux de différentes applications sans fil et la mobilité des données.

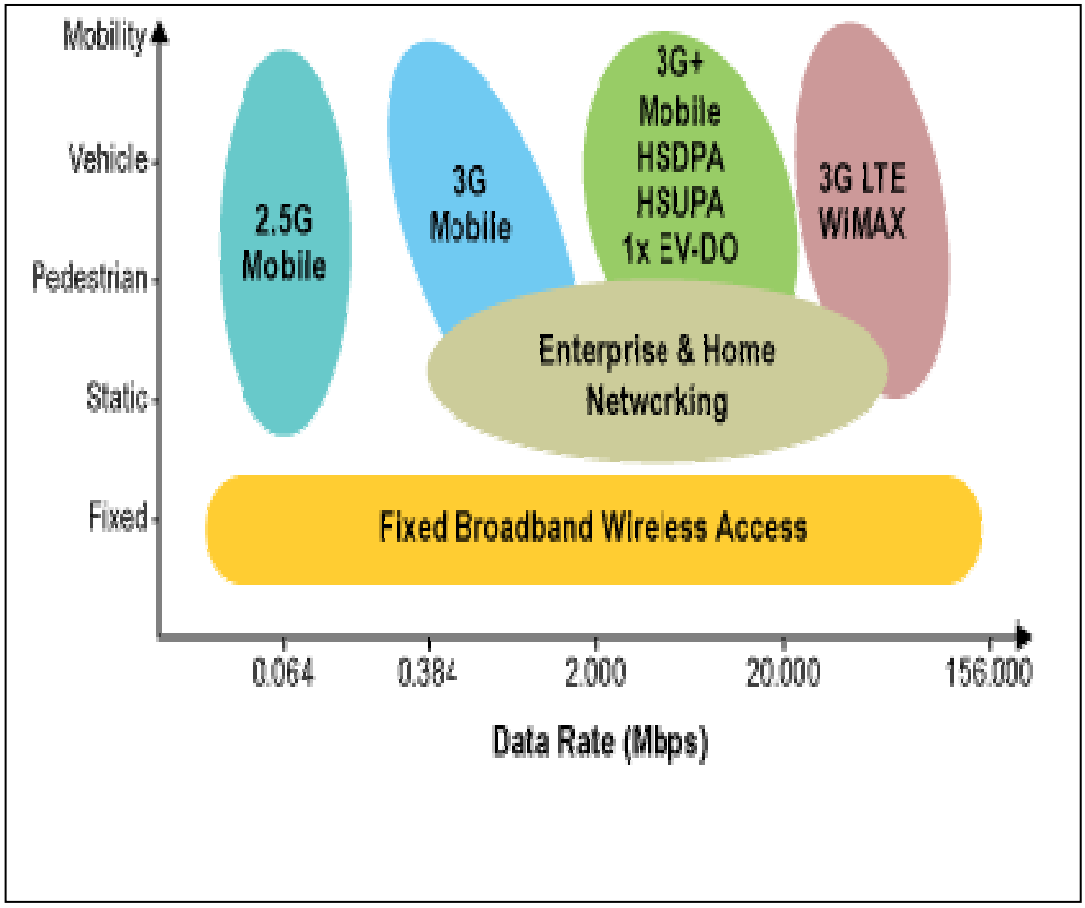


Figure II-23 : Le taux de différentes applications sans fil et la mobilité des données [8].

Worldwide Interoperability for Microwave Access (WiMAX) gagne en popularité en tant que technologie d'accès sans fil à large bande avec un potentiel de marché important. WiMAX 802.16e-2005 prend en charge un accès Internet haut-débit mobile dans un réseau métropolitain (MAN), et utilise des systèmes avancés de traitement du signal tel que la fréquence accès multiple par répartition orthogonale (OFDMA) et de la technologie à entrées multiples sorties multiples (MIMO). WiMAX 802.16e-2005 jouera un rôle clé dans les déploiements vert-terrain et les marchés émergents comme la technologie sans fil fixe.

Avec les avancées de la technologie des semi-conducteurs et des techniques de traitement du signal , les normes sans fil et les systèmes sont en constante évolution. Une plate-forme matérielle, qui peut fournir une bande passante élevée de traitement, la flexibilité et un avantage time-to-market est nécessaire pour répondre à ces exigences. Avec un portefeuille complet de haute performance , à faible coût des FPGA et un chemin de migration sans risque à faible coût HardCopy ® ASIC technologie, Altera ® dispositifs logiques programmables (PLD) fournir la bande passante de traitement et la flexibilité nécessaires pour des

conceptions de l'infrastructure sans fil. Pour accélérer le time-to-market, Altera fournit également un large éventail de la propriété intellectuelle (IP) , des modèles de référence , et compléter les solutions des partenaires off-the-shelf via l'écosystème sans fil Altera. 40-nm silicium leadership, des outils de soutien et des solutions d'Altera permettent désormais système hautement intégré sur une puce (SoC) pour les plates-formes d'infrastructure sans fil.

II.8.6. Véhicules électriques

Les équipementiers et les constructeurs automobiles utilisent FPGA et CPLD pour différencier leurs voitures puissantes, avec des plates-formes rentables conception flexible qui répondent à la performance, la qualité, le cycle de vie et les besoins d'évolutivité de leurs systèmes numériques de plus en plus complexes. Avec les produits de l'automobile, vous pouvez réduire les coûts du système, améliorer la fiabilité, et de simplifier la complexité de conception pour accélérer les délais de commercialisation.

Apprenez comment FPGA de classe automobile, comme faible coût, de faible puissance Cyclone série, peuvent vous aider à résoudre certains de vos plus grands défis de conception d'applications automobiles.

Véhicules électriques hybrides (HEV) et les véhicules électriques (EV) avec leurs moteurs électriques, la conversion de puissance et les systèmes de gestion de batterie apporter un besoin croissant pour une durée de vie de la batterie et une plus grande efficacité.

Les boucles de régulation rapides de leurs systèmes de contrôle du moteur doivent haute efficacité énergétique à travers toutes les gammes de couple et de vitesse pour une plus longue portée conduite. Cela nécessite un contrôle algorithme de traitement de signaux vectoriels complexes que les performances du microcontrôleur classique ne peuvent pas soutenir. FPGA programmables, cependant, peuvent accélérer les boucles de régulation pour augmenter les performances du système.

Vous pouvez utiliser les FPGA dans les applications où le traitement du signal numérique (DSP) améliore les performances du système, tels que le AC / DC, des systèmes de gestion de batterie, convertisseurs DC / DC, et systèmes d'onduleurs moteur comme indiqué sur la Figure II-24 [8].

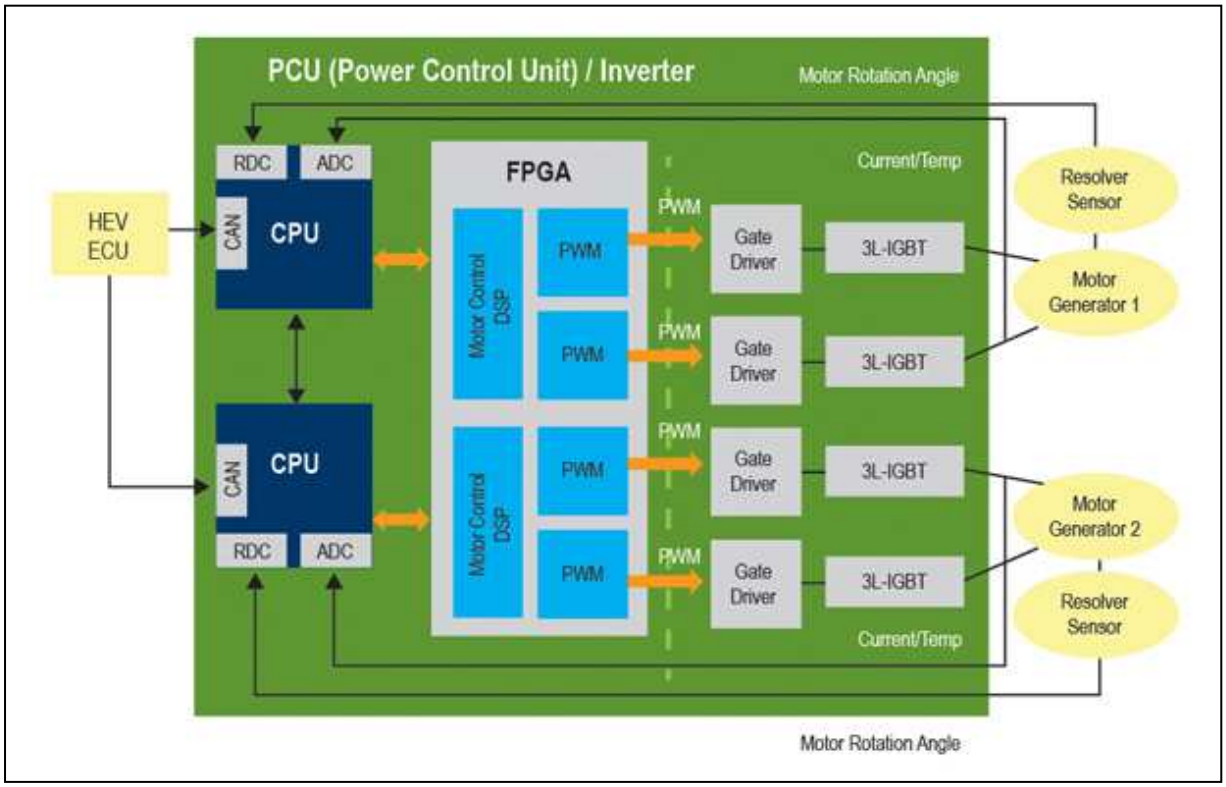


Figure II-24 : Unité de contrôle et commande du véhicule électrique [8].

II.9. Tendances

Quelques tendances apparaissent ces dernières années dont l'objectif est d'augmenter l'efficacité des FPGA. Sans vouloir être exhaustif nous pouvons en décrire quelques unes. Au niveau de la mémoire de configuration, les densités de configuration et la maitrise des technologies ont permis l'émergence de composants de technologie mixte FLASH-SRAM, tel que les composants Lattice XP2 et Xilinx Spartan AN. L'offre commerciale a évolué en proposant de plus en plus des séries spécialisées à l'intérieur d'une même famille de composants.

II.10. Conclusion

La part de marché des FPGA dans le marché globale des circuits matériels pour l'électronique numérique ne cesse d'augmenter. Les évolutions technologiques et architecturales qui ont eu successivement lieu depuis le début des années 2000 ont fait de ces circuits de réels et rentables alternatives aux classiques ASIC. Avec ces évolutions c'est tout un nouveau domaine de l'électronique numérique qui s'est ouvert. Aujourd'hui les FPGA sont utilisés

dans tous les domaines, des systèmes embarqués aux systèmes de communications, ils sont au cœur d'un important champ de recherche académique et industrielle.

CHAPITRE III

Méthodologie de développement d’un algorithme de commande pour l'implantation sur puce FPGA

III.1. Introduction

L'implantation des algorithmes de contrôle dans leur intégralité sur des cibles matérielles telles que les FPGA est une démarche qui nécessite une parfaite maîtrise des processus de conception et un travail spécifique d'adéquation entre l'algorithme et l'architecture de commande à intégrer. Donc, un savoir faire méthodologique est nécessaire aux concepteurs utilisant les composants de type FPGA afin de satisfaire l'ensemble des contraintes inhérentes de l'implantation, tout en apportant une flexibilité de développement suffisante. Par ailleurs, les applications de commande sont des applications qui décloisonnent un savoir faire dans plusieurs domaines. En effet, cela nécessite du concepteur la maîtrise d'un savoir faire dans les domaines de la microélectronique, de l'électronique faible et grande puissance, des machines électriques et de leur commande. Cette difficulté pousse les concepteurs à préférer les implantations standards des solutions logicielles. Par conséquent, l'implantation des algorithmes de contrôle sur de nouvelles solutions matérielles tels que les FPGA doit suivre des étapes bien déterminées afin de guider le concepteur et faciliter le processus de conception [49].

III.2. Description d'un système de commande

Vu la complexité et la diversité des systèmes de commande, il est difficile de définir d'une manière universelle une structure générale pour de tels systèmes. Cependant, en ayant une réflexion par rapport aux éléments les plus communément rencontrés dans ces systèmes, il est possible de définir au mieux une structure générale d'un système de commande comme la suit [50].

III.3. Structure générale d'un système de commande

III.3.1. Commande en Boucle Fermée

Sont des systèmes de commande en Boucle Fermée, sont constitués, dans la plupart des cas, de la façon suivante :

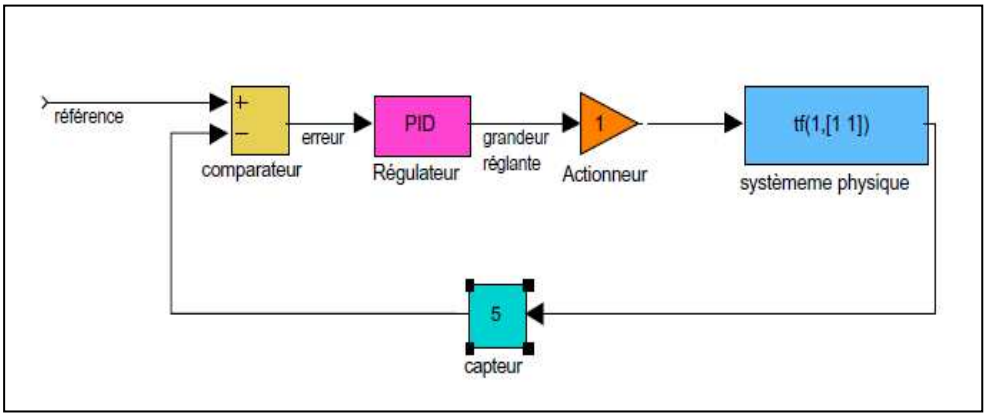


Figure III-1 : Structure générale d'un système de commande [51].

Les fonctions principales dans la boucle sont :

- **Le système physique (ou processus) :** il génère la variable que l'on désire asservir.
- **L'actionneur (organe de puissance) :** il peut être inclus dans le système physique à asservir.
- **Le capteur :** il réalise la mesure de la grandeur commandée.
- **Le comparateur :** il calcule la différence entre la grandeur désirée et la grandeur obtenue (c'est à dire l'erreur).
- **Le régulateur :** c'est l'organe de commande : son rôle consiste à ajuster l'action à partir de l'erreur, Il élabore la variable qui va agir et commander l'actionneur.
- **Les perturbations :** ce sont des modifications non prévisibles sur le système.

Le choix du régulateur étant lié à la caractéristique "entrée sortie" du système à asservir, il sera tout d'abord nécessaire d'étudier le système physique à asservir et d'en établir un "modèle mathématique".

III.3.2. Commande en Boucle Ouverte

L'objectif est de trouver un modèle, juste ou approché, du système physique à asservir. Nous ouvrons la boucle dans le schéma bloc. Les paramètres accessibles nous donnent l'entrée et la sortie possibles du système [51] :

- **entrée :** grandeur régulant.

- **sortie :** image de la grandeur de sortie (généralement à travers le capteur).

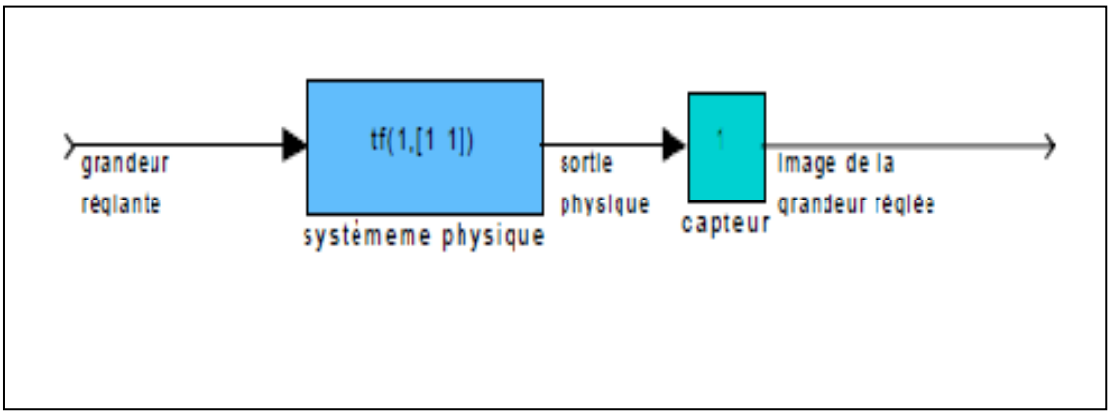


Figure III-2 : schéma bloc en boucle ouverte [51].

Un exemple d'une structure générale du commande d'une machine électrique est constitué de quatre parties comme indiquer la figure suivant :

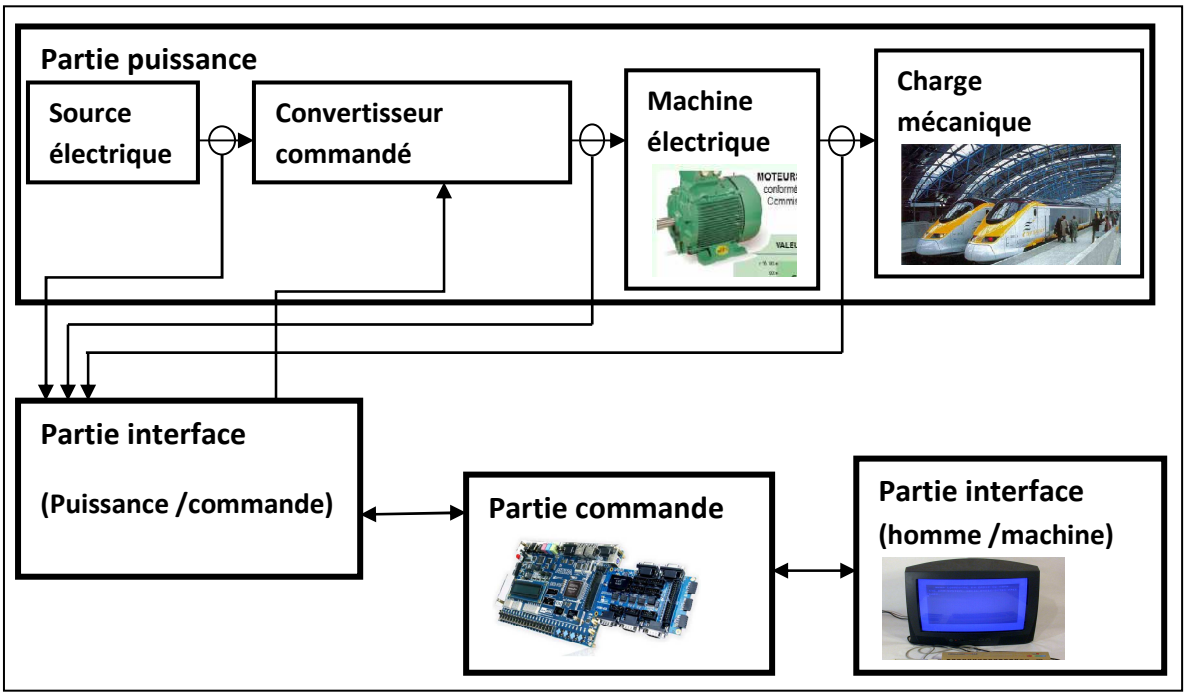


Figure III-3 : Structure générale du commande d'une machine électrique.

- **Partie Puissance**

Cette partie elle-même inclue quatre éléments. Le premier étant une source électrique. Cette dernière peut être fournie via des batteries, des générateurs, un réseau électrique (monophasé ou triphasé)… Elle peut aussi contenir des composants d'électrotechnique et d'électronique de puissance tels que les transformateurs, les Autotransformateurs, les ponts redresseurs (commandés ou non commandés), les filtres capacitifs… Le deuxième élément est un ou plusieurs convertisseurs commandés. Il s'agit de dispositifs d'électronique de puissance incluant des interrupteurs de puissance (IGBTs, Thyristors,…) qui sont commandés à travers des signaux de commande à l'ouverture et/ou à la fermeture. Le rôle d'un convertisseur commandé est de convertir l'énergie électrique fournie par la source électrique d'une forme à une autre à travers la commande d'interrupteurs de puissance. Par exemple, il est possible de trouver des convertisseurs continu-continu tels que les hacheurs, des convertisseurs continu-alternatif tels que les onduleurs, des convertisseurs alternatif-continu tels que les redresseurs (commandés ou non commandés), des convertisseurs alternatif-alternatif tels que les gradateurs… Le troisième élément est une machine électrique. Cette dernière constitue une charge électromécanique qui est alimentée via l'énergie électrique fournie à la sortie du convertisseur commandé. La machine électrique permet de convertir l'énergie électrique qu'elle reçoit en une énergie mécanique sous forme de couple. Plusieurs types de machines sont utilisés dans l'industrie selon l'application considérée et les performances souhaitées. Par exemple, on y trouve les moteurs pas à pas, les machines à courant continu, les machines asynchrones, les machines synchrones, les machines à réluctance variable… Le quatrième et dernier élément qui constitue la partie puissance est la charge mécanique. Cette dernière utilise l'énergie mécanique délivrée par la machine électrique pour remplir une fonctionnalité donnée.

- **Partie Interface (Puissance/Commande)**

  Cette partie assure le traitement électronique des signaux électriques échangés entre la partie puissance et la partie commande. Elle est constituée d'éléments tels que les capteurs électriques (capteurs de tension, de courant…), les capteurs mécaniques (couple, vitesse, position…), l'électronique de filtrage des perturbations, l'électronique de conversion analogique numérique et de conversion numérique

analogique, l'électronique de pilotage des interrupteurs de puissance du convertisseur commandé…etc.

- **Partie Commande**

  Cette partie assure le contrôle de l'état de la machine électrique (contrôle du courant, du couple, de la vitesse, de la position,…). Ce contrôle étant assuré par un algorithme de commande qui est implanté sur cible analogique ou numérique. Le contenu algorithmique de la partie commande dépend du cahier de charge de l'application considérée et des performances souhaitées. La partie commande acquiert dans un sens les signaux électriques générés par la partie interface et dans un autre, elle envoie les signaux de commande vers le convertisseur commandé.

- **Partie Interface (Homme/Machine)**

  Cette partie permet le contrôle de l'état du système à travers un échange bidirectionnel d'informations entre le manipulateur et le système commandé. Elle permet dans un sens d'envoyer les consignes de référence (consignes de courant, de couple, de vitesse, de fréquence d'échantillonnage,…) vers la partie commande et de récupérer dans un autre l'état d'évolution des grandeurs de la machine électrique. L'objectif de cette partie est de pouvoir acquérir des informations sur l'évolution des variables du système commandé et en même temps d'assurer un contrôle simple et transparent de l'état de la machine électrique [52].

III.4. Méthodes d'implantation d'un algorithme de commande

La notion d'implantation est définie comme étant l'introduction d'une fonctionnalité donnée sur un support physique. Dans le cadre de commande des machines, la fonctionnalité à introduire constitue l'algorithme de commande, dont l'objectif est de contrôler l'état d'évolution de variables mécaniques ou électriques de l'objet a commander (courant, flux, puissance, couple, vitesse…). Quant au support physique, il constitue la cible d'implantation. Cette dernière peut être de nature analogique ou numérique. Lorsqu'il s'agit d'une cible de nature analogique, le contrôle est conçu via des circuits analogiques. Lorsqu'il s'agit d'une cible de nature numérique, l'algorithme de commande est discrétisé et est réalisé via des solutions numériques.

Les premiers contrôles de machines électriques furent réalisés via des implantations de nature analogiques. Les principaux avantages des solutions analogiques sont les suivants :

- Réalisation de contrôles avec une large bande passante.
- Faible coût.
- Réalisation de contrôles avec une haute résolution.

Cependant les implantations de nature analogique présentent les inconvénients suivants :

- Variations des paramètres de contrôle liées aux contraintes thermiques inhérentes du circuit de contrôle.
- Sensibilité aux perturbations.
- Grand nombre de composants (Diminution de fiabilité).
- Nécessité d'entretien et de réajustement régulier.
- Faible flexibilité de la modification des structures de contrôles.

Avec l'avancement technologique, les implantations de nature numérique sont devenues les plus répandues. Les principaux avantages des solutions numériques sont les suivants :

- Grande flexibilité de modification des structures de contrôle.
- Immunité vis-à-vis des perturbations.
- Pas de problèmes de variations de paramètres de contrôle.

Ainsi, de nos jours, la plupart des implantations numériques d'algorithmes de commande de machines électriques sont basées sur des solutions logicielles telles que les microcontrôleurs, les microprocesseurs ou les DSP (Digital Signal Processor). Cependant, les avantages des solutions analogiques restent toujours difficiles à atteindre et ces solutions présentent les inconvénients suivants :

- La présence de délais de temps de calcul implique une dégradation de la bande passante du contrôle.
- Les périodes d'échantillonnage utilisées sont limitées par les délais de temps de calcul.
- Le fonctionnement discret et la quantification des algorithmes font que la précision et les performances de contrôle sont dégradées.
- La compensation des délais dus au temps de calcul augmente la complexité des algorithmes à implanter.

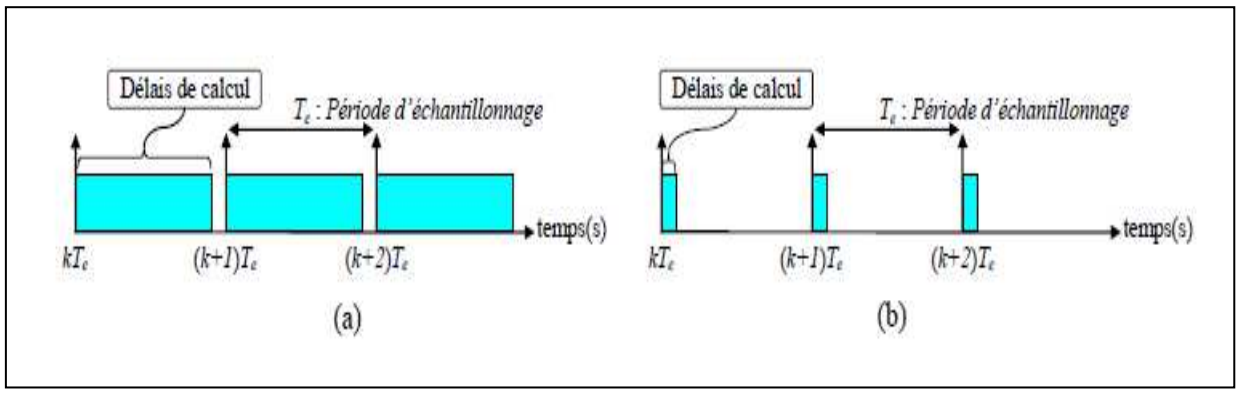


Figure III-4 : Différence de capacité de calcul entre les solutions (a) logicielles_(b) matérielles [52].

Au-delà des solutions logicielles traditionnelles, les nouvelles solutions matérielles telles que les FPGA peuvent aussi être considérées comme étant des solutions numériques appropriées pour l'implantation des algorithmes de commande. Par ailleurs, le parallélisme inhérent des composants FPGA ainsi que leurs grandes capacités de calcul permettent de réaliser des techniques de contrôle avec des délais de temps d'exécution très petits en dépit de la performances analogiques en augmentant la bande passante des contrôles et en affinant leur résolution temporelle [52].

III.5. Contribution des FPGA dans la commande

Les structures de contrôle comportent plusieurs boucles de régulation imbriquées (boucle de régulation de courant, de vitesse, de position,…). La boucle qui consomme le plus de temps de calcul par exemple est la boucle interne de régulation de courant, et seulement des ressources de calcul limitées sont dédiées aux boucles externes. Pour ce faire, il est avantageux d'utiliser des solutions numériques ayant de grandes capacités de calcul pour l'implantation des algorithmes de commande. Les solutions numériques les plus utilisées pour la commande des machines électriques sont les microprocesseurs et les DSP. Cependant, les exigences de contrôle modernes dans le domaine de commande de machines électriques dépassent les capacités de calcul offertes par ces solutions. Et bien que les nouveaux multiprocesseurs et les nouveaux DSP de hautes performances peuvent résoudre ce problème, ils présentent l'inconvénient d'avoir un coût élevé.

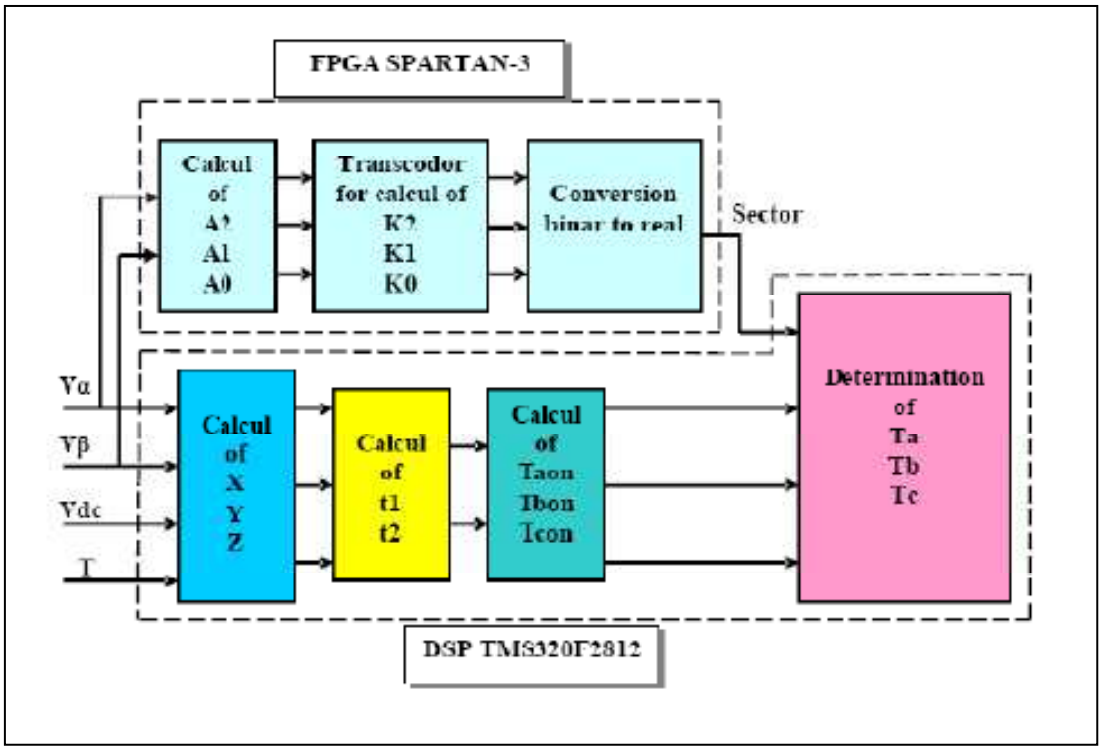


Figure III-5 : Mode de fonctionnement parallèle entre un FPGA et un DSP [57].

La migration du mode de fonctionnement séquentiel des solutions logicielles au mode de fonctionnement parallèle des solutions matérielles est un nouveau degré de liberté offert aux concepteurs qui s'est avéré bénéfique dans le domaine de commande de machines électriques et qui a permis de répondre aux exigences de contrôle modernes.

Parmi les nouvelles solutions matérielles, les composants FPGA ont été utilisés avec succès dans différentes applications liées à la commande de machines électriques. En effet, ils ont été utilisés pour le contrôle des convertisseurs de puissance tels que les onduleurs de tension triphasé, les convertisseurs alternatif/continu, les convertisseurs multi niveaux, les filtres actifs,… Les FPGA ont aussi été utilisés pour le contrôle des machines asynchrones, des machines synchrones, des machins à réluctance variable .Ainsi, grâce aux caractéristiques propres des FPGA, il est possible de :

- **Améliorer les performances de contrôle :** La rapidité de calcul des FPGA permet une augmentation de la bande passante des boucles de régulation et une meilleure résolution temporelle.

- **Implanter des algorithmes complexes :** Avec l'avancement technologique, l'augmentation d'intégration des composants FPGA ne cesse d'augmenter. De nos jours, la densité des composants FPGA peut atteindre l'équivalent de 10 millions de portes logiques avec des fréquences de commutation de l'ordre de 500 MHz. Ceci permet l'implantation d'algorithmes de contrôle complexes dans leur intégralité avec un faible délai de temps de calcul.
- **Réaliser des reconfigurations dynamiques** : Le parallélisme inhérent des composants FPGA offre la possibilité de faire tourner plusieurs algorithmes de commande en parallèle et de reconfigurer entre eux selon des critères bien définis. La reconfiguration dynamique entre les algorithmes de commande permet de sélectionner les algorithmes appropriés selon les points de fonctionnements. Elle peut être utile aussi pour assurer une continuité de fonctionnement en cas de défauts (capteurs, interrupteurs, …).
- **Renforcer la confidentialité** : L'architecture de contrôle implanté sur cible FPGA n'est pas facilement duplicable [9].

III.6. Méthodologie de développement pour implantation sur cible FPGA

Plusieurs méthodologies de développement pour la conception d'architectures matérielles. Elles ont toutes été conçues en ayant comme objectif le développement d'architectures génériques et réutilisables afin de pouvoir les réutiliser dans différentes applications. La notion de réutilisabilité est toujours de première importance étant donné qu'elle permet de créer une bibliothèque de modules réutilisables appelés aussi fonctions IP (Intellectuel Property). La spécificité de la méthodologie de développement utilisée dans ce travail est qu'elle est facile à appréhender par l'ingénieur électrotechnicien sans qu'il soit expert en microélectronique. Les étapes de développement de l'architecture à implanter sont principalement effectuées via le logiciel Matlab-Simulink ainsi que les outils CAO des solutions matérielles. Les différentes étapes de la méthodologie de développement considérée sont détaillées dans les paragraphes qui suivent [9,8].

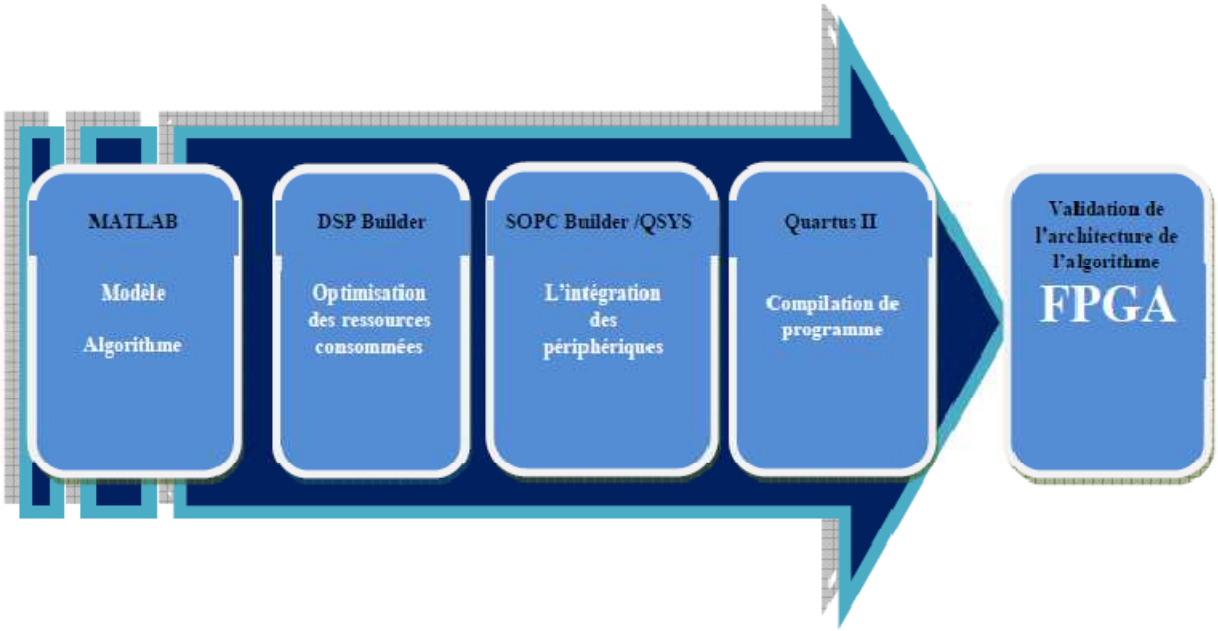


Figure III-6 : Les étapes optimisées de l'implantation d'un algorithme de commande sur FPGA.

III.6.1. Partitionnement modulaire de l'algorithme de commande

Cette étape est spécialement importante lorsque les algorithmes à implanter sont de nature complexe. En effet, l'objectif de cette étape est de décomposer l'algorithme de commande à implanter en plusieurs "sous-algorithmes" appelés modules ayant des fonctions bien définies.

Ceci permet, d'une part de faciliter les conceptions à réaliser, et d'autre part de minimiser le temps de développement. Cependant, le partitionnement modulaire d'un algorithme de commande nécessite une certaine réflexion de la part du concepteur. Ce dernier doit partitionner l'algorithme de manière fonctionnelle, et ce en identifiant des modules qui soient indépendants et réutilisables tels que les régulateurs, les fonctions de modulation, les estimateurs, les opérateurs vectoriels… Ainsi, le concepteur doit extraire un maximum de modules réutilisables en vue de rendre possible leur réutilisation comme des éléments d'une bibliothèque spécifique. En même temps, l'extraction des modules doit être effectuée de manière hiérarchique afin d'être adaptée à la complexité de la conception.

A la fin de cette étape, plusieurs modules réutilisables, avec différents niveaux hiérarchiques sont extraits afin d'être ajoutés (à la fin de leur conception) à une bibliothèque spécifique dédiée à la commande de machines électriques comme le montre la figure III.7. Trois niveaux hiérarchiques sont suffisants pour caractériser dans son intégralité la bibliothèque de modules dédiés à la commande de machines électriques. Le premier niveau hiérarchique inclut les opérateurs de "grain fin" tels que les registres, les multiplexeurs, les additionneurs, les multiplieurs… Ensuite, le deuxième niveau hiérarchique comporte les modules des fonctions les plus communément rencontrées dans la commande des machines électriques tels que les régulateurs PI, les techniques de modulation de largeur d'impulsion (MLI), les transformations de coordonnées… Les modules du deuxième niveau hiérarchique sont construits en utilisant les modules du premier niveau hiérarchique. Finalement, les algorithmes de contrôle constituent les modules du troisième niveau hiérarchique de la bibliothèque. Ces modules d'algorithmes de contrôle sont construits en utilisant des modules du premier et deuxième niveau hiérarchique de la bibliothèque [9].

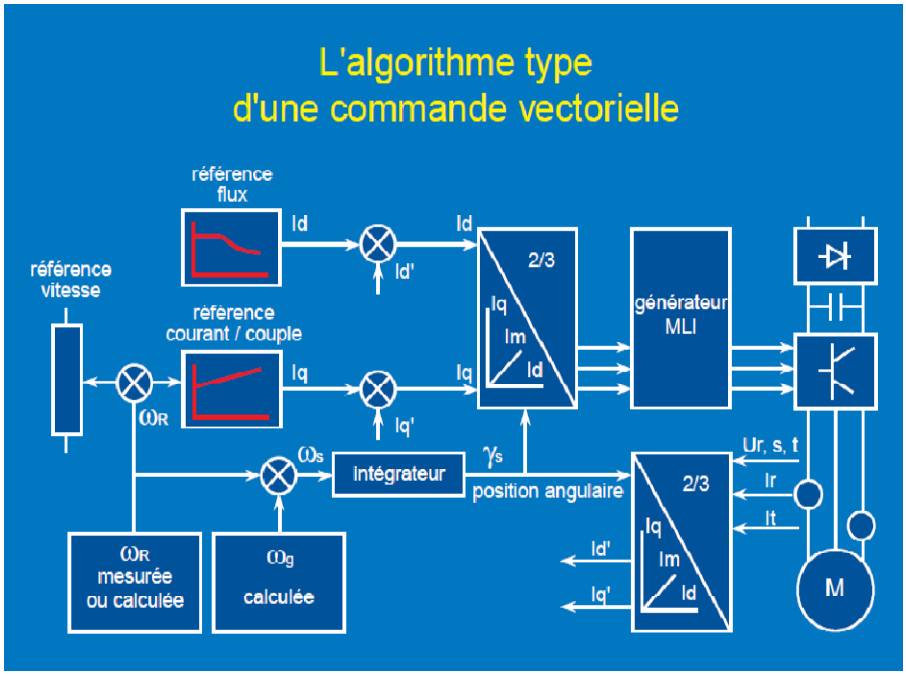


Figure III-7 : Partitionnement modulaire d'un algorithme type commande vectoriel [50].

III.6.2. Etape de simulation

La procédure de simulation est effectuée en utilisant le logiciel Matlab-Simulink. L'objectif de cette étape est de :

- Vérifier la fonctionnalité de l'algorithme de contrôle lorsqu'il est inséré dans l'application considérée.
- Déterminer une période d'échantillonnage et un format à virgule fixe qui permettent d'atteindre les performances de contrôle souhaitées.

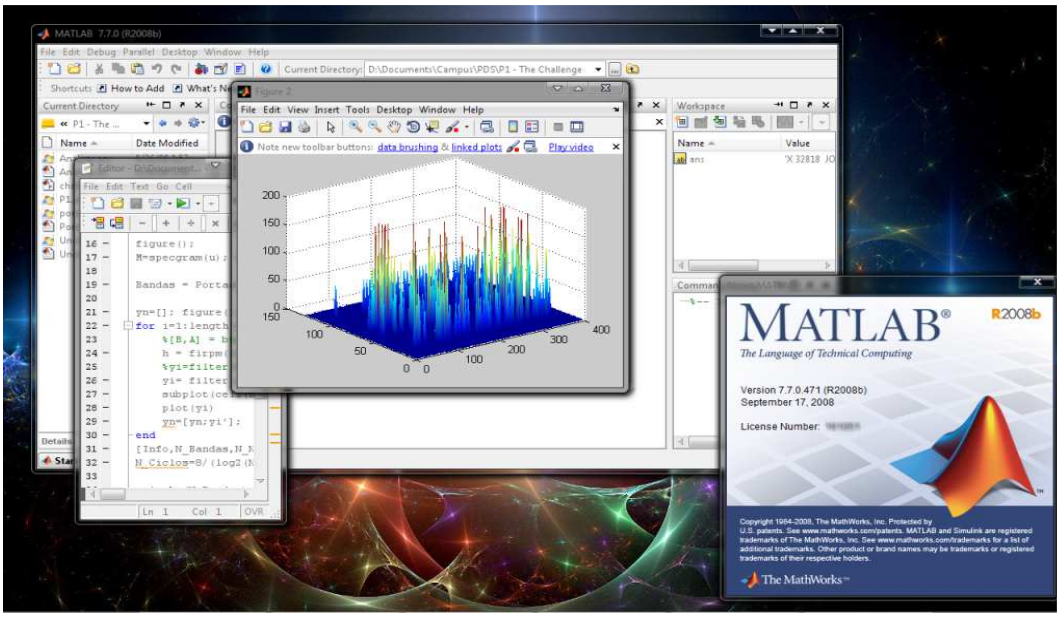


Figure III-8 : Fenêtres de logiciel Matlab-Simulink [10].

La vérification de la fonctionnalité de l'algorithme est effectuée à travers le développement d'un modèle fonctionnel en utilisant les blocs en temps continu de Matlab-Simulink. L'algorithme de commande est par la suite discrétisé et normalisé. La quantification de l'algorithme de commande discrétisé et normalisé est alors effectuée en étudiant l'influence de la période d'échantillonnage et celle du format à virgule fixe sur les performances de contrôle. Plusieurs méthodes analytiques sont citées en bibliographie portant sur la détermination du format et de la période d'échantillonnage. Dans le cadre de ce travail, et pour des raisons de simplicité, l'opération de quantification est effectuée uniquement par simulation à travers le développement d'un modèle de spécification à virgule fixe de l'algorithme de commande discrétisé et normalisé. Ce modèle est développé en utilisant la

toolbox "fixed point" de Matlab-Simulink. Ce choix est du au fait que la densité des composants FPGA ne cesse d'augmenter avec l'avancement technologique et que les procédures d'optimisation de ressources qui seront présentées par la suite sont suffisantes pour satisfaire les contraintes matérielles de la cible FPGA. En effet, de nos jours même des algorithmes de contrôle complexes peuvent être implantés intégralement sur des cibles FPGA de faible coût, même si le format à virgule fixe choisi est élevé, voire surdimensionné pour l'application visée.

Une fois le développement du modèle de spécification discret et à virgule fixe achevé, un GFD (Graphe de Flot de Données) est défini pour chaque sous algorithme des différents modules de deuxième niveau hiérarchique extraits lors de l'étape de partitionnement modulaire. Un algorithme donné peut être décrit de différentes manières. Le GFD constitue une description graphique de l'algorithme, où ce dernier est décomposé en plusieurs opérations élémentaires implantables telles que l'addition, la soustraction, la multiplication, le retard, la comparaison, les fonctions trigonométriques… Le GFD mentionne aussi le format à virgule fixe utilisé [9].

- **Exemple d'un modèle Simulink de la machine asynchrone**

  La figure III-9 représente le modèle SIMULINK du moteur asynchrone décrit par des équations. Il est constitué de sept blocs principaux, qui sont [53]:

- La source d'alimentation triphasée,
- Le bloc de transformation abc/dq,
- Le bloc calculant les courants statoriques et rotoriques dans le plan dq,
- Le bloc calculant le couple électromagnétique du moteur,
- Le bloc calculant le couple résistant de la charge,
- Le bloc calculant la vitesse du moteur,
- Et le bloc calculant les trois signaux triphasés du courant statorique.

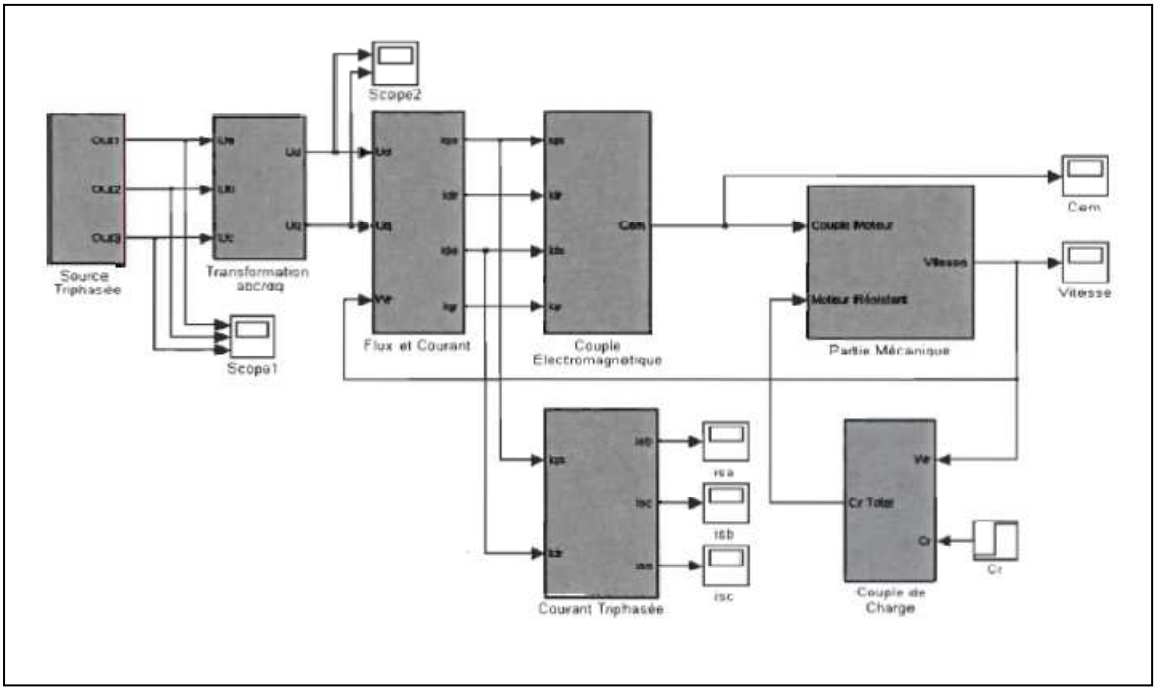


Figure III-9 : Modèle Simulink du moteur asynchrone [53].

III.6.3. Optimisation des ressources consommées

III.6.3. a. Programmé FPGA Utilisation de HDL Coder

Si vous utilisez MATLAB pour modéliser le traitement numérique du signal (DSP) ou vidéo et des algorithmes de traitement d'images …etc, qui finissent par se retrouver dans les FPGA ou ASIC, Utilisation de HDL Coder, vous pouvez programmer les FPGA, y compris les dispositifs d'Altera ®, Xilinx ®, et d'autres fournisseurs de FPGA. Cette fonctionnalité vous permet de prototyper rapidement votre conception sur le matériel FPGA. Le conseiller Workflow dans HDL Coder s'intègre avec Xilinx ISE ® et Altera Quartus ® II (des logiciel de conception de l'architecture à implanter) suites de conception de programmer automatiquement vos FPGA à partir de MATLAB et Simulink [10].

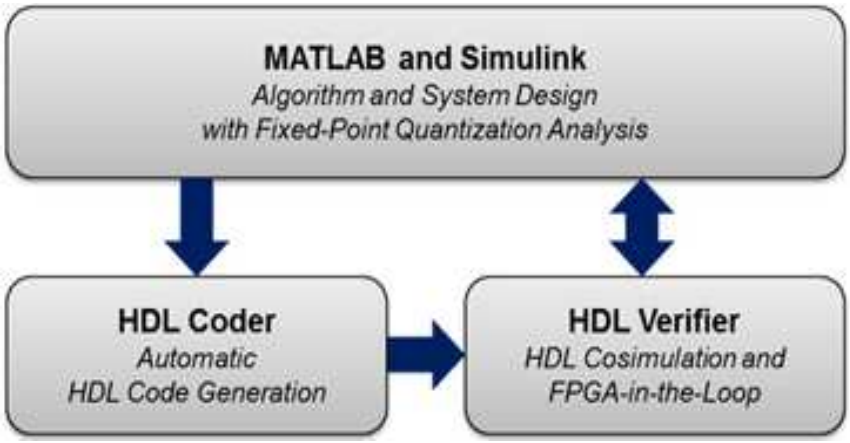


Figure III-10 : Générer du code HDL à partir de MATLAB et Simulink [10].

- **HDL workflow conseiller**

Le workflow conseiller HDL (voir la figure ci-dessous) permet d'automatiser les étapes et fournit un parcours guidé à partir de MATLAB pour le matériel. Vous pouvez voir les étapes clés suivantes du flux de travail dans le volet gauche du conseiller de flux de travail:

1. Conversion en virgule fixe
2. Génération de code HDL
3. Vérification HDL
4. Synthèse et analyse HDL

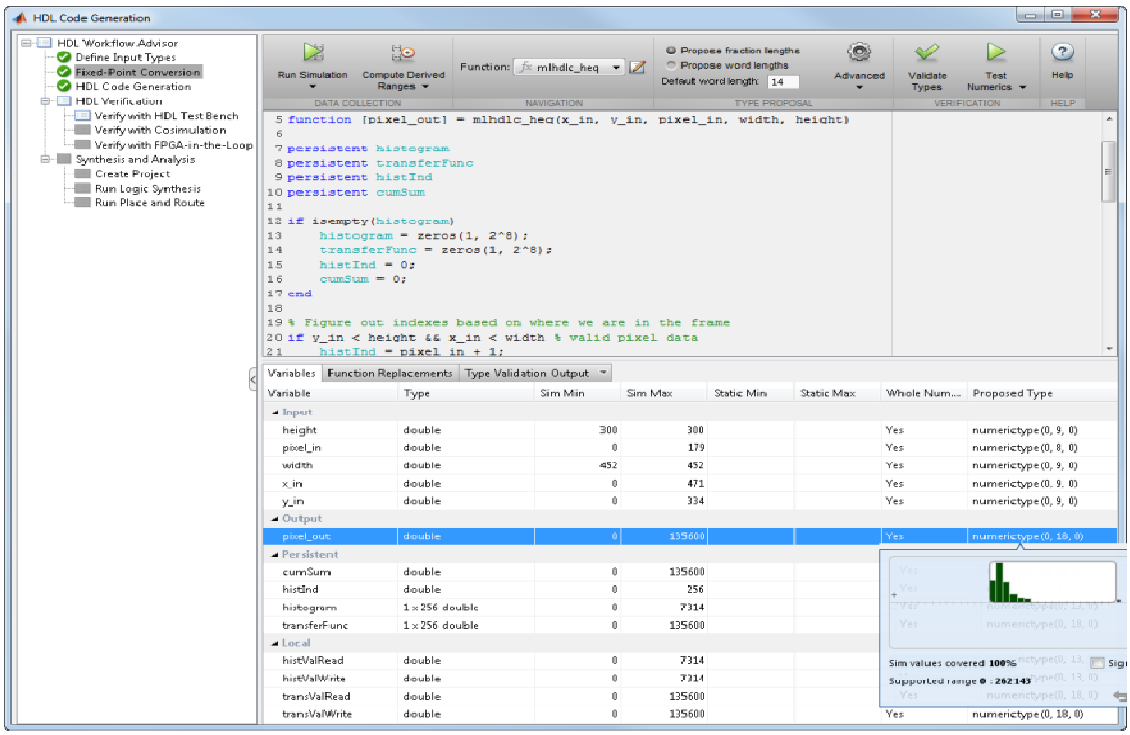


Figure III-11 : Fenêtre de Workflow conseiller [10].

III.6.3. a.1. Conversion en virgule fixe

Applications de traitement du signal sont typiquement implémentées en utilisant les opérations en virgule flottante dans MATLAB. Toutefois, pour pouvoir, le coût et le rendement des raisons, ces algorithmes doivent être convertis pour utiliser les opérations en virgule fixe pour cibler matériel. Conversion en virgule fixe peut être très difficile et de longue haleine, exigeant généralement de 25 à 50 pour cent de la conception globale et le temps de mise en œuvre. La virgule flottante automatique à point fixe conversion flux de travail dans HDL Coder ™ peut grandement simplifier et accélérer le processus de conversion. A noté que cette étape est facultative. Vous pouvez sauter cette étape si votre conception MATLAB est déjà mise en œuvre en virgule fixe.

III.6.3. a.2. Génération de code HDL

L'étape de génération de code HDL génère du code HDL à partir du code MATLAB en virgule fixe. Vous pouvez générer soit VHDL ou Verilog code qui implémente votre conception MATLAB. En plus de générer du code HDL synthétisable, HDL

Coder ™ génère également différents rapports, dont un rapport de traçabilité qui vous permet de naviguer entre votre code MATLAB et le code HDL généré, et un rapport d'utilisation des ressources qui vous montre, au niveau de l'algorithme, à peu près ce ressources matérielles sont nécessaires pour mettre en œuvre la conception, en termes de vipères, des multiplicateurs et RAM.

Lors de la génération de code, vous pouvez spécifier plusieurs options d'optimisation pour explorer l'espace de conception sans avoir à modifier votre algorithme. Dans la section Exploration de l'espace de conception et d'optimisation des options ci-dessous, vous pouvez voir comment vous pouvez modifier les options de génération de code et optimiser votre conception de la vitesse ou de la zone.

III.6.3. a.3. Vérification HDL

Standalone test de HDL génération de banc:

HDL Coder ™ génère VHDL et Verilog bancs d'essai de vos scripts MATLAB pour la vérification rapide de code HDL généré. Vous pouvez adapter un banc d'essai de HDL en utilisant une variété d'options qui s'appliquent à des stimuli du code HDL. Vous pouvez également générer des fichiers de script pour automatiser le processus de compilation et de simulation de votre code dans les simulateurs de HDL. Ces mesures aident à s'assurer que les résultats de la simulation MATLAB correspondent aux résultats de la simulation HDL.

III.6.3. a.4. HDL Synthèse

Outre les défis liés à la langue, la programmation de FPGA nécessite l'utilisation d'outils de CAO complexes. Génération d'un flux binaire de la conception de HDL et la programmation du FPGA peut être lourdes tâches. HDL Coder ™ offre automatisation ici, en créant des fichiers de projet pour Xilinx ® et Altera ® qui sont configurés avec le code HDL généré. Vous pouvez utiliser les étapes du flux de travail pour synthétiser le code HDL dans l'environnement MATLAB, voir les résultats de la synthèse, et itérer sur la conception MATLAB pour améliorer les résultats de synthèse [10].

III.6.3. b.Programmé FPGA Altera Utilisation Altera DSP Builder

Altera et Mathworks travaillent en étroite collaboration pour donner une nouvelle technologie qui appeler le DSP Builder qui nous permet de passer de la définition et de la simulation système en utilisant les outils MathWorks Simulink standard de l'industrie pour la mise en œuvre du système en quelques minutes. Le DSP Builder est un signal numérique traitant l'outil de développement (DSP) qui connecte entre le logiciel de Quartus II et les outils de MathWorks MATLAB/Simulink. La figure III-12 montre l'écoulement de la mise en œuvre de la conception à l'aide DSP Builder.

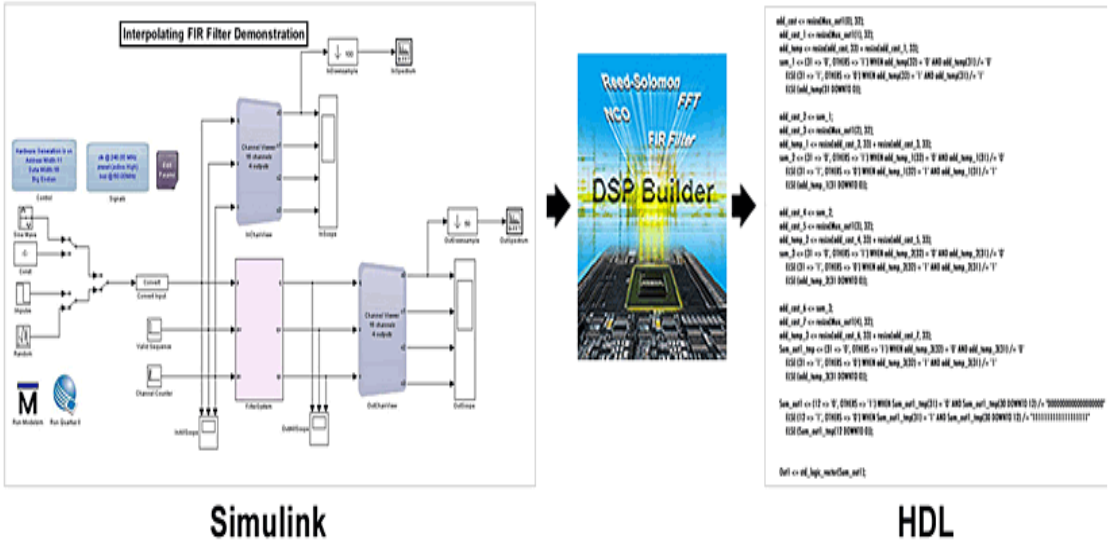


Figure III-12 : DSP Builder Conception débit [8].

DSP Builder vous permet de créer des modèles optimisés pour les FPGA Altera dans Simulink sans exiger que vous travailliez directement avec le code HDL. Avec DSP Builder avancée Blockset, vous spécifiez les contraintes de conception de haut niveau, tels que la fréquence d'horloge souhaitée et le nombre de canaux de votre modèle Simulink. DSP Builder génère alors automatiquement RTL pipeline ciblé et optimisé pour votre appareil FPGA choisi. Parce avancée Blockset utilise un multiplexage temporel pour optimiser l'utilisation de la logique et insère automatiquement étages de pipeline et enregistre pour répondre aux contraintes de conception, vous pouvez obtenir des performances similaires sur FPGA en main optimisée du code HDL.

DSP Builder intègre une haute performance, à faible latence écoulement de l'outil à virgule flottante en utilisant la technologie de chemin de données fusionnée. Cette capacité permet à l'ingénieur de conception pour construire les chemins des données de traitement de signaux qui combinent opérations en virgule flottante et en virgule fixe [8].

III.6.4. L'intégration des périphériques

Deux étapes sont nécessaires à la programmation du NIOS. La première partie concerne la conception du système avec le processeur embarqué NIOS II :

- Construction de la plateforme matérielle
- Choix des périphériques matériels et des interconnexions.
- Intégration de ces périphériques

Cela se fait grâce au logiciel Quartus. Cet ensemble d'outils est développé par Altera. Il fourni plusieurs environnement de conception dont l'outil de conception matérielle : SOPC Builder.

La deuxième étape concerne le jeu d'instruction programmé au niveau logiciel du NIOS en langage C. Altera a prévu un environnement de développement appelé Eclipse qui permet cette programmation, ainsi que la compilation et la simulation du programme .

Quartus II intègre l'outil SoPC Builder qui permet de construire un système SoPC (System on Programmable Chip). Il permet graphiquement de construire par exemple un microcontrôleur intégrant des périphériques d'E/S divers et variés, On peut ainsi intégrer autant de périphériques que l'on veut, n'étant limité que par le nombre de broches et de cellules logiques du circuit FPGA ciblé.

C'est l'outil SOPC-Builder qui permet de configurer et de générer la description matérielle (en VHDL) du processeur et de ses périphériques. Il génère également le code (en assembleur et en langage C) d'un système d'exploitation minimal (chargeur, entrées-sorties par liaison série) qui sera compilé puis exécuté sur le processeur Nios II, vous pourrez visualiser les différents composants du système ainsi que leurs configuration (adresses de base, numéro d'interruption, etc.) comme indiqué dans figure III-13.

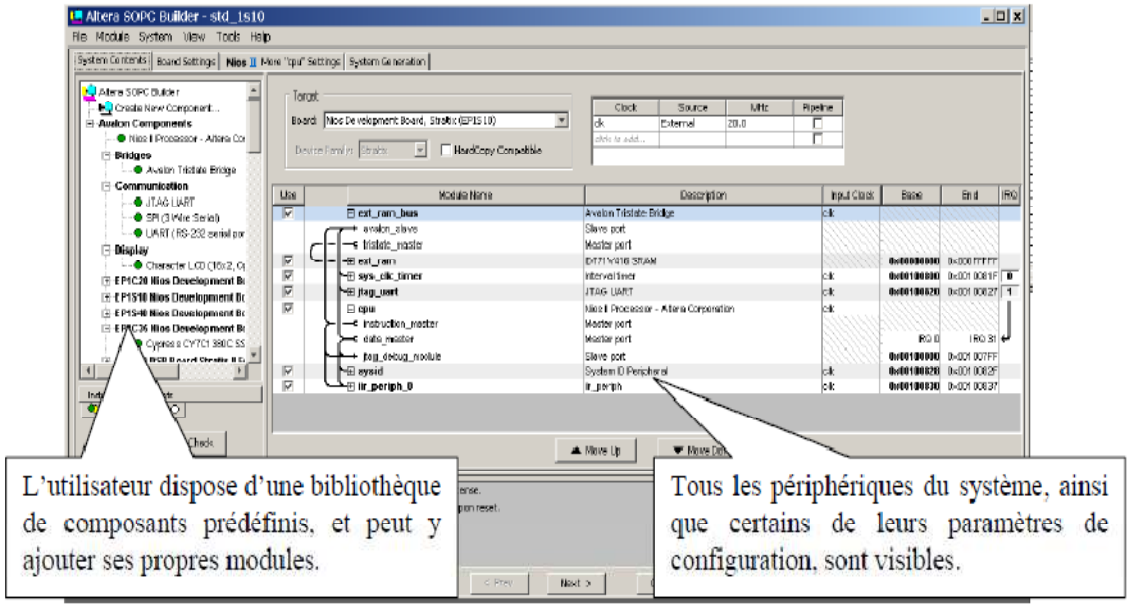


Figure III-13 : La fenêtre principale de SOPC-Builder [11].

La deuxième étape de L'environnement que nous allons utiliser est basé sur *Eclipse*, il permet de gérer l'édition, la compilation l'exécution et le débogage d'une application NIOS-II écrite en langage C. Une vue de la fenêtre principale est donnée figure III-14 [11].

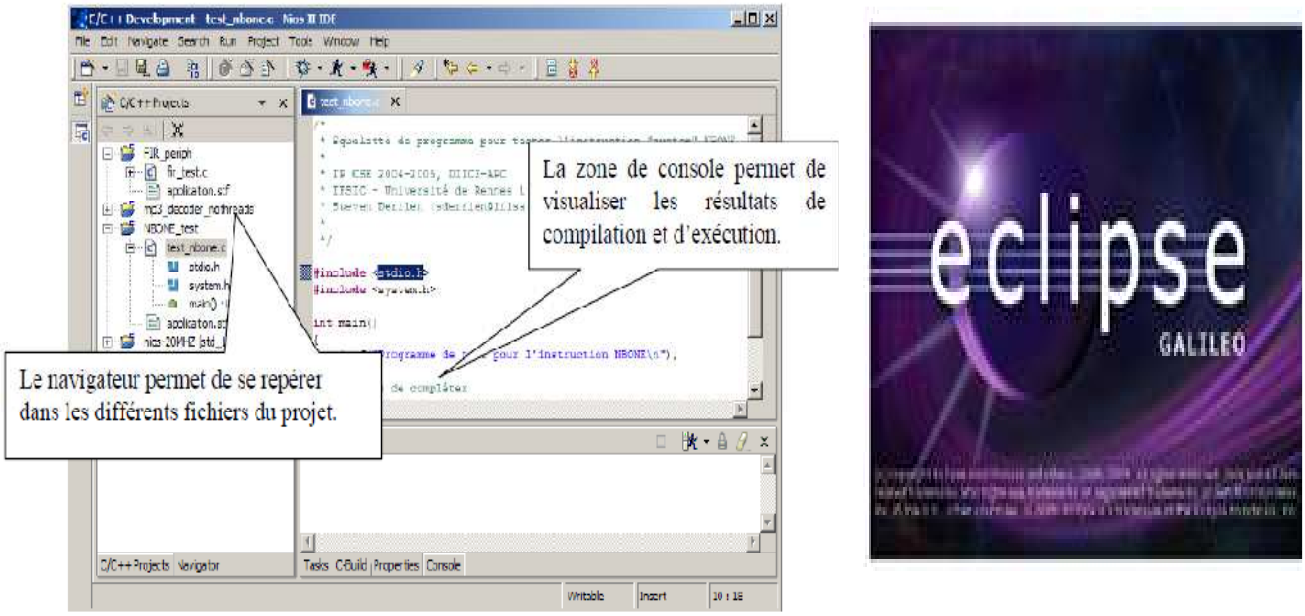


Figure III-14 : Environnement Eclipse pour NIOS-II [11].

III.6.5. Conception modulaire de l'architecture de commande

La conception modulaire de l'architecture d'algorithme et assurer par le logiciel Quartus II et le MAX+PLUS II (pour Altera) ou le ISE (pour Xilinx) qui comprend une suite de fonctions de conception au niveau système, permettant d'accéder à la large bibliothèque d'IP et un moteur de placement-routage intégrant la technologie d'optimisation de la synthèse physique et des solutions de vérification.

Figure III-15 : Icône du Navigateur de projet ISE et Quartus disponible sur le bureau [11,38].

Flots de conception des fabricants de FPGA

Editeur	Actel	Altera	Lattice	Xilinx
Nom du flot	Libero 5.0	Quartus II 3.0	IspLever 3.1	ISE 6.1i
Points marquants des nouvelles versions	Vue graphique de la conception au niveau physique et logique; éditeur de paramétrage des E/S; outil de calcul de la consommation du circuit (SmartPower)	Compilation incrémentale; conception et implantation indépendantes de modules (fonction logic clock); outil graphique de connexion d'IP et de cœurs de processeurs (SOPC Builder)	Outil de gestion de projet; simulateur propre avec édition et visualisation des chronogrammes; outil d'analyse et d'estimation de temps critiques sans recompilation	Préplacement de blocs intégrant des contraintes de timing; assignation automatique des broches du FPGA pour le circuit imprimé; outil de débogage temps réel avec déclenchement croisé avec le logiciel
Principaux outils tiers utilisés dans le flot	**Simulation** ModelSim (Mentor) **Génération de testbench** Wave Former Lite 9.0 (SympaticCAD) **Synthèse** Leonardo, Precision (Mentor), Synplify 7.3 (Edition Actel de Synplify de Synplicity) **Synthèse physique** Palace (Magma)	**Simulation** ModelSim (Mentor), NC Sim (Cadence), Sirocco (Synopsys) **Synthèse** Design Compiler (Synopsys), Leonardo, Precision (Mentor), Synplify FPGA (Synplicity) **Analyse statique de délais** Prime Time (Synopsys)	**Simulation** ModelSim (Mentor) **Synthèse** Synplify FPGA (Synplicity)	**Simulation** NC Sim, Verilog XL (Cadence), ModelSim (Mentor), VCS, Sirocco (Synopsys) **Synthèse** Synplify (Synplicity), Leonardo, Precision (Mentor), FPGA Compiler, Design Compiler (Synopsys) **Synthèse physique** Palace (Magma)

Tableau III-1 : Flots de conception des fabricants de FPGA [20].

De manière générale, un flot de conception ayant pour but la configuration de composants programmables se déroulent de la manière suivante:

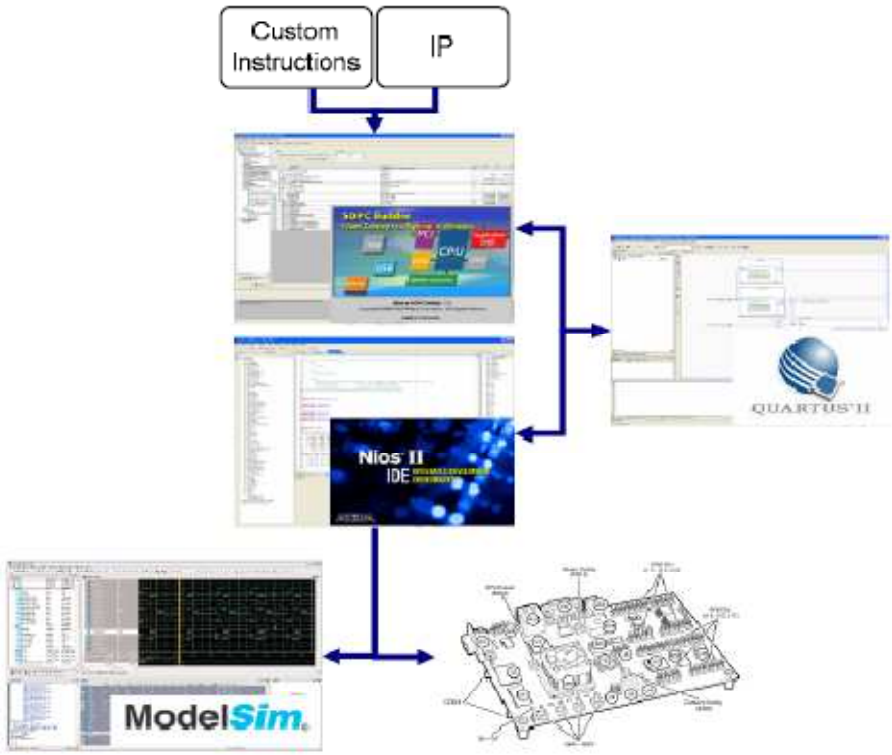


Figure III-16 : Déroulement de la configuration de composants FPGA [55].

- **Quartus II**

  Quartus est un logiciel développé par la société Altera, permettant la gestion complète d'un flot de conception CPLD ou FPGA. Ce logiciel permet de faire une saisie graphique ou une description HDL (VHDL ou verilog) d'architecture numérique, d'en réaliser une simulation, une synthèse et une implémentation sur cible reprogrammable (la même fonction pour ISE de Xilinx) [12].

  Altera propose deux versions de Quartus II :

1. Quartus II (version complète).
2. Quartus II (Web Edition) c'est une version gratuit qui ne contiens pas tout les fonctions nécessaire pour le développement du la puce FPGA.

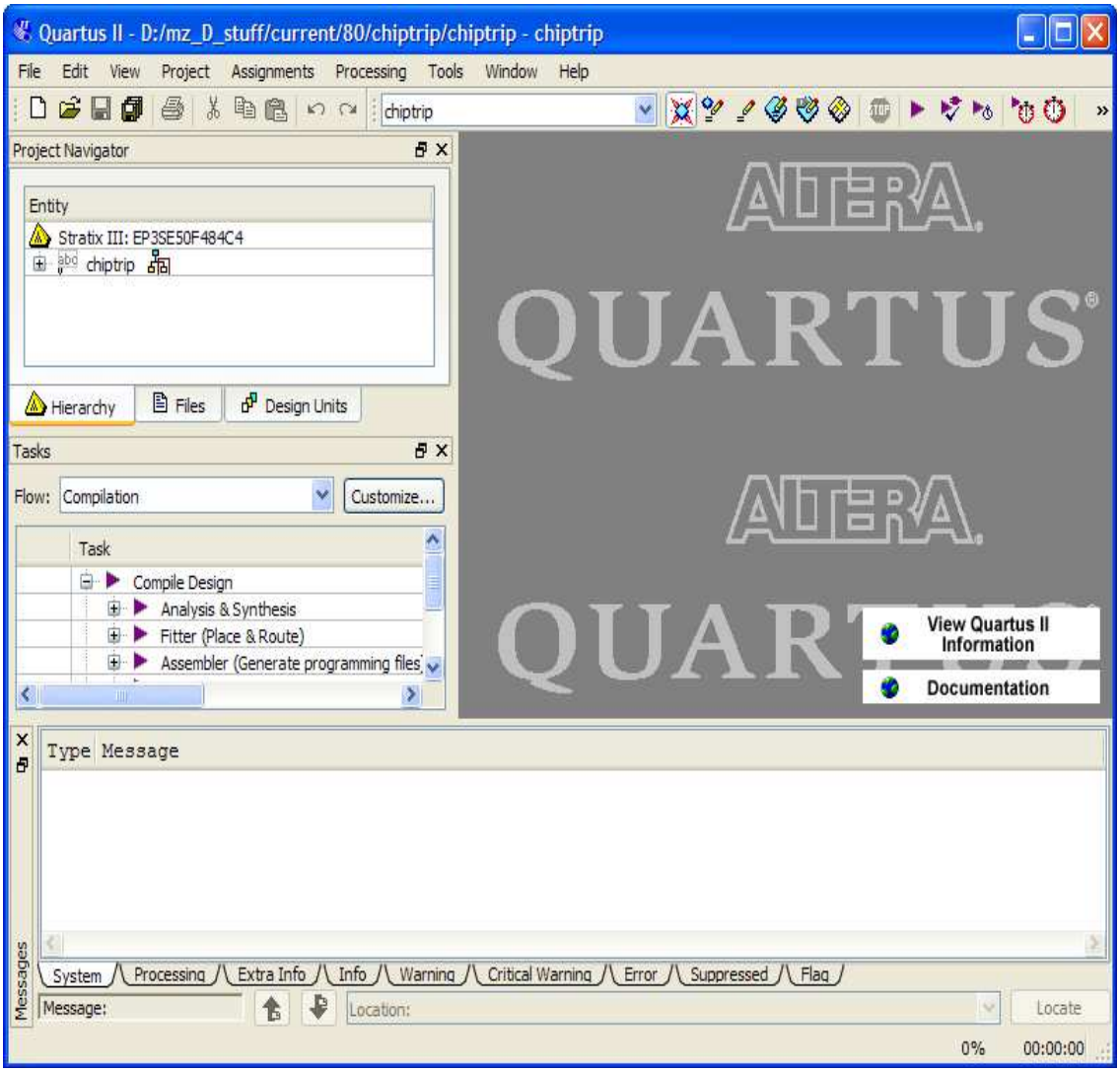


Figure III-17 : Interface graphique Quartus II [55].

Quartus II est un logiciel de conception fourni par le constructeur de FPGA, Altera. C'est un environnement de programmation qui gère complètement le processus de conception d'un circuit programmable. La Figure 19 présente les étapes de développement qui seront décrites au fur et à mesure de l'implémentation des algorithmes [4].

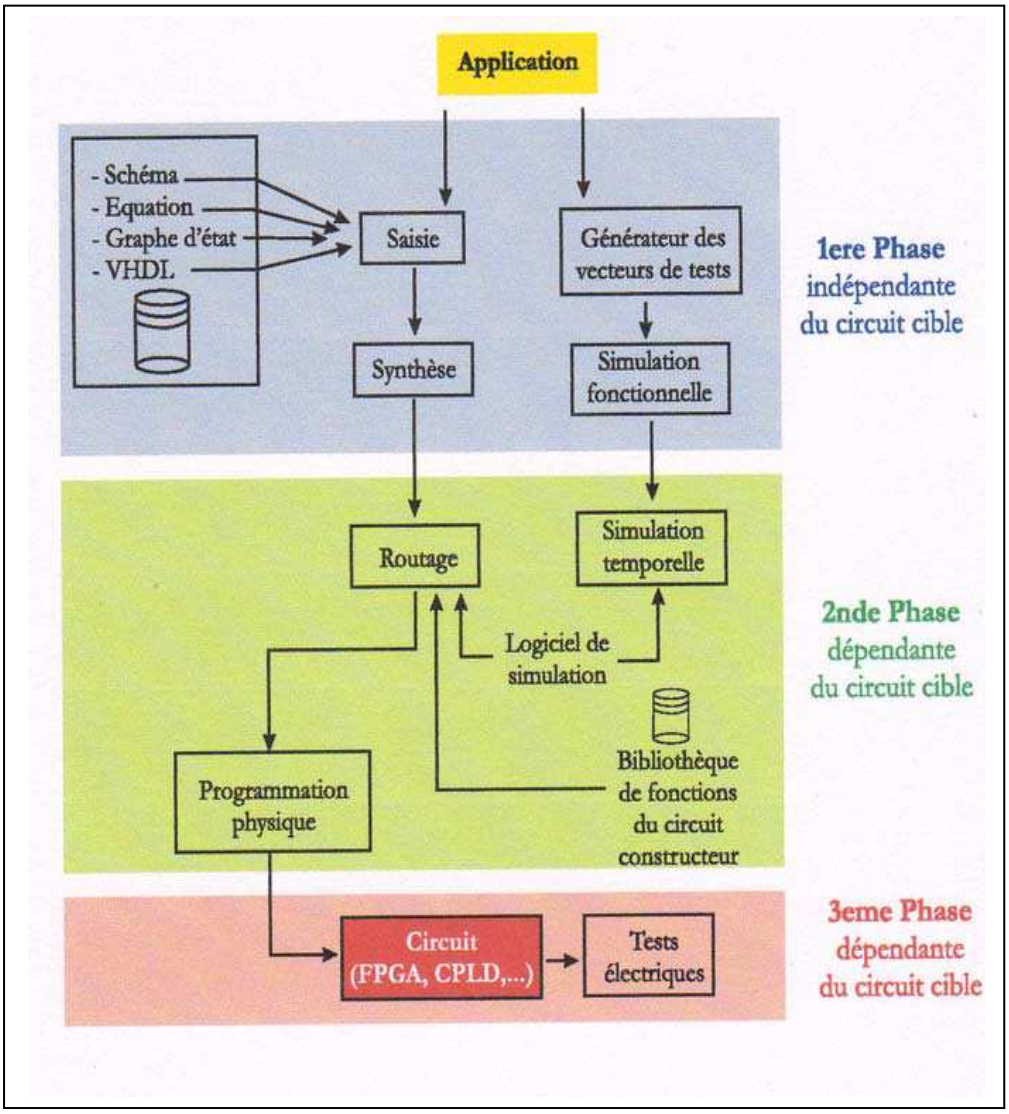


Figure III-18: Processus de conception d'un circuit FPGA [12].

Le Quartus est un logiciel qui travaille sous forme de projets c'est à dire qu'il gère un design sous forme d'entités hiérarchiques. Un projet est l'ensemble des fichiers d'un design que ce soit des saisies graphiques, des fichiers VHDL ou bien encore des configurations de composants (affectation de pins par exemple) [12].

La programmation d'un circuit se décompose en 4 phases :

1. Saisie de la description du circuit. Pour cela 3 outils sont utilisables :

- description sous la forme d'un schéma électronique,
- description textuelle en utilisant un langage de programmation (VHDL ou AHDL),
- description sous la forme de chronogramme.

2. Vérification de la description par compilation du circuit.

3. Vérification du bon fonctionnement de la description par simulation logico-temporelle. A ce stade, il peut-être nécessaire de reprendre à la 1ère phase [12].

4. Programmation du circuit physique.

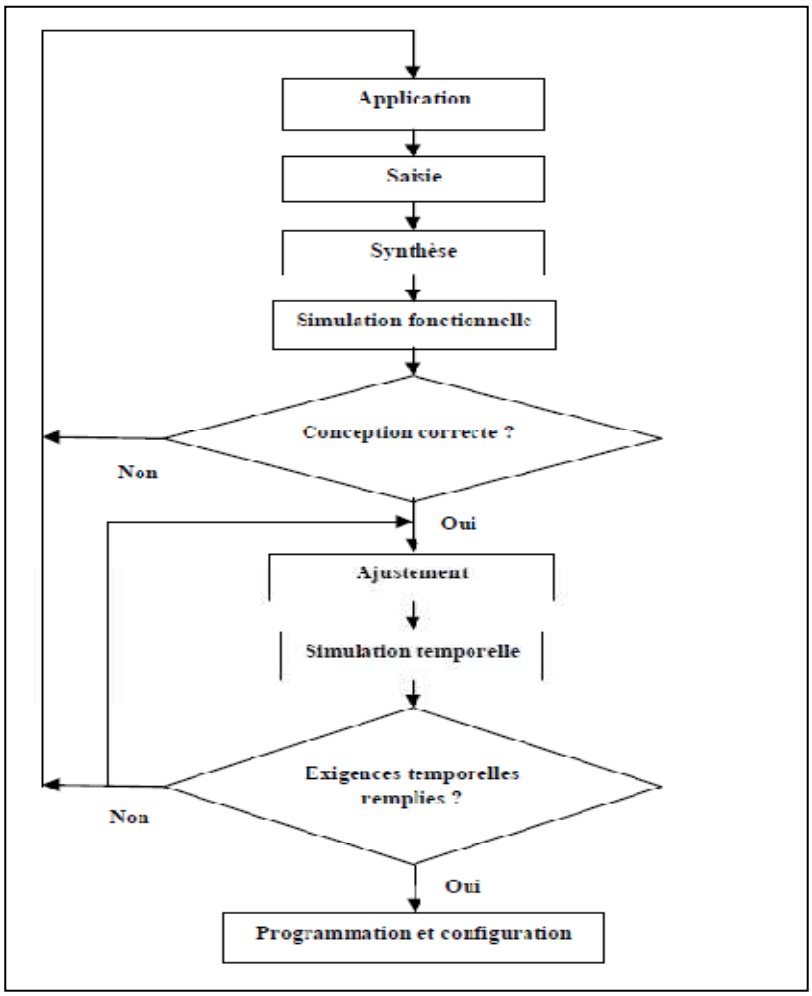


Figure III-19 : Organigramme de la conception.

Pour le logiciel Quartus II, un projet consiste en un ensemble de fichiers de conception, de fichiers d'assignation, de fichiers de simulation, d'options de configuration et d'informations sur le projet. Le module ***Création d'un Projet*** du didacticiel vous guidera à travers les étapes qui sont nécessaires à la création de votre projet.

- **Conception du projet**
- **Définition du projet**

  Quartus II gère des projets, c'est à dire des entités hiérarchiques, pour lesquels il effectue les opérations de CAO; pour illustrer cela, prenons notre module (nommé LOUVIERS), on va d'abord définir une feuille le décrivant et, pour cela, on va faire appel à des feuilles (de niveau hiérarchique inférieur) décrivant les sous-ensembles (COMPTEUR4, GENE_3 et DECODEUR).

  La hiérarchie de notre projet est accessible par dans la fenêtre en haut à gauche (il y a 3 onglets sur cette fenêtre), le résultat est le suivant :

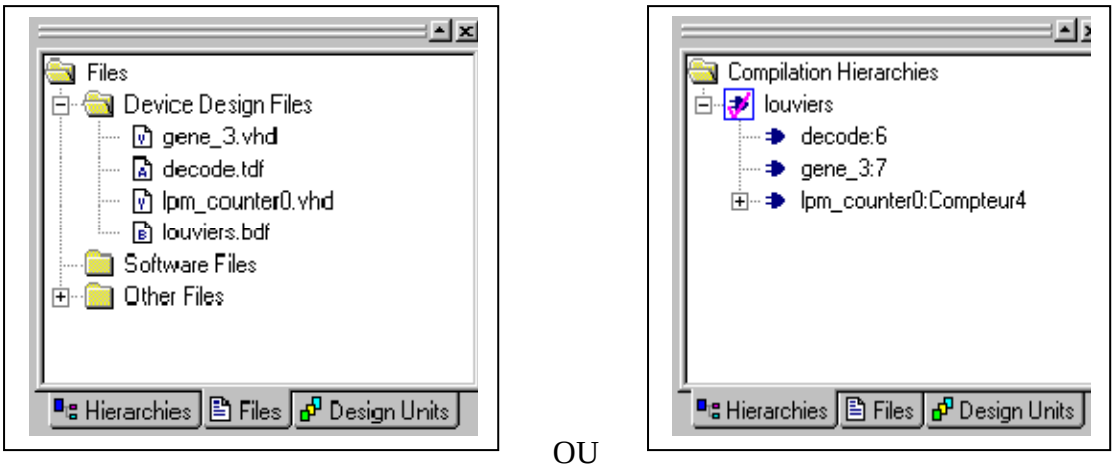


Figure III-20 : Affichage de la hiérarchie (suivant 2 onglets) [13].

La fenêtre de Quartus II est composée d'une barre de menus et de boutons (avec icônes) permettant un accès rapide aux fonctions ; le plan de travail est découpé en plusieurs fenêtres, sur la gauche, on trouve la hiérarchie du projet et l'état d'avancement des tâches, la fenêtre de droite sert de fenêtre de travail ; cette dernière dispose, sur sa gauche d'une colonne d'icônes 'contextuels".

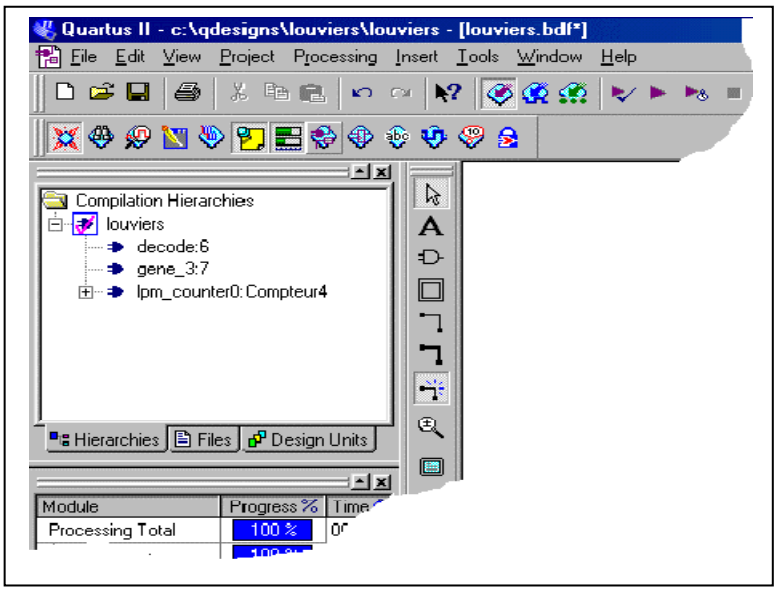


Figure III-21 : Fenêtre de Quartus II [13].

- **Saisie**

  Le logiciel permet plusieurs modes de saisie :

  Schéma : mode de saisie graphique par association de symboles fichiers associés : *.GDF ou *.BDF.

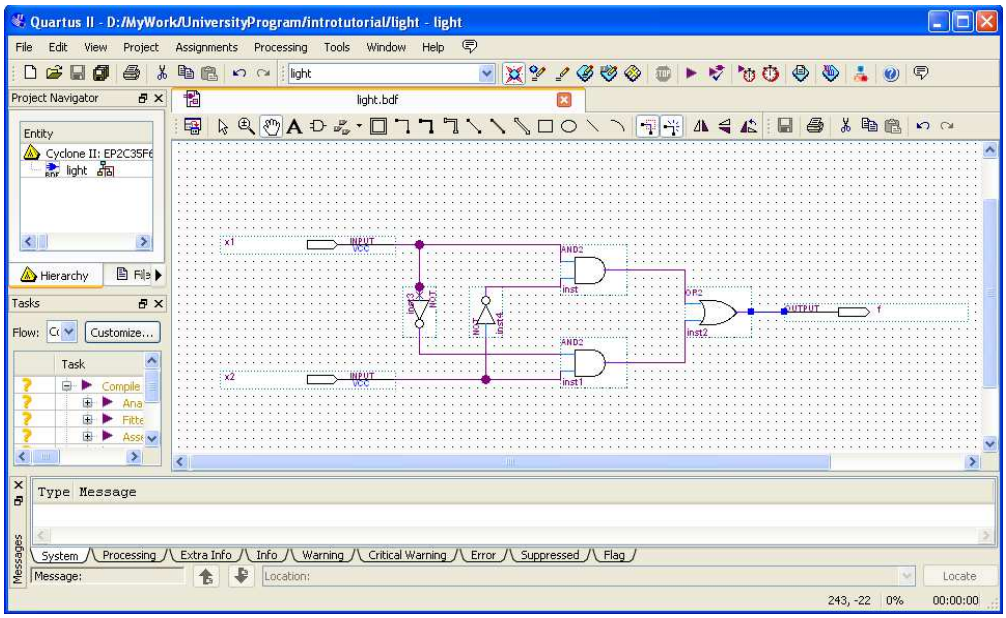


Figure III-22 : Mode de saisie schématique [8].

Textuel : AHDL et VHDL fichiers associés : *.TDF et *.VHD [8,13].

code_conv.vhd

```
1  ENTITY code_conv IS
2      PORT
3      (
4          d, c, b, a  :IN BIT;
5          p, q, r, s  :OUT BIT
6      );
7  END code_conv;
8
9  ARCHITECTURE solution3 OF code_conv IS
10 SIGNAL  input   :BIT_VECTOR (3 DOWNTO 0);
11 SIGNAL  output  :BIT_VECTOR (3 DOWNTO 0);
12 BEGIN
13     input <= d & c & b & a; -- concatenate input bits
14     p <= output(3);      -- connect each output port
15     q <= output(2);
16     r <= output(1);
17     s <= output(0);
18     PROCESS (input) -- change on input invokes process
19     BEGIN
20         CASE input IS
21             WHEN "0000"     =>      output <= "0000";
22             WHEN "0001"     =>      output <= "0001";
23             WHEN "0010"     =>      output <= "0010";
24             WHEN "0011"     =>      output <= "0011";
25             WHEN "0100"     =>      output <= "0100";
26             WHEN "1011"     =>      output <= "1000";
27             WHEN "1100"     =>      output <= "1001";
28             WHEN "1101"     =>      output <= "1010";
29             WHEN "1110"     =>      output <= "1011";
30             WHEN "1111"     =>      output <= "1100";
31             WHEN OTHERS     =>      output <= "1111";
32         END CASE;
33     END PROCESS;
34 END solution3;
35
```

Figure III-23 : Mode de saisie Textuel [8].

- **Langage VHDL**

VHDL est l'abréviation de « Very high speed integrated circuits Hardware Description Langage ». L'ambition des concepteurs du langage est de fournir un outil de description homogène des circuits, qui permette de créer des modèles de simulation et de « compiler » le silicium à partir d'un programme unique.

Initialement réservé au monde des circuits numériques, VHDL est en passe d'être étendu aux circuits analogiques Contrairement à C ou PASCAL, VHDL est un langage qui comprend le « parallélisme », c'est à dire que des blocs d'instructions peuvent être exécutés simultanément, par opposition à séquentiellement comme dans un langage procédural traditionnel. Autant ce parallélisme est fondamental pour comprendre le fonctionnement d'un simulateur logique, et peut être déroutant pour un programmeur habitué au déroulement séquentiel des instructions qu'il écrit, autant il est évident que le fonctionnement d'un circuit ne dépend pas de l'ordre dans lequel ont été établies les connexions. L'utilisateur de VHDL gagnera beaucoup en ne se laissant pas enfermer dans l'aspect langage de programmation, en se souvenant qu'il est en

train de créer un vrai circuit. Les parties séquentielles du langage, car il y en a, doivent, dans ce contexte, être comprises soit comme une facilité offerte dans l'écriture de certaines fonctions, soit comme *le* moyen de décrire des opérateurs fondamentalement séquentiels : les opérateurs synchrones. .

Deux des intérêts majeurs du langage sont :

1. Des *niveaux de description* très divers: VHDL permet de représenter le fonctionnement d'une application tant du point de vue système que du point de vue circuit, en descendant jusqu'aux opérateurs les plus élémentaires. A chaque niveau, la description peut être structurelle (portrait des interconnexions entre des sous-fonctions) ou comportementale (langage évolué).

2. Son aspect « *non propriétaire* »: le développement des circuits logiques a conduit chaque fabriquant à développer son propre langage de description. VHDL est en passe de devenir le langage commun à de nombreux systèmes de CAO, indépendants ou liés à des producteurs de circuits, des (relativement) simples outils d'aide à la programmation des PALs aux ASICs, en passant par les FPGAs.

La description qui suit est loin d'être exhaustive, héritier d'ADA, VHDL est un « gros » langage. Nous en présentons un sous-ensemble qui, nous l'espérons, doit permettre à un néophyte d'aborder ses premières réalisations avec un bagage minimum, limité à des *constructions synthétisables*, et, en principe, portables sur n'importe quel compilateur [54].

- **La gestion des librairies**

La conception des projets nécessite l'utilisation des portes logiques ou des bascules D. Ils sont disponibles dans différentes bibliothèques, et doivent être associés à la librairie propre au projet lorsqu'ils sont utilisés. Voici les deux bibliothèques de composants utilisées :

1. La librairie standard d'Altéra : Quartus propose les composants les plus basiques comme les portes de la logique combinatoire (ET, NON, OU), mais également les bascules de base (D, T) de la logique séquentielle.

2. La librairie spécifiée par l'utilisateur : Il serait fastidieux d'utiliser les composants de base en l'état. Et il serait surtout contraignant de devoir recréer toujours les mêmes blocs diagrammes. Les nombreuses structures créées lors de la conception d'un schéma sont ensuite intégrées dans cette bibliothèque. Ainsi les composants nouvellement créées sont directement disponibles pour les projets suivants.

Il est important de spécifier que l'utilisation de ces librairies est propre à Quartus, il est donc impossible l'utiliser en dehors de cet environnement. C'est le grand inconvénient de cette méthode de conceptions.

Pour un exemple, nous allons choisir le schéma pour le projet, les megafonctions pour le compteur, la description textuelle (AHDL) pour le décodeur et la description textuelle (VHDL) pour le générateur de phases.

La feuille de description du projet est alors la suivante :

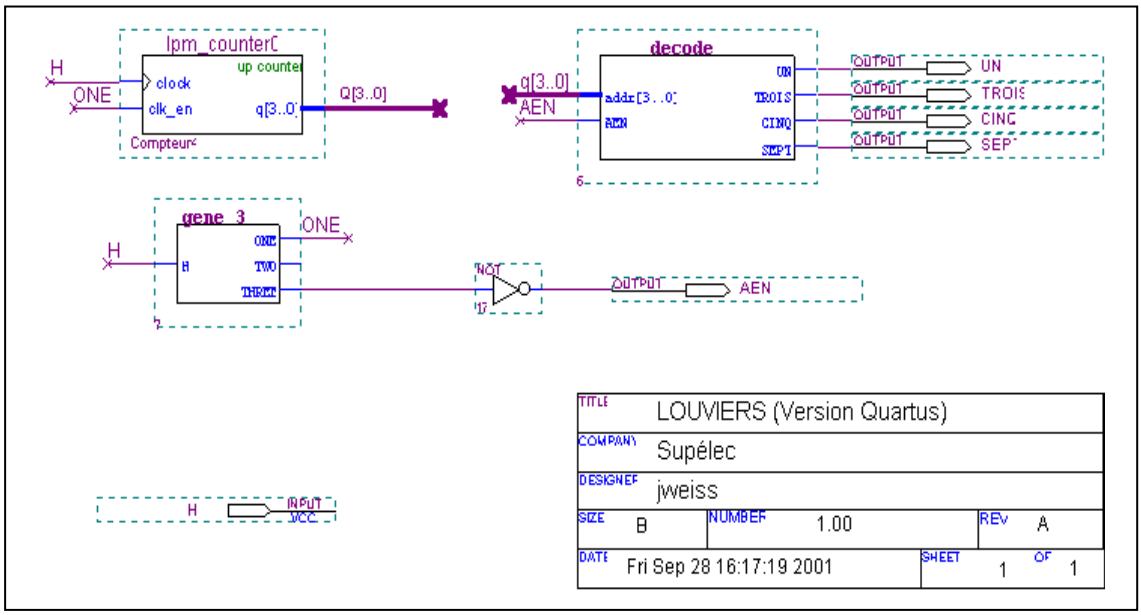


Figure III-24 : Exemple d'une feuille de description du projet (LOUVIERS.GDF) [4].

Une fois la saisie achevée, il va falloir compiler la conception [4].

III.6.6. Compilation

Le Compilateur de Quartus II est constitué d'une série de modules qui vérifient s'il n'y a pas d'erreurs dans le fichier VHDL et génèrent les fichiers pour la simulation, l'analyse temporelle et la programmation du composant.

1. Compilez le projet en choisissant la commande Start compilation (▶) du menu Processing.

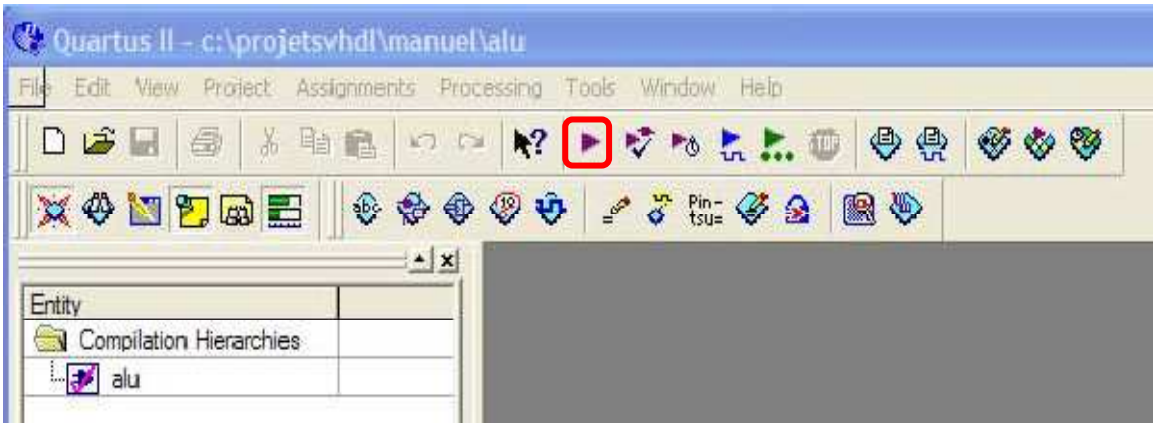


Figure III-25 : Début de processus de compilation [14].

2. Si vous recevez un message indiquant que la compilation est réussie et qu'il n'y a pas eu d'erreurs, cliquez sur OK sinon corriger votre projet et recommencer une compilation.

Remarque :

Durant la compilation, la fenêtre **Compilation Report** apparaît automatiquement. Le rapport procure des informations détaillées sur la compilation. La section **Summary** permet par exemple d'obtenir de l'information sur :

• Le statut final de la compilation

• Les requis temporels

• Le nom des entités compilées

• Le nombre total de cellules logiques, broches, mémoires et de PLL utilisés

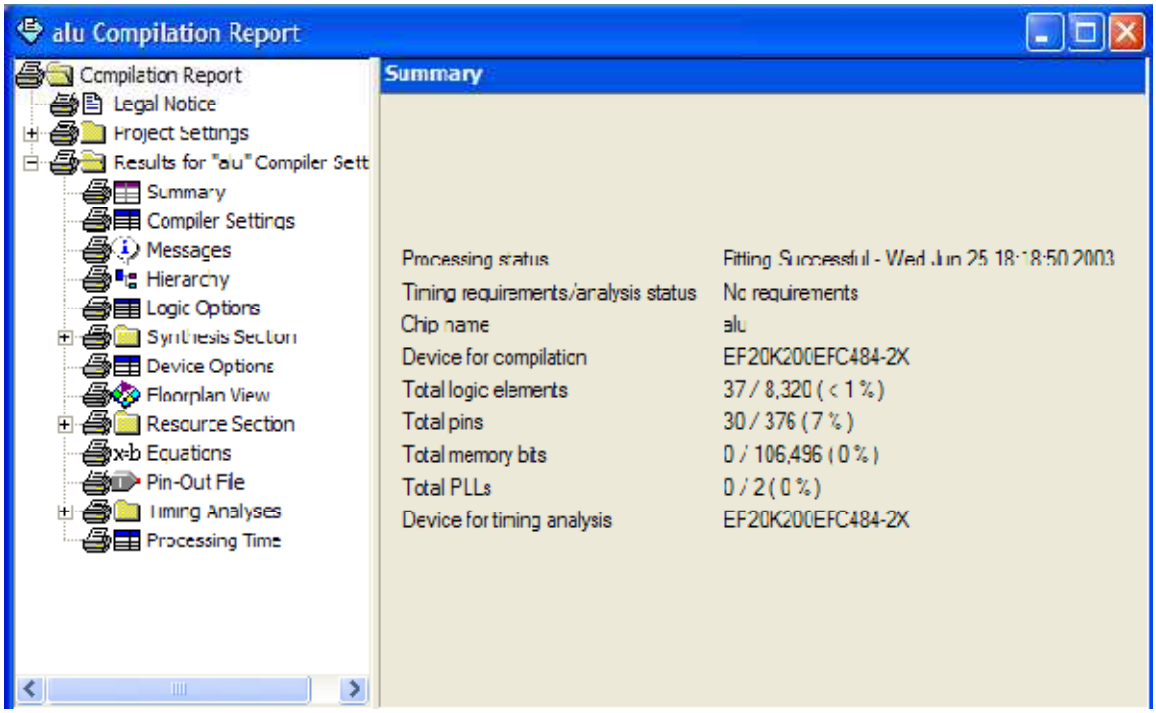


Figure III-26 : le rapport affiché a la fin de processus de compilation [14].

III.6.7. La simulation (La simulation avec Le simulateur Modelsim)

La simulation est l'étape qui suit une compilation réussie et doit normalement précéder la programmation du circuit. Elle consiste à fabriquer des vecteurs de test (les entrées à appliquer au système), puis à lancer la simulation proprement dite. A la fin de celle-ci, le simulateur affiche la réponse du circuit aux vecteurs de test sous la forme de chronogrammes.

Quartus II contient un simulateur intégré (Qsim tools), mais puisqu'il ne sera plus supporté dans les versions futures du logiciel, il est préférable d'utiliser Modelsim- Altera de Mentor Graphics pour simuler un design.

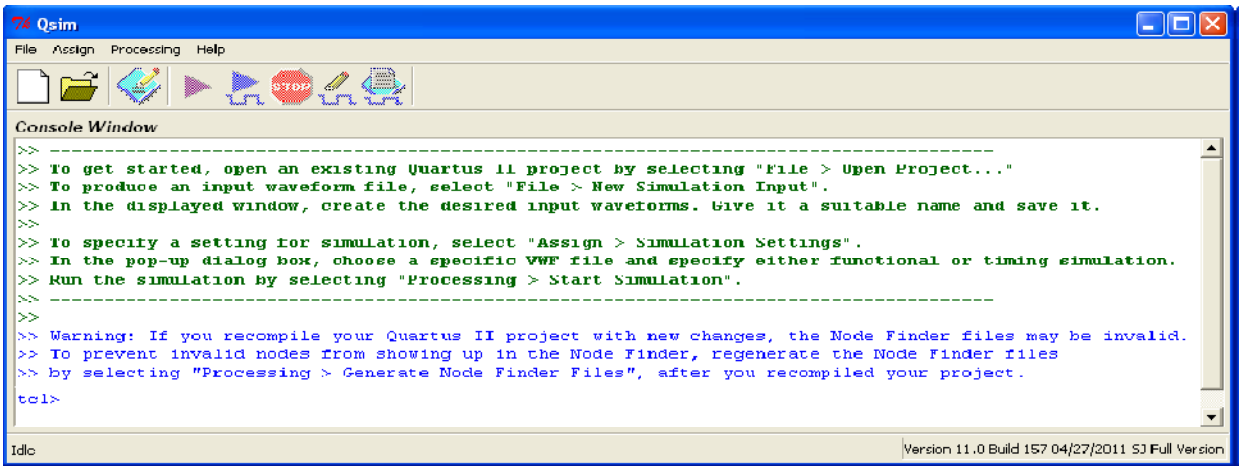


Figure III-27 : Fenêtre QSim [14].

Modelsim est un logiciel indépendant qu'on peut aussi mettre en œuvre à partir de Quartus II. Il est possible de lancer ce logiciel indépendamment ou non de Quartus II. Dans ce laboratoire, nous l'utilisons conjointement avec Quartus II. De plus, il est possible de l'utiliser avec ou sans banc de test (test bench). Toutefois, l'utilisation d'un banc de test facilite grandement la simulation et c'est donc la méthode que nous préconisons [15,16].

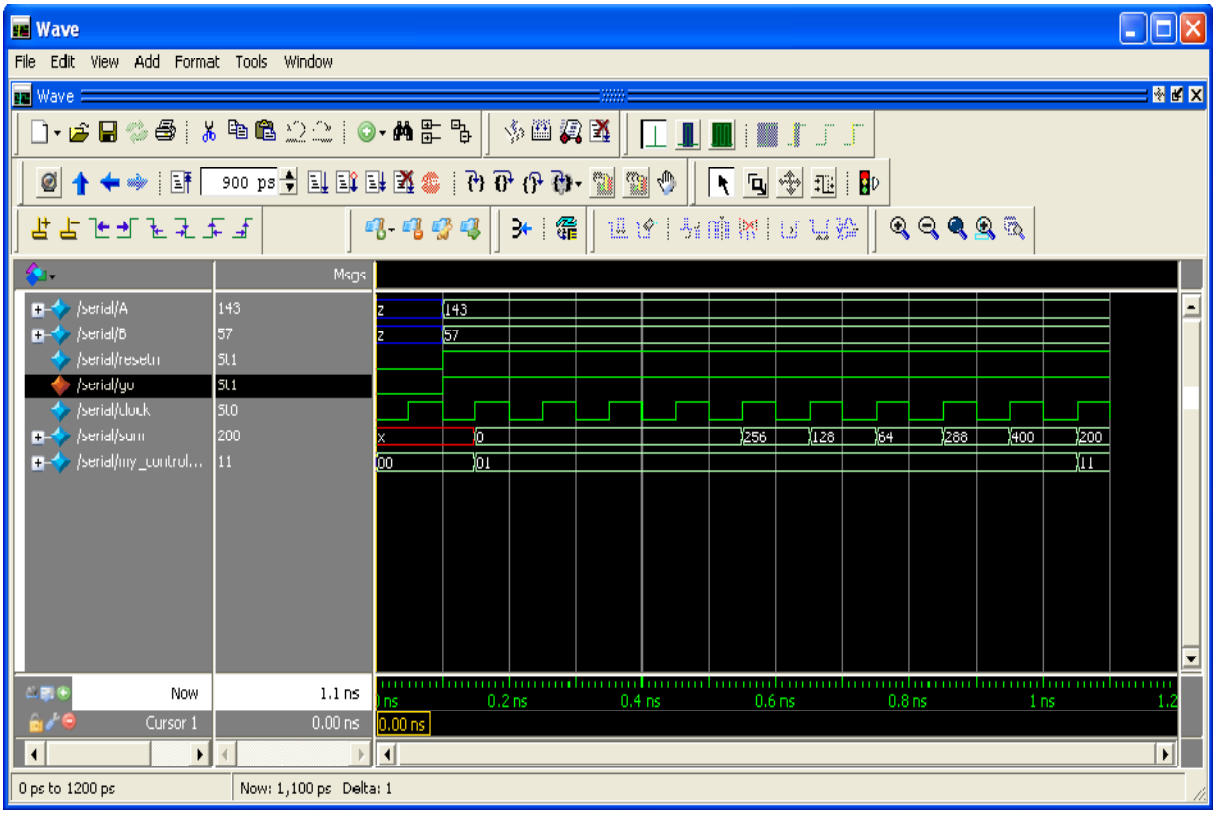


Figure III-28 : Résultat de la simulation sur le Modelsim [14].

III.6.8. Validation de l'architecture de l'algorithme (chip placement FPGA)

Une fois la génération du système effectuée, vous allez implanter l'architecture sur le circuit. Cette implantation (abusivement appelée « compilation ») se décompose en plusieurs étapes :

1. La synthèse logique, qui convertit la description VHDL de l'architecture en un ensemble de blocs logiques élémentaires, qui correspondent à la structure interne du circuit.
2. Le placement/routage qui va placer ces blocs logiques sur le circuit spécifié, puis qui les reliera entre eux au moyen du réseau d'interconnexion programmable du circuit.
3. La génération du fichier de configuration (fichier au format .sof) qui produit un fichier bitstream qui sera téléchargé dans le FPGA.

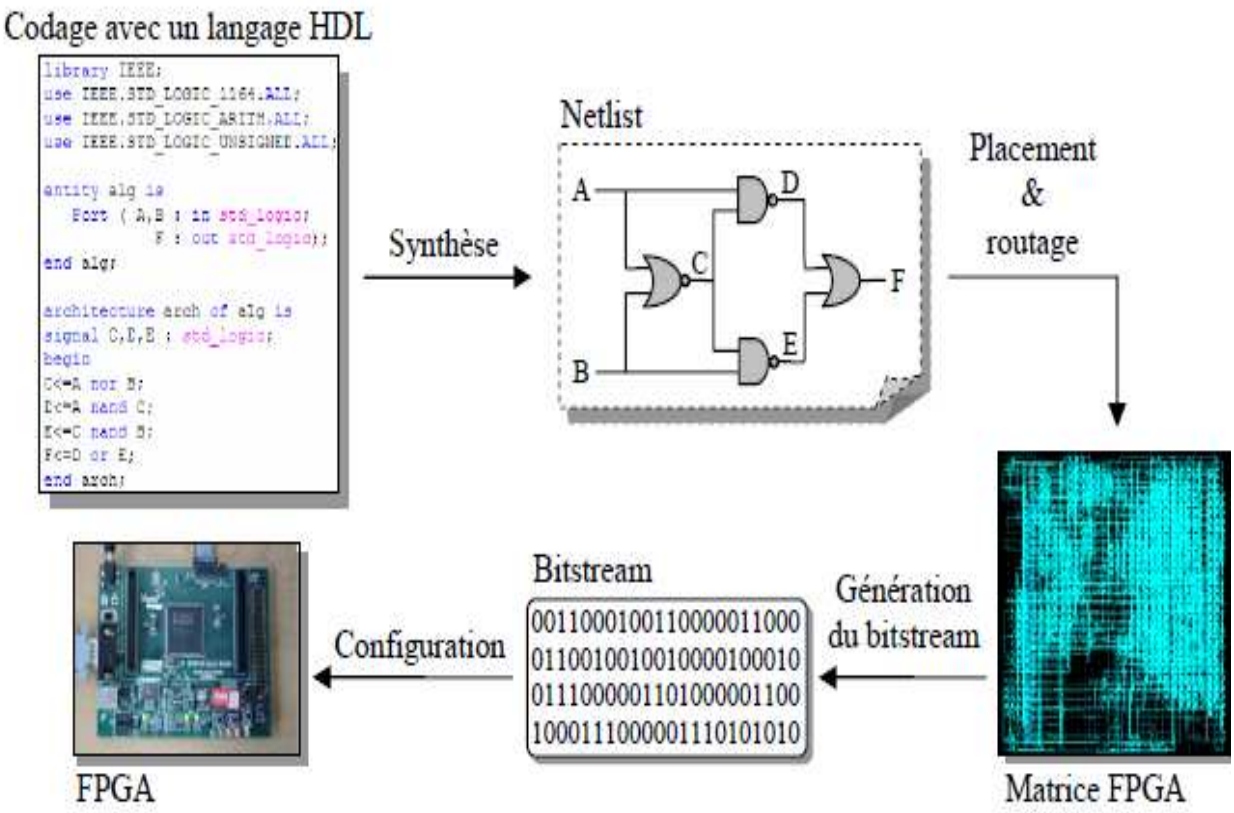


Figure III-29 : Les étapes de compilation [9].

La programmation du circuit se fait via le protocole JTAG (Joint Test Action Group). Pour cela vérifier que les connections entre le PC et la carte via le module ByteBlaster sont opérationnelles. Si tout est bon et que la carte Altera est sous tension, lancer le programmateur. Si le PC ne détecte pas la carte, une erreur doit apparaître du type Unable to scan device Chain. Hardware is not connected. Vérifier dans la figure que le fichier .sof est bien là et que la case Program/Configure est cochée, puis cliquer sur Start.

L'implantation se lance par le bouton ▶ Start de la barre d'action (en haut au milieu). Ici encore cette étape est assez longue (entre 2 et 4 minutes) [12].

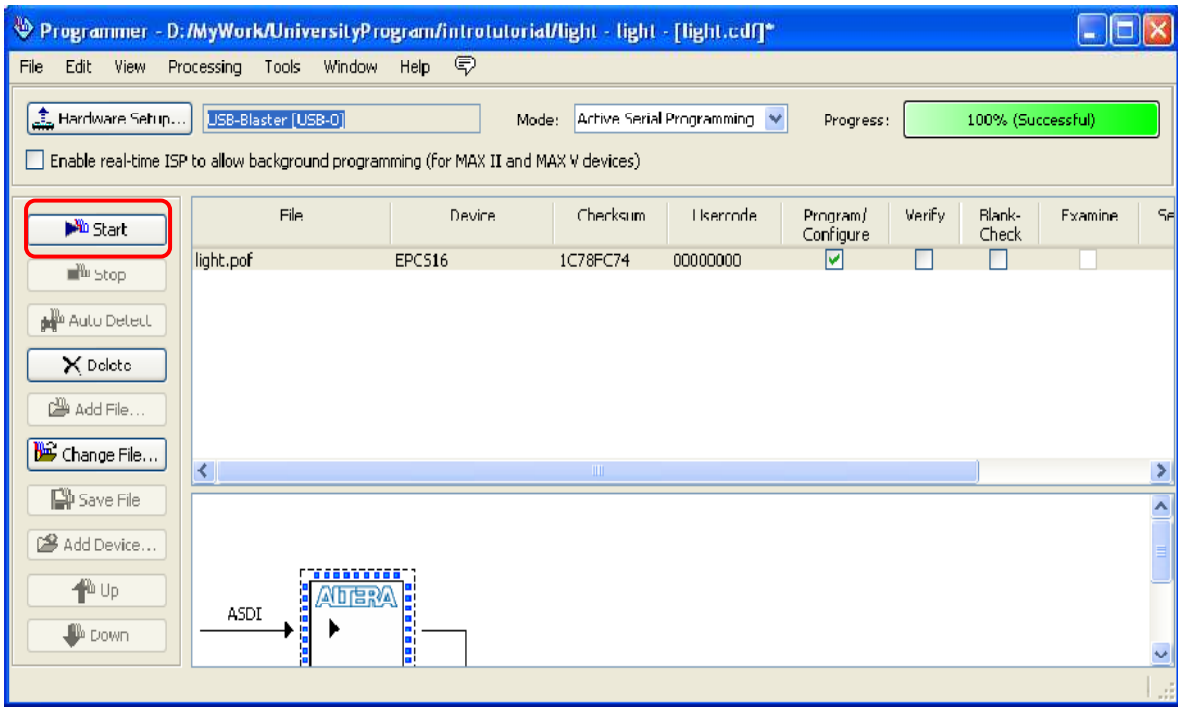


Figure III-30 : Fenêtre de fin programmation [12].

A ce stade, l'architecture est prête à être testée dans son environnement expérimental, La dernière étape de validation consiste à tester l'algorithme de contrôle implanté dans son environnement expérimental et à vérifier que les résultats expérimentaux obtenus répondent aux performances souhaitées lors de la phase de conception de l'architecture de contrôle [8,9 ,11].

III.7. Conclusion

La méthodologie développée dans cette travaille de recherche essaie de combler cette lacune : comment réaliser le pont entre d'une part l'algorithme et son architecture d'exécution, et d'autre part les composants hardware de la plateforme qui l'héberge. Le concept méthodologique adopté est présenté suivi par la description des différents éléments et outils développés pour sa mise en œuvre (langage de programmation, outil assembleur, modèles virtuel et soft-core).

L'intérêt de ce genre d'approche est notamment, comme il a déjà été cite, de permettre aux programmeurs et non-programmeurs de concevoir des applications de façon simple et rapide.

Les méthodes sélectionnées pour un projet ne sont pas toujours les bonnes. Il est important dans ce cas de remettre en question ces choix et d'évaluer l'impact de la mise en œuvre d'une nouvelle stratégie.

CHAPITRE IV

Réalisation d’un algorithme de commande pour moteur asynchrone triphasé

IV.1. Introduction

Ce chapitre a pour objectif de présenter comment sont conçus les circuits numériques complexes actuels. Ces systèmes peuvent être de plus autonomes et communicants. Les systèmes embarqués font partie de cette dernière catégorie et l'on peut s'apercevoir au quotidien que l'on est littéralement envahi par eux…

De réaliser un dispositif de commande numérique à base d'une puce d'FPGA pour un onduleur triphasé pédagogique pour prouvé et pratiqué ce que nous avant étudient dans la partie théorique, alors on a débuté dans cette partie par voire ce que nous avant comme matérielle pédagogique, surtout les cartes FPGA dans le laboratoire de faculté après on va faire des petits essais sur la carte choisie puis faire tracer la structure générale de notre programme et finalement la simulation de ce dernier.

IV.2.Choix de la carte cible

Pour pouvoir réaliser un système SoPC, on a bien sûr besoin de matériels spécifiques... Altera propose des cartes de développement pour mettre en œuvre notamment son offre de codesign. Ces cartes intègrent toutes un circuit FPGA plus ou moins gros, associé à des périphériques externes (JTAG, SDRAM, interface Ethernet, port série...). La carte choisie est une carte moyenne gamme. Le choix s'est naturellement porté sur Altera qui possède un programme universitaire pour l'enseignement et la recherche. Le choix Xilinx est aussi possible. Cela permet en fait d'utiliser les produits des 2 principaux vendeurs de composants FPGA…pour moteur asynchrone [27].

IV.3. Commande d'un onduleur triphasé

En électronique de puissance, les onduleurs de puissance deviennent de plus en plus incontournables. Ils sont présents dans les domaines d'application les plus variés dont le plus connu est sans doute celui de la variation de vitesse des machines à courant alternatif. Leur forte évolution est appuyée, d'une part, sur le développement de composants à semi-conducteurs entièrement commandables, puissantes, robustes et rapides, et autre part, sur l'utilisation quasi-généralisée des techniques dites de modulation de largeur d'impulsion.

Il existe différentes stratégies de commande de modulation. Elles peuvent êtres classées comme suit:

• Commande en pleine onde.

• Modulation de largeur d'impulsion (MU) ou PWM:

o PWM Sinusoïdale

o PWM Vectorielle ou SVPWM

Dans notre cas, dans ce chapitre, on s'intéresse à l'étude, la modélisation et la commande de l'onduleur de tension triphasé à deux niveaux en utilisant la stratégie du commande vectoriel (SVpwm).

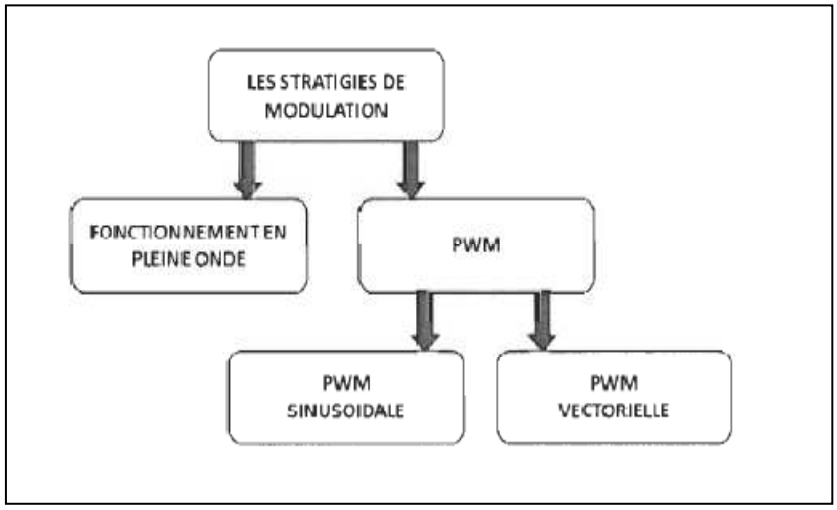


Figure IV-1 : Les différentes stratégies de modulation pour la commande des moteurs [31].

On utilise le logiciel PSIM pour analyser le fonctionnement d'un onduleur de tension à commande MLI (modulation de largeur d'impulsions). Le principe d'une commande MLI est étudié sur une structure simple (onduleur monophasé 1/2 pont), puis sur un onduleur triphasé. On termine par la simulation de la structure de puissance complète d'un variateur de vitesse pour moteur asynchrone triphasé (redresseur + onduleur) [31].

IV.4. PWM Sinusoïdale

IV.4.1. Etude d'un onduleur monophasé 1/2 pont

Le principe d'une commande MLI est étudiée sur un bras d'onduleur (onduleur monophasé ½ pont), avec une charge passive R, L série. Cette structure de base permet de comprendre le

fonctionnement d'un onduleur MLI et l'influence des grandeurs réglantes. Le schéma de simulation est donné ci-dessous :

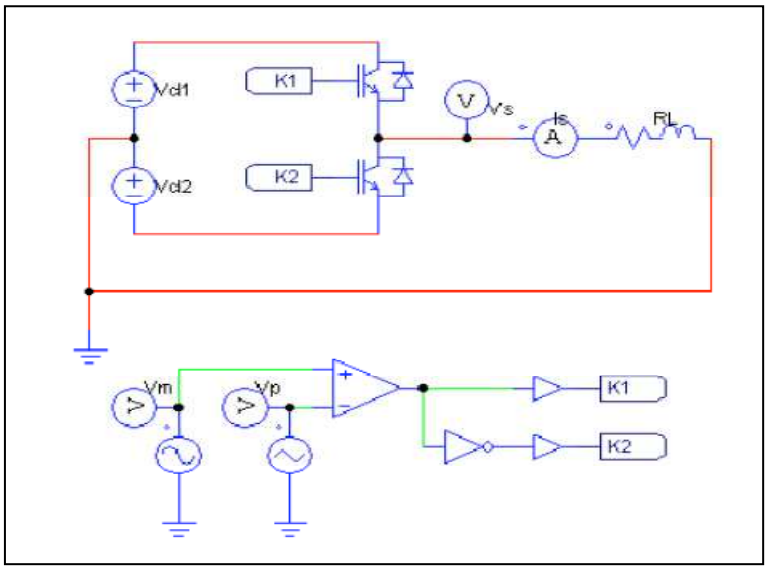


Figure IV-2 : Le schéma de simulation onduleur monophasé ½ pont sur PSIM [28].

Le circuit de puissance comprend :

- une source de tension continue à point milieu : **Vd1**, **Vd2.**

- la cellule onduleur constituée par les interrupteurs **K1** et **K2**, de type IGBT (insulated gate bipolar transistor).

- la charge **RL**, connectée entre le point milieu de la cellule onduleur et celui de la source.

Le circuit de commande comprend :

- un générateur sinusoïdal fournissant le signal de modulation **Vm**, de fréquence 50 Hz.

- un générateur triangulaire fournissant la porteuse **Vp** de fréquence 2000 Hz.

- un comparateur qui génère les signaux de commande de **K1** et **K2** à partir de Vm et Vp.

Les commandes de K1 et K2 sont complémentaires [28].

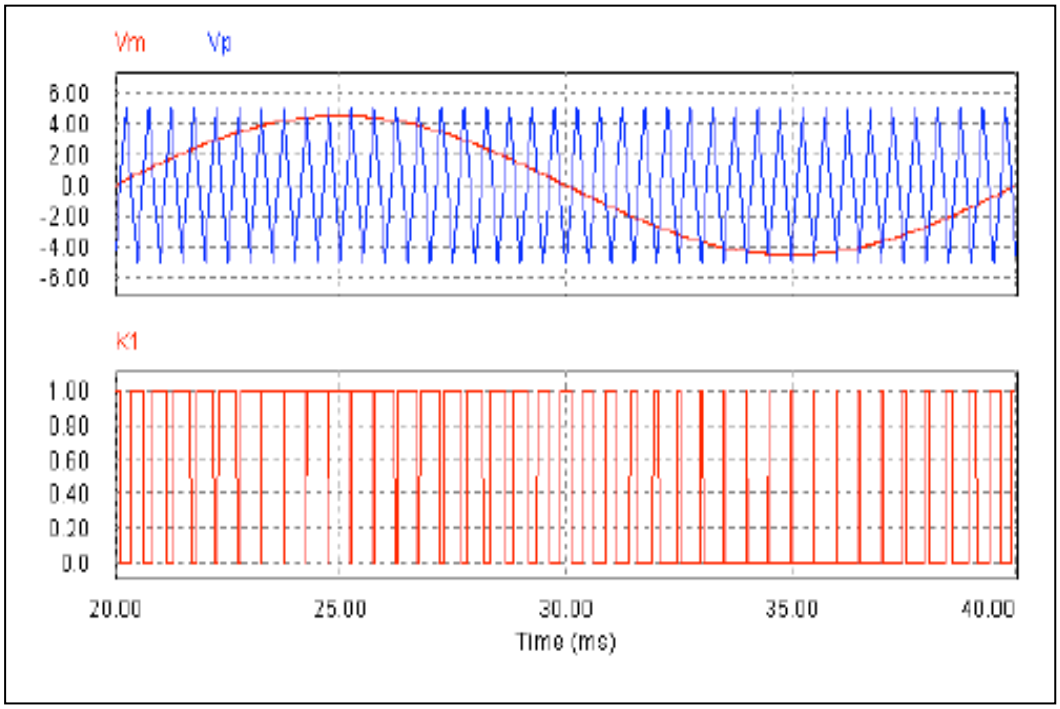


Figure IV-3 : Graphes des signaux d'entrée /sortie comparateur

Vm : signal de modulation – Vp : porteuse – K1 : commande de K1[28].

IV.4.2. Etude d'un onduleur triphasé

Un onduleur triphasé est constitué de 3 cellules identiques à celle étudiée dans la partie 1, les signaux de modulation générant la commande de chaque cellule étant décalés de $2\pi/3$. La porteuse triangulaire est commune aux 3 cellules [28].

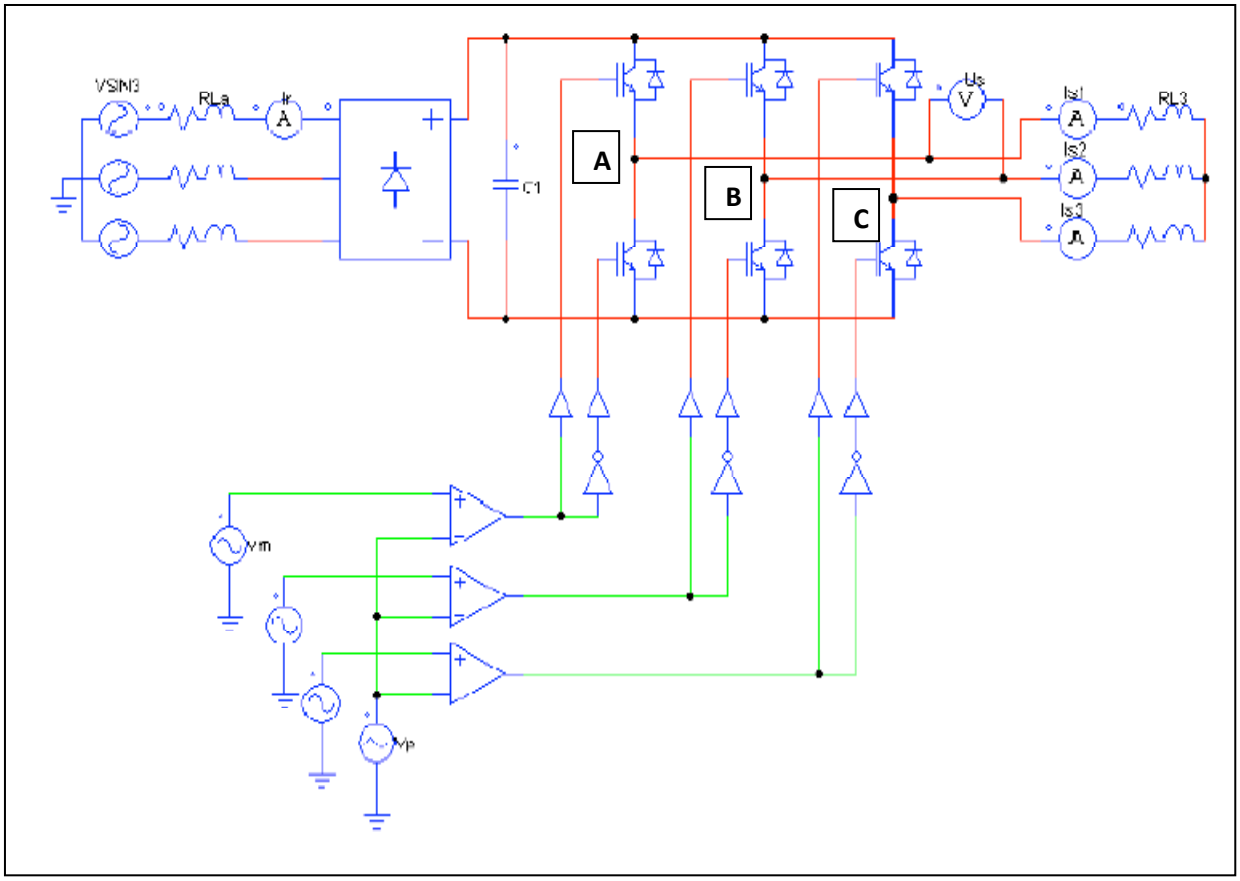


Figure IV-4 : Schéma de simulation de l'onduleur triphasé [28].

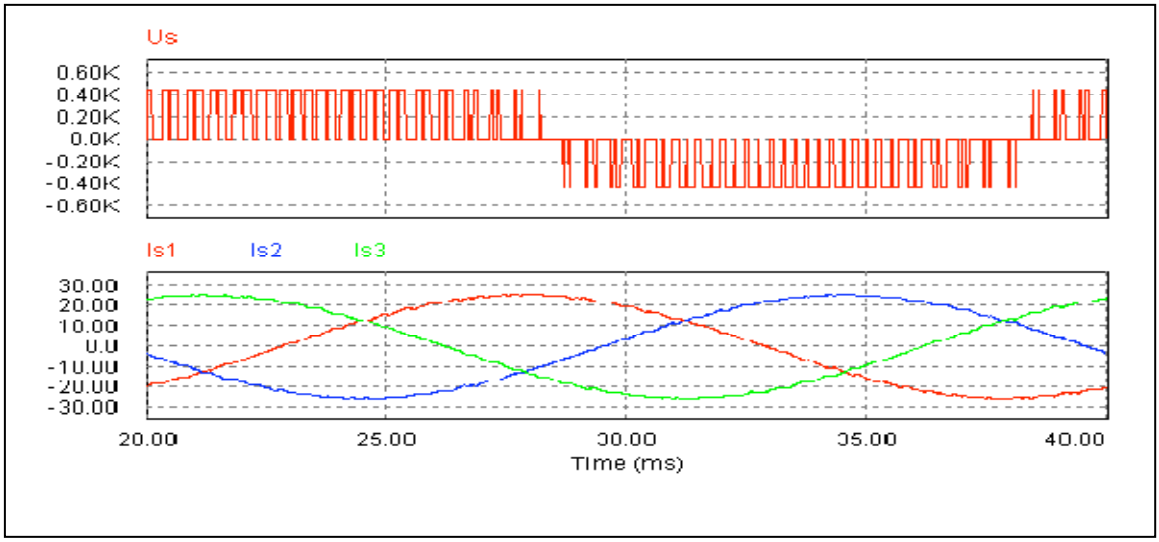


Figure IV-5 : Graphes d'une tension entre phases Us et des courants de sortie [28].

IV.5. PWM Vectorielle ou SVPWM

C'est la méthode qu'on a choisie pour notre projet, le circuit modèle d'un onduleur typique PWM (MLI) triphasé est montré sur la Figure IV-4, sont les six interrupteurs qui déterminent la sortie. Ils sont contrôlés par les variables a, a'(baratte transistor A), b, b', c et c'. Lorsque un transistor de la partie supérieure est commuté sur ON, c'est-adire quand a, b ou c est à 1, la partie inférieure correspondante est commuté sur OFF, autrement dit a', b' ou c'est à O [31].

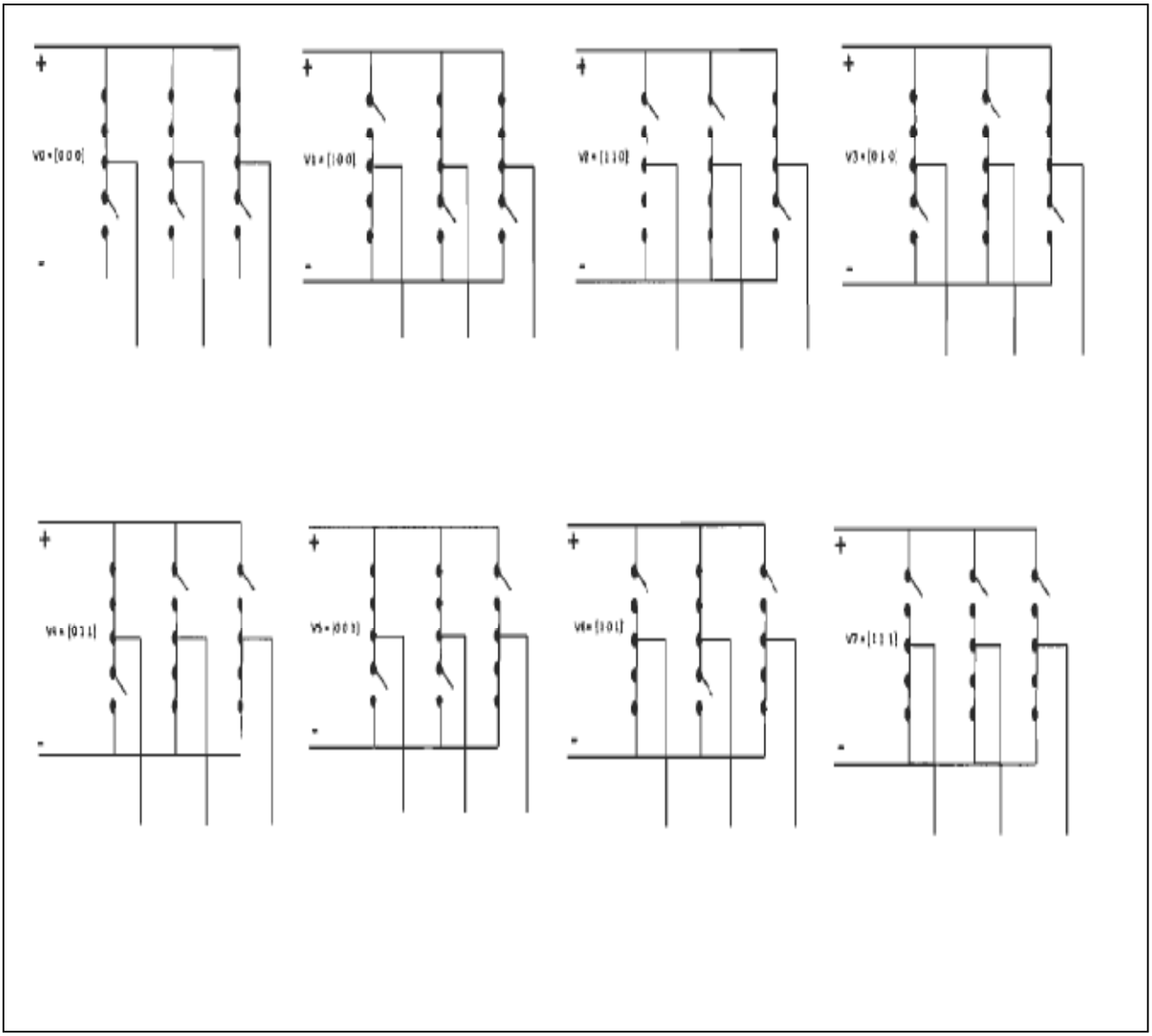
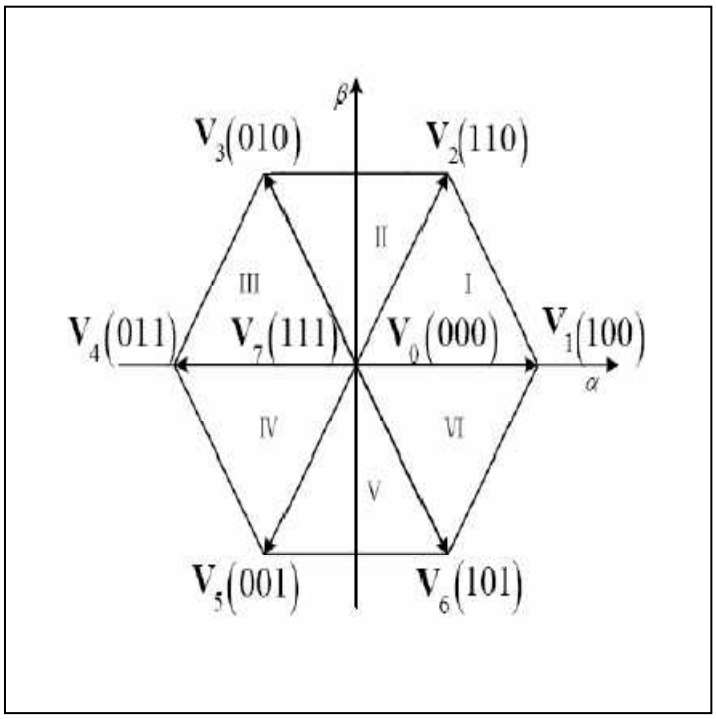


Figure IV-6 : Les huit vecteurs tensions de l'onduleur (V 0 to V 7) [31].

Vecteur tension	Vecteur Switch		
V0	0	0	0
V1	1	0	0
V2	0	1	0
V3	1	1	0
V4	0	0	1
V5	1	0	1
V6	0	1	1
V7	1	1	1

a)



b)

Figure IV-7 : a) Tableau switching vecteurs de tension. **b)** Plans vecteurs [31].

IV.6. Présentation de la plate-forme de développement (La carte choisie)

- **CARTE DE DEV DE0 FPGA CYCLONE III**
- **Silicon Fabricant:** Altera
- **Architecture du Coeur:** FPGA
- **Sous-Architecture du coeur:** Cyclone
- **Nombre de noyau de silicium:** EP3C
- **Numéro de famille Silicone:** Cyclone III
- **Contenu du kit:** Carte de développement pour EPCS4, Câble USB, CD ROM, Capot Plastique, Alimentation
- **Caractéristiques:** Connecteur Carte SD, FPGA 3C16, Transceiver RS-232

Quand vous voudrez utiliser la carte, il faudra que vous vérifiiez si l’alimentation est branchée et si la carte s’allume en pesant le bouton de démarrage. Lorsque la carte s’allume, vous verrez toutes les DEL clignoter et l’affichage hexadécimal incrémenter de 0 jusqu’à F en même temps.

P0037 est une carte de développement et d'enseignement DE0 disposant du FPGA Altera Cyclone III 3C16, la carte est conçue pour une utilisation éducative. Elle est adaptée pour une large gamme d'exercices sur la logique numérique et l'organisation PC, de la simple tache qui illustre la conception fondamentale aux conceptions complexes. Pour fournir un maximum de flexibilité pour l'utilisateur, toutes les connexions sont faites via le FPGA Cyclone III. L'utilisateur peut configurer le FPGA pour compléter n'importe quel système. Livrée avec une configuration pré-chargée pour démontrer quelques caractéristiques de la carte, ce qui permet à l'utilisateur de voir si la carte fonctionne. Le panneau de commande permet d'accéder à de nombreux composants sur la carte à partir d'un ordinateur hôte. L'ordinateur communique avec la carte via une connexion USB et peut être utilisé pour vérifier la fonctionnalité de composants sur la carte ou comme outil de débogage lors de développement de code RTL [29].

- FPGA EP3C16F484C6 Cyclone III avec périphérique de configuration série EPCS4 4Mo
- USB Blaster (embarqué) pour programmation et contrôle API utilisateur
- Sortie Vidéo (VGA 4 bits CNA)
- Port PS/2 souris ou clavier
- Embases d'extension (2 embases 40 broches)
- 8Mo de SDRAM, 4Mo de flash
- Emplacement pour carte mémoire SD
- 4 afficheurs 7 segments

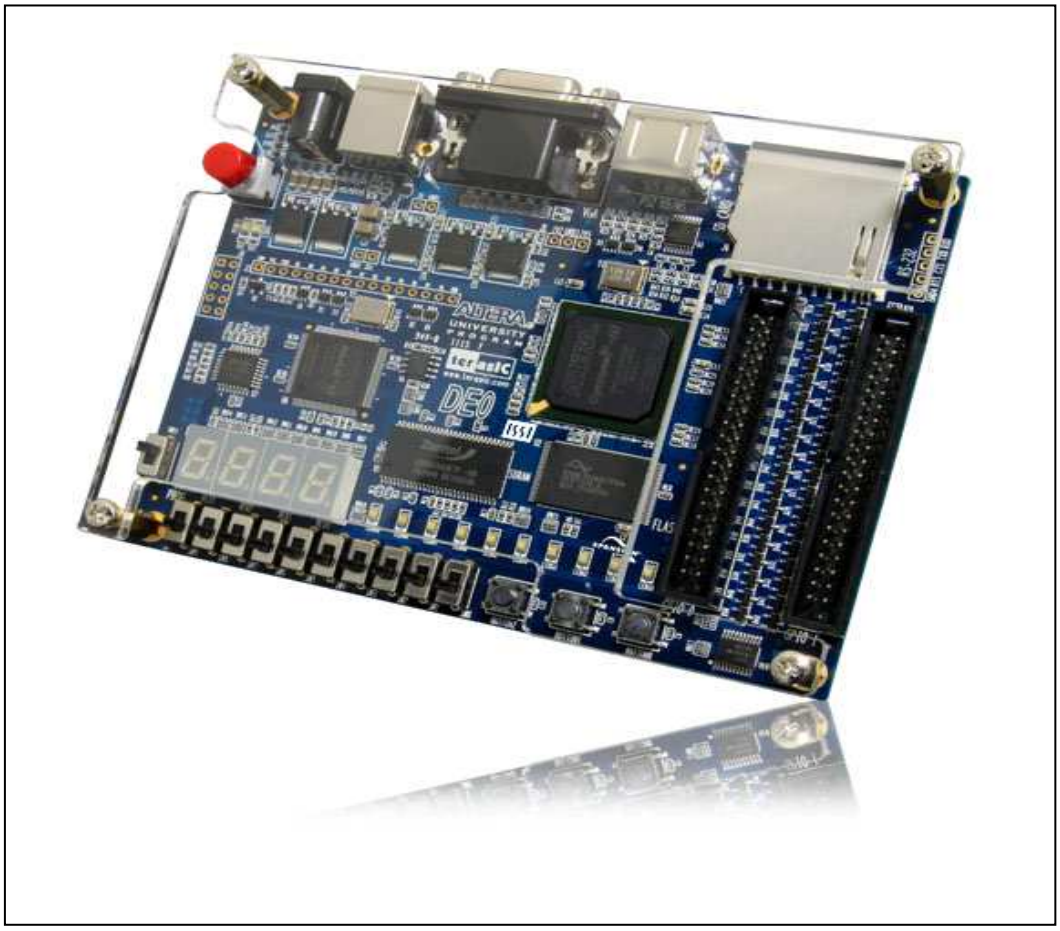


Figure IV-8 : Carte de développement et d'enseignement DE0 [29].

IV.7. Simulation

La simulation d'un circuit logique reproduit le comportement de ce dernier au sein d'un environnement contrôlé, fréquemment dénoté banc d'essai. Typiquement, il est plus facile de vérifier le fonctionnement d'un circuit dans un environnement contrôlé, et d'autre part, la compilation requise pour réaliser une simulation requiert typiquement moins de temps que celle requise pour réaliser un circuit physique.

L'outil ModelSim permet la simulation de circuits décrits en langages VHDL/Verilog. Afin de simuler un circuit décrit schématiquement, l'outil Quartus II permet la conversion automatique vers une description VHDL [30].

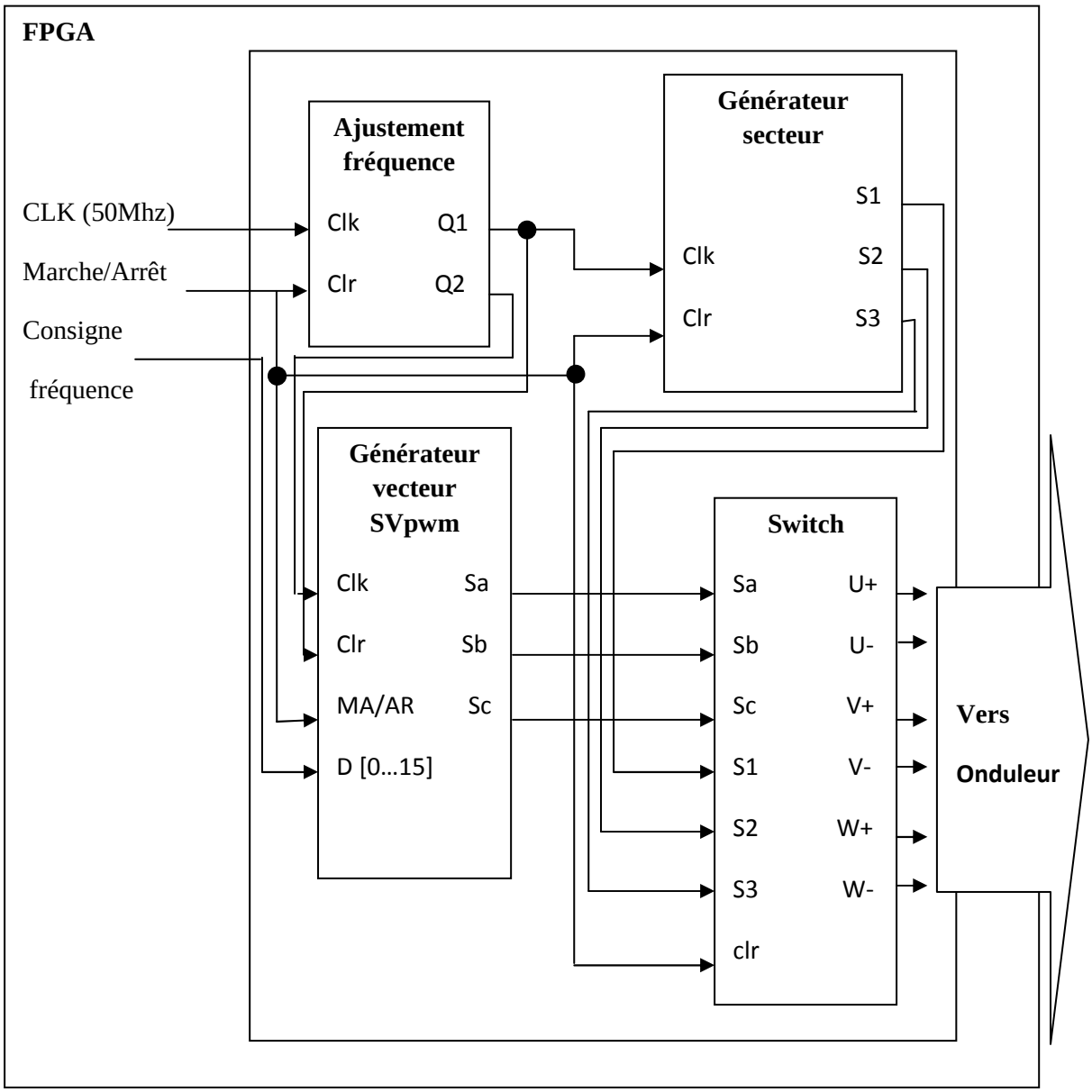


Figure IV-9 : l'ensemble de la conception du notre projet SVPWM.

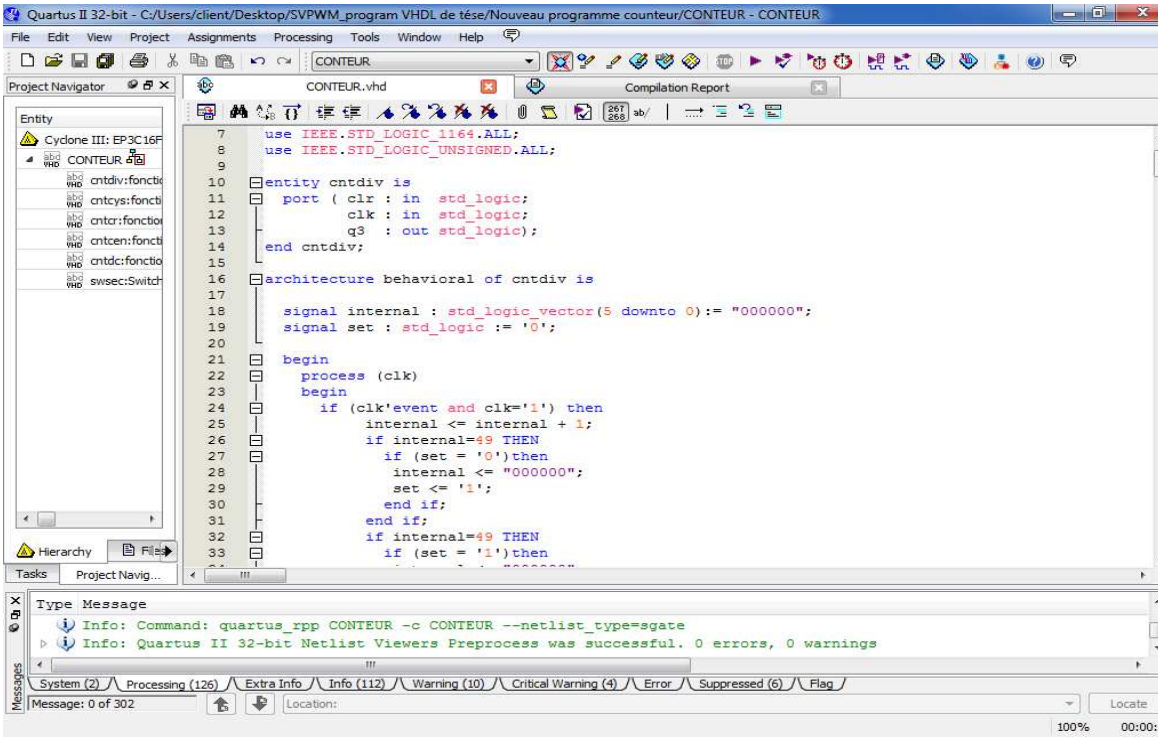


Figure IV-10 : Programme écrit en VHDL (Quartus II 11.1sp2 Web Edition (32-Bit)).

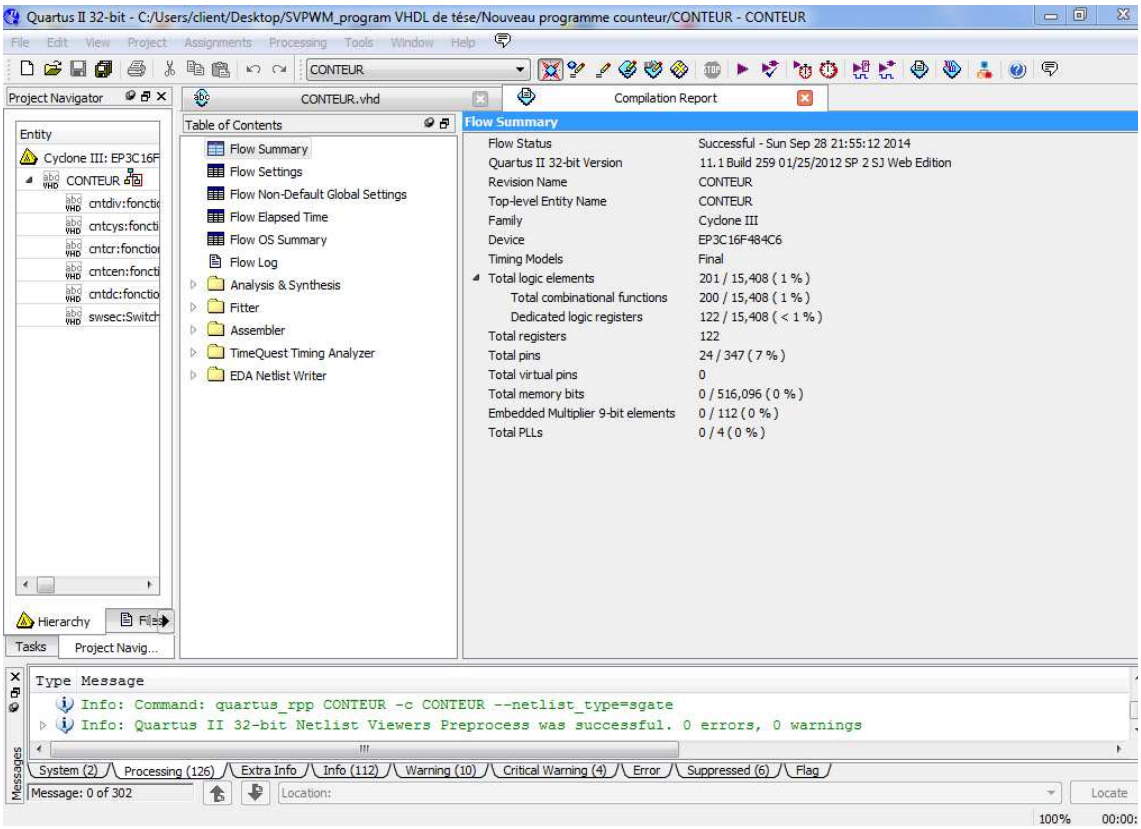


Figure IV-11 : Rapport Compilation du programme VHDL.

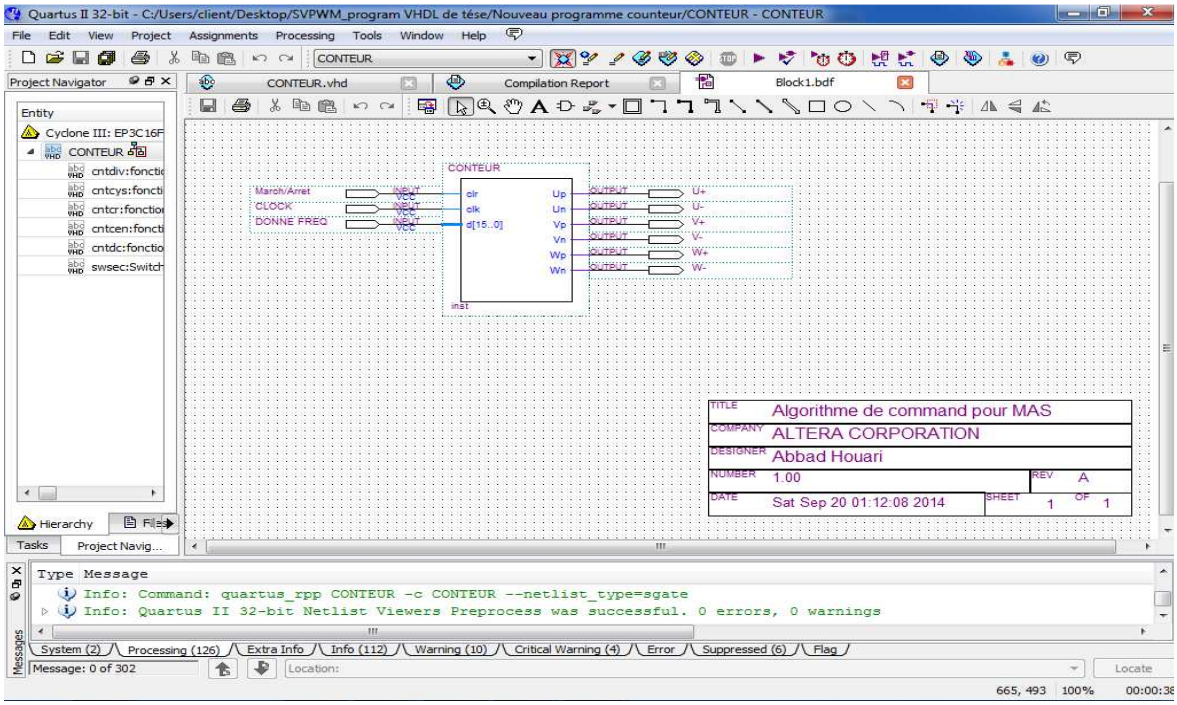


Figure IV-12 : L'ensemble du projet SVPWM on schéma block (.bdf).

Le projet ayant une description VHDL il est possible d'effectuer une simulation fonctionnelle. La simulation fonctionnelle (RTL) sera effectuée par rapport au design du concepteur. La simulation temporelle (GATE) sera effectué par rapport à la compilation et à l'optimisation des phases d'analyse et de synthèse.

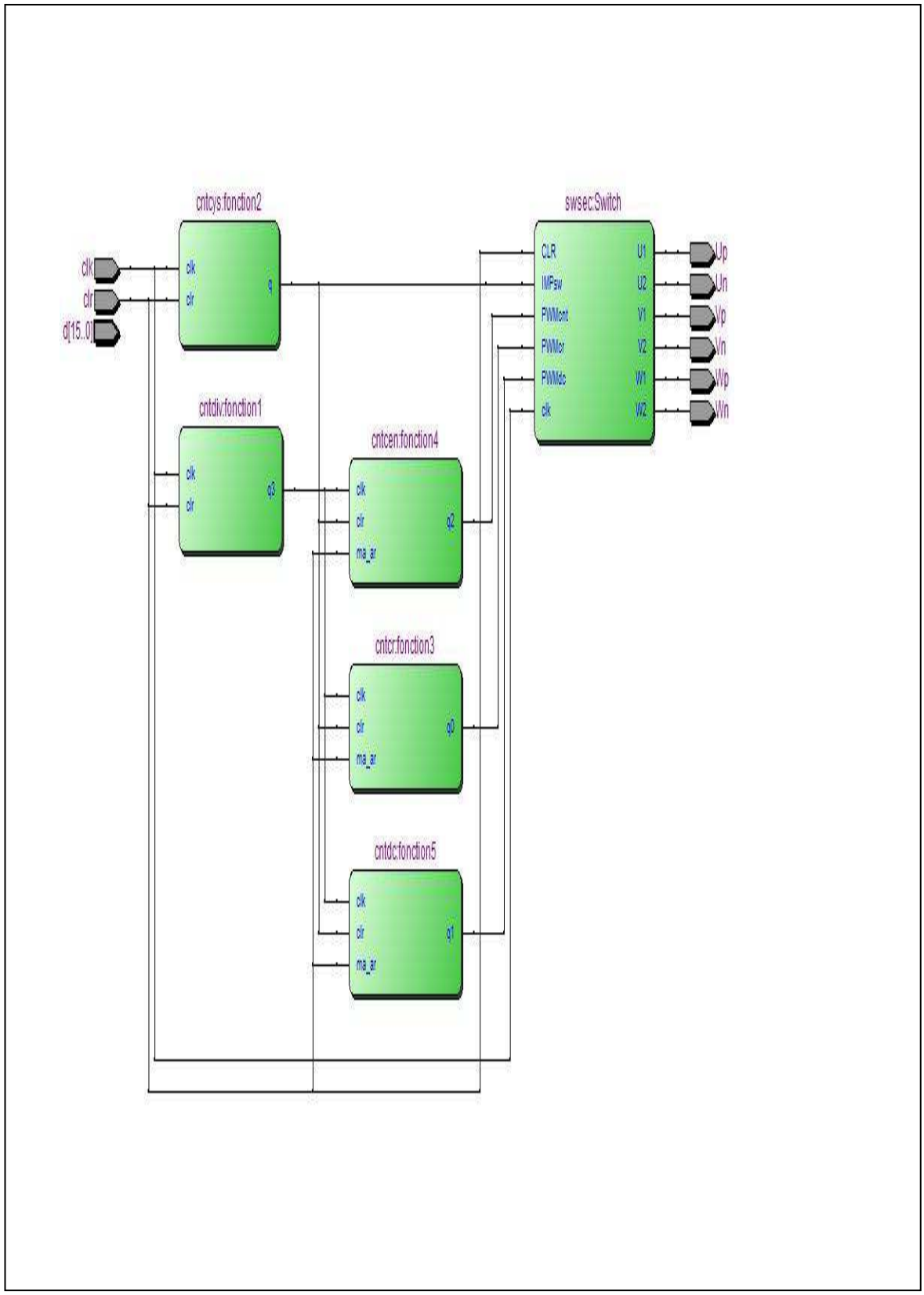


Figure IV-13 : Schéma RTL (Schéma fonctionnel) de la SVPWM.

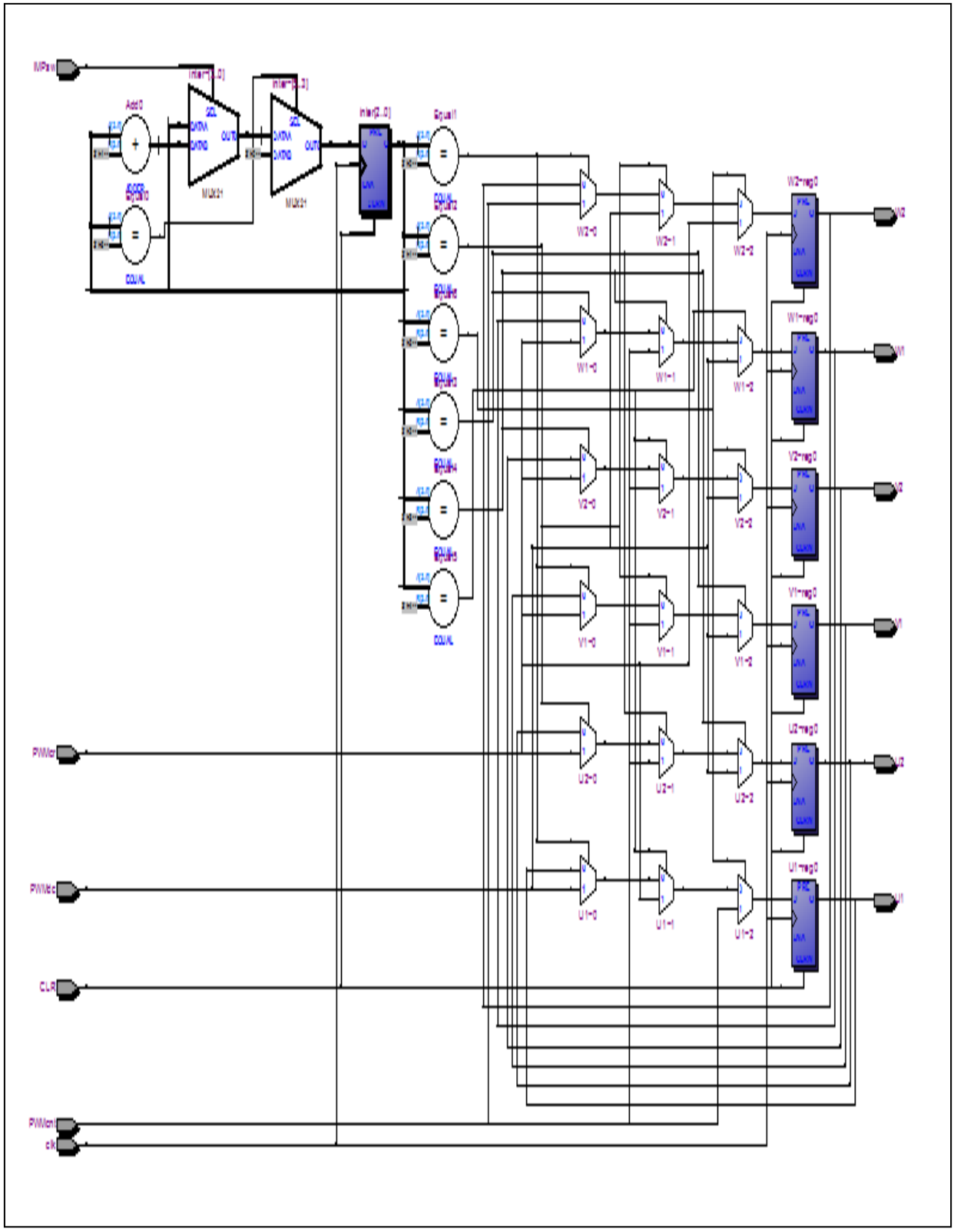


Figure IV-14 : la structure logique de block Switch.

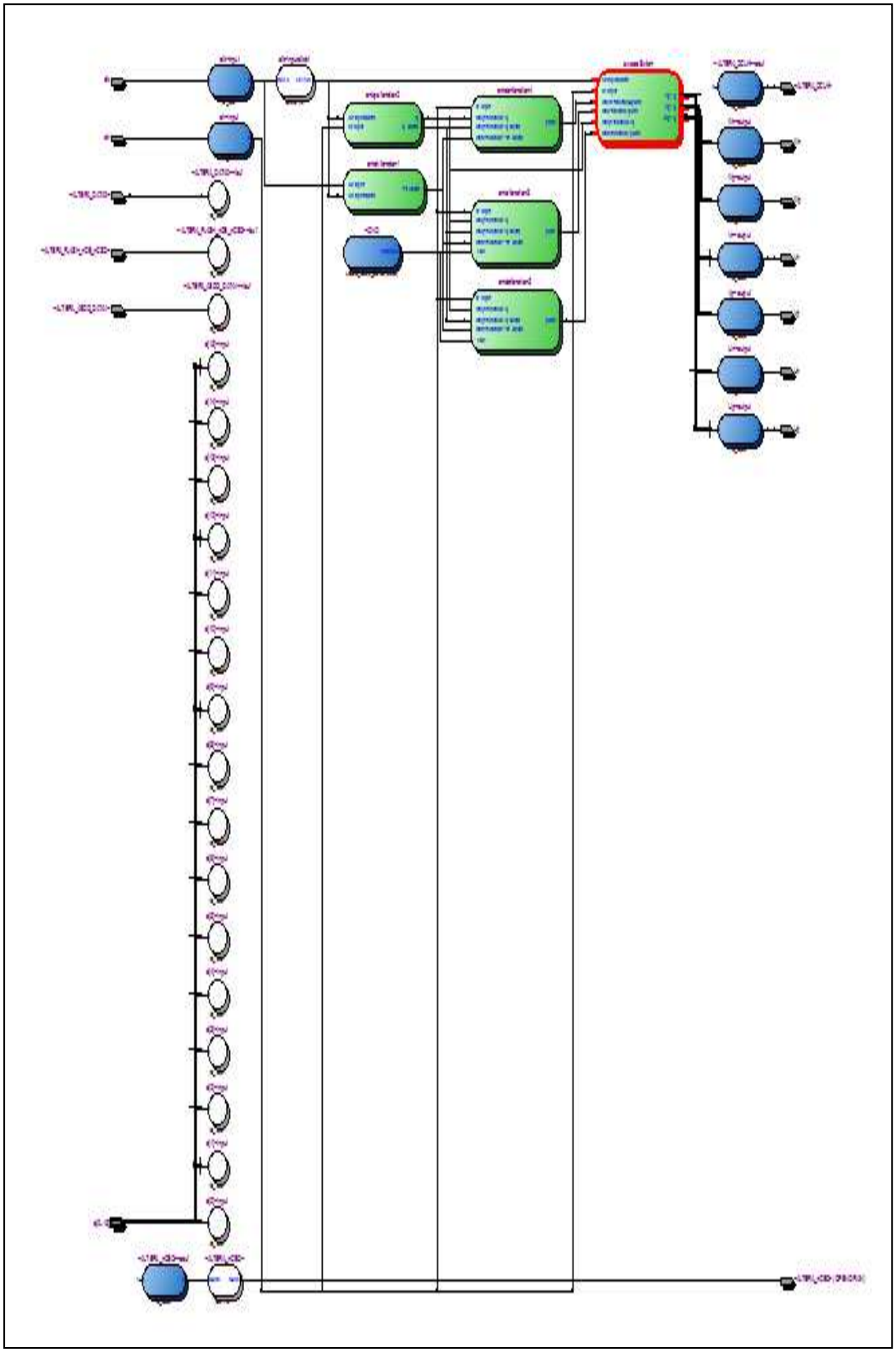


Figure IV-15 : Schéma GATE.

IV.8. Lancement de la simulation

Il est possible de lancer 3 différents types de simulations avec ModelSim-Altera depuis Quartus II. Le premier type de simulation est dit de type RTL (ou fonctionnelle), tandis que les deux autres sont des simulations au niveau des portes logiques. Il vous sera demandé d'identifier ce qui différencie ces types de simulations [30].

Le logiciel ModelSim-Altera sera lancé automatiquement, et les résultats de simulation (chronogrammes) devraient être visibles dans la fenêtre sur fond noir, illustrée à la Figure IV-16 et 17 .Pour ajuster la vue du chronogramme, cliquez dans l'espace noir et puis appuyez la touche clavier "F".

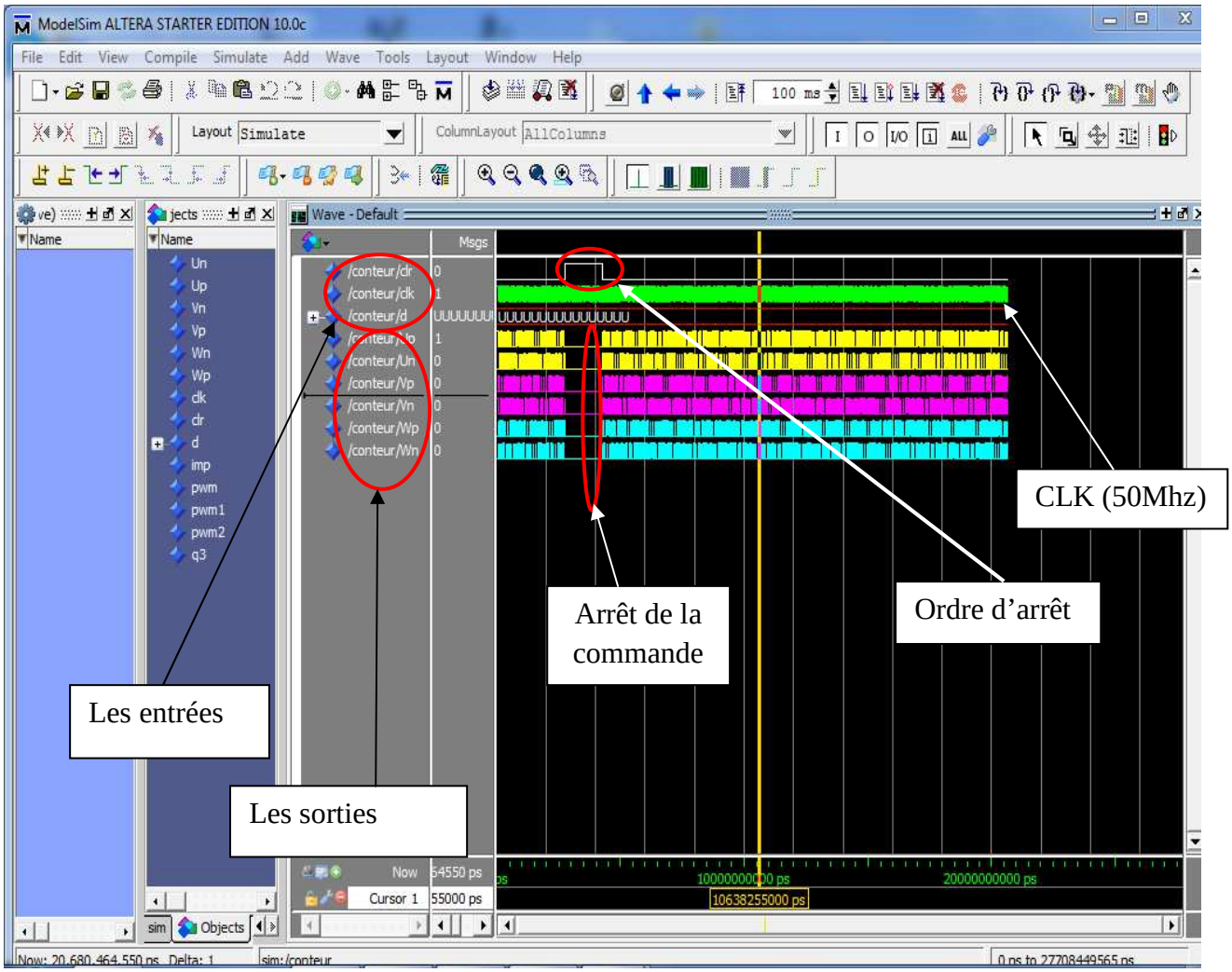


Figure IV-16 : Les résultats de la simulation sur Modelsim (ModelSim-Altera 10.0c) du générateur proposé SVPWM.

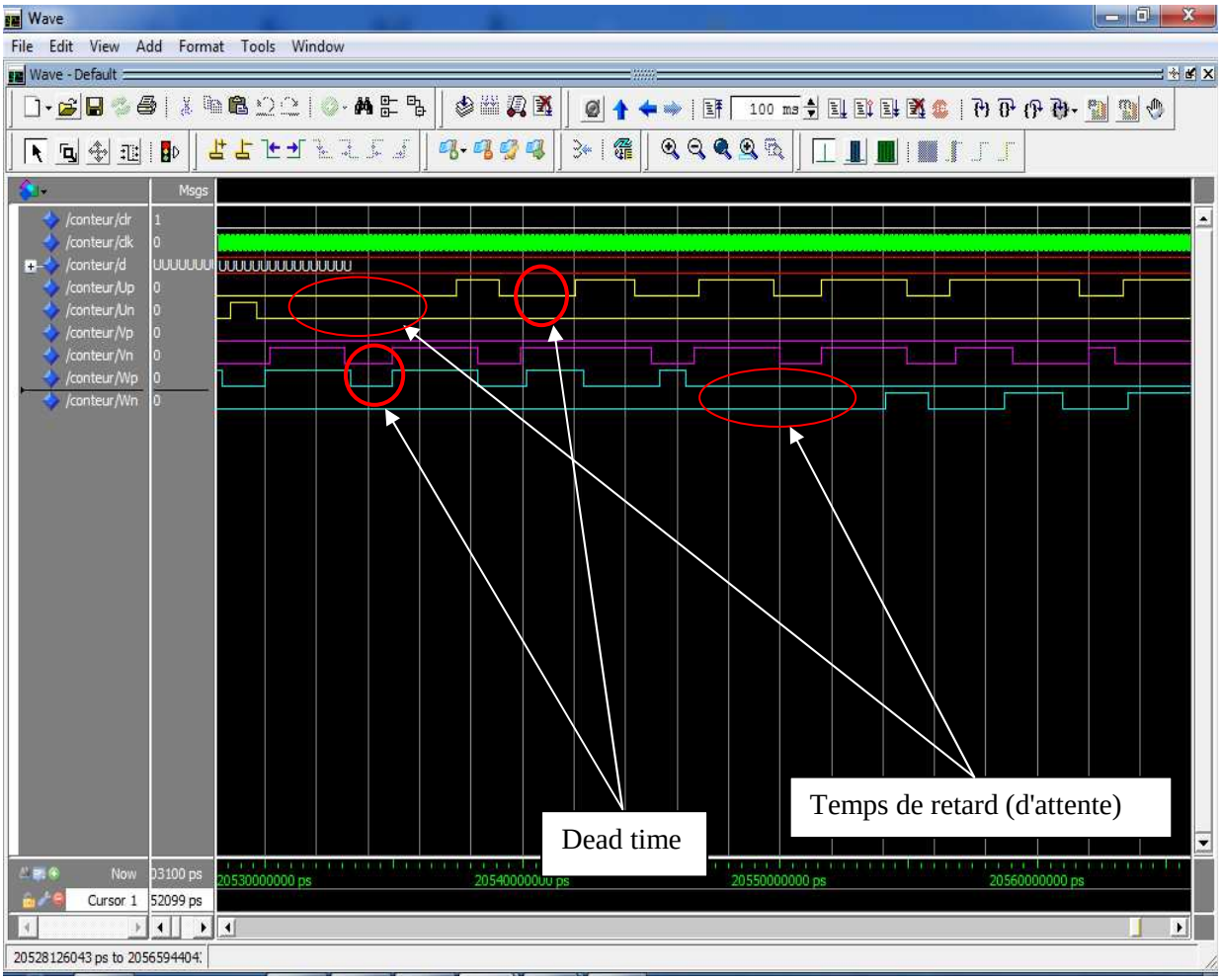


Figure IV-17 : Zoome pour la zone de Dead time (temps mort) et temps d'attente entre secteur.

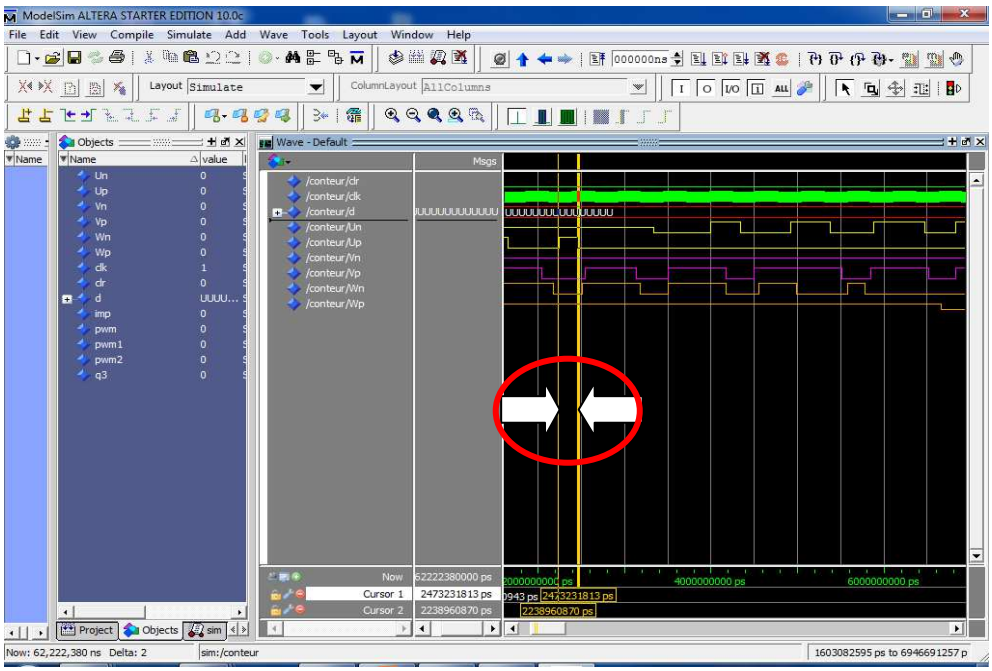


Figure IV-18 : Fréquence de commande jusqu'à 4,27khz (234 us).

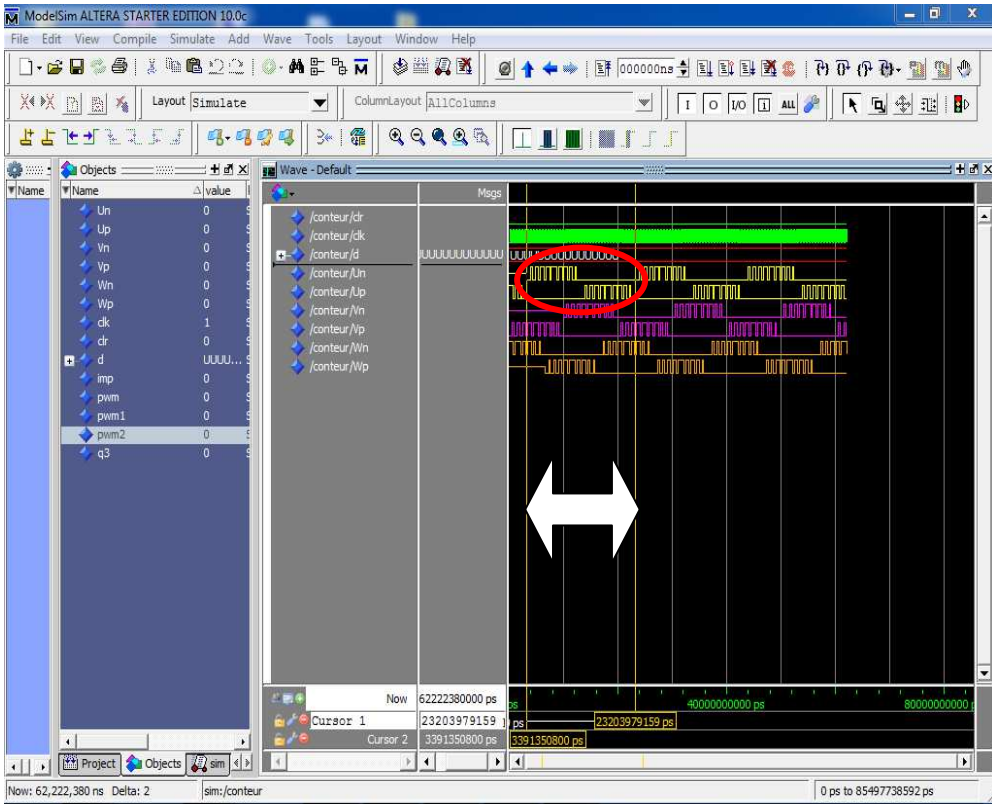


Figure IV-19 : Mesure de la période d'onde 0,02s (50Hz).

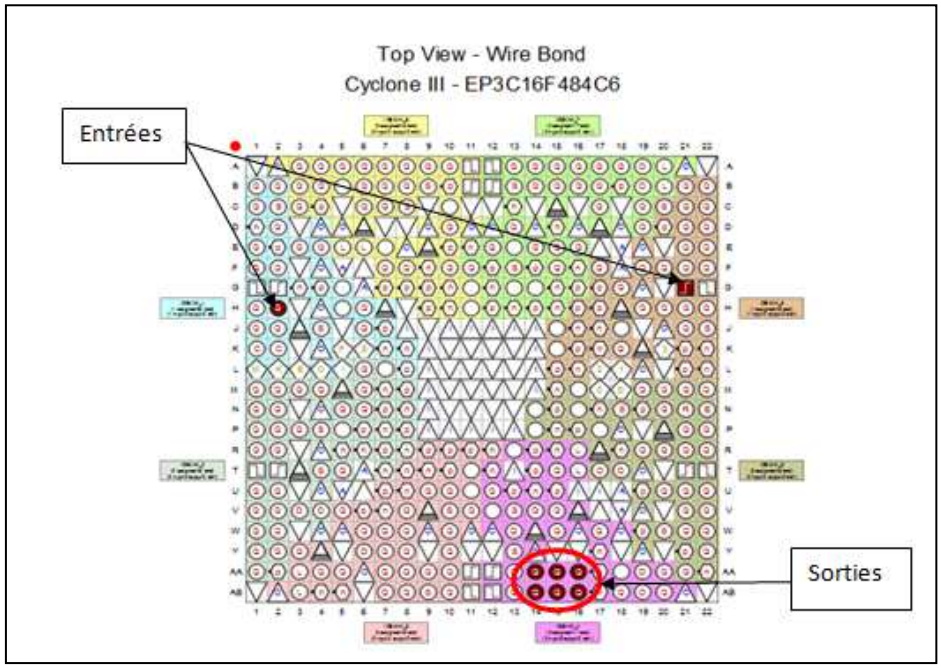


Figure IV-20 : Plan de câblage du FPGA.

IV.9. Conclusion

Ce chapitre présente la conception et la réalisation simple à base de FPGA un algorithme de commande type SVPWM, une approche de la différence, dans lequel le jugement de secteurs et du calcul de la durée d'activation pour générer le SVPWM à forme d'onde simple, et aussi les pertes de commutation est nulle (pas des harmonies indésirable). Le régime de SVPWM proposé a été conçu et mis en œuvre avec succès en utilisant carte DE0 Altera FPGA. Il est simple et sans calculer les angles de chaque secteur afin de déterminer numéro de secteur et la structure de commutation et qu'elle a programmé pour FPGA avec succès dans une fréquence porteuse variable jusqu'à 4 kHz pour une période onde de 50 Hz. En outre, ce régime a algorithme simple qu'elle peut être mise en œuvre facilement sans baser sur le FPGA mais d'une manière peu difficile.

Le circuit de commande performant basé autour d'un FPGA contribue à véhiculer une image moderne et permet une liaison avec les enseignements d'électronique.

Conclusion Générale

Conclusion générale

L'objectif de ce travail a été de mettre en évidence d'une méthode de développement et d'implantation d'un algorithme de commande sur le puce FPGA. Les travaux de recherche menés au cours de ce mémoire constituent l'ensemble des étapes indispensables pour l'implantation d'un algorithme de commande sur circuit FPGA d'une manière générale est sur la puce FPGA Altera en particulier.

Après avoir les généralités sur les réseaux logiques programmables combinatoires (PAL, PLA, PROM) et les réseaux logiques programmables séquentiels (PLD, FPGA, ASIC…) dans le premier chapitre, une étude sur les circuits FPGAs a été présentée au deuxième chapitre tel que la structure interne de puce, les principes fabricants et les différentes applications de ces circuits.

L'implantation des algorithmes de contrôle dans leur intégralité sur des cibles matérielles telles que les FPGA est une démarche qui nécessite une parfaite maîtrise des processus de conception et un travail spécifique d'adéquation entre l'algorithme et l'architecture de commande à intégrer. Donc, un savoir-faire méthodologique est nécessaire aux concepteurs utilisant les composants du type FPGA afin de satisfaire l'ensemble des contraintes inhérentes de l'implantation, tout en apportant une flexibilité de développement suffisante. Afin de réduire la complexité de l'utilisation des solutions FPGA, ce problème est surmonté, au troisième chapitre, un travail méthodique et moins intuitif a été utilisée. à travers l'utilisation de ce travail qui est basé sur une méthode de développement appropriée qui permet de répondre aux différentes contraintes de conception architecturales des algorithmes de commande.

Plusieurs méthodologies de développement pour la conception d'architectures matérielles. Elles ont toutes été conçues en ayant comme objectif le développement d'architectures génériques et réutilisables afin de pouvoir les réutiliser dans différentes applications. La notion de réutilisabilité est toujours de première importance étant donné qu'elle permet de créer une bibliothèque de modules réutilisables appelés aussi fonctions IP (Intellectuel Property). La spécificité de la méthodologie de développement utilisée dans ce travail est qu'elle est facile à appréhender par l'ingénieur électrotechnicien sans qu'il soit expert en microélectronique. Les étapes de développement de l'architecture à implanter sont principalement effectuées via le logiciel Matlab-Simulink ainsi que les outils CAO des solutions matérielles.

La méthode dont nous les mettons dans notre projet est une collection d'étapes en cascade, chaque étape est une station, nous pouvons débuter de toute étape veut et annuler les étapes avant, cela dépend des données que nous avons, les exigences du projet et les objectifs que nous voulons atteindre.

Le dernier chapitre et un projet basé sur une commande directe (boucle ouverte), une table de commutation utilisant six vecteurs actifs de l'onduleur de tension a été synthétisée au moyen de la théorie du mode de commande SVPWM, ce dernier est caractérisé par une fréquence de commutation variable qui nous donne la possibilité de produit des formes d'onde sinusoïdale presque parfaites (forme d'onde courant de sortie onduleur), son implantation sur cible FPGA permis d'avoir un temps d'exécution réduit a quelques microsecondes par rapport aux autres solutions, cet avantage à nous offre une grande bande de fréquences de commutation quelle que soit la complexité du programme avec des impulsions calculer pour respecter les capacités des barrettes transistor d'onduleur (pas des harmonies indésirables), alors une très bonne stabilité aux systèmes de commande .

Pour réussie sur un projet de boucle de commande avec une FPGA il faut utiliser des cartes qui sont dédiés à la commande des systèmes électrique pour assuré à un meilleur résultat puisque ces nouvelle cartes FPGA embarquent des parties analogiques, dans la plupart des cas il s'agit de convertisseurs analogiques numériques aux performances limitées qui permettent au composant d'évaluer son environnement (température, pression etc…) , ce qui permet de connecter directement le FPGA à un nombre plus important de sources. Il existe aujourd'hui des FPGA spécialisés sur l'acquisition et le traitement numérique du signal, pour l'utilisation de processeurs embarqués, pour les communications très haut débit (GigaBits Ethernet) etc…

Les méthodes sélectionnées pour un projet ne sont pas toujours les bonnes. Il est important dans ce cas de remettre en question ces choix et d'évaluer l'impact de la mise en œuvre d'une nouvelle stratégie.

Des évolutions ont eu lieu dans le domaine des applications, parmi celles-ci il est à noter l'importance croissante des FPGA dans le domaine de la sécurité et de la cryptographie appliquée. Cette évolution a entrainé d'importantes questions en ce qui concerne la sécurité des FPGA et de leur configuration, de nombreux travaux sont en cours dans ce domaine.

Bibliographie

[1]. L’enseignement des systèmes numériques complexes _Patrice Kadionik, Patrice Nouel, Ahmed Ben Attitalah, Philippe Dondon _ ENSEIRB-IXL BP 99 33402 TALENCE Cedex kadionik@enseirb.fr.

[2]. La Magazine française Electronique N7 juillet_Aut2010.

[3]. NATIONAL INSTRUMENTS,NI-Tutoriel 8043-fr.pdf.

[4]. 2011. TH16773.mariani.johanna.pdf.

[5]. Lilian Bossuet, 2010. Les FPGA. Technologie, architecture et utilisation.pdf.

[6]. uuu.enseirb.fr/~nouel/sopc/textes_apex/cournios.pdf.

[7]. Thèse présentée par Mouna BAKLOUTI KAMMOUN Méthode de conception rapide d’architecture massivement parallèle sur puce, Ecole Nationale d’Ingénieurs de Sfax & Université Lille 1 Sciences et Technologies.pdf.

[8]. WWW.ALTERA.COM.

[9]. Commande numérique à base de composants FPGA d’une machine synchrone_Mohamed Wissem NAOUAR L'UNIVERSITE DE CERGY PONTOISE.pdf.

[10]. WWW.MATHWORK.COM.

[11]. Utilisation du NIOS_ DIIC2-ARC_INSTITUT DE FORMATION Supérieur d’informatique et de communication. PDF.

[12]. notice de prise en main de logiciel QuartusII.pdf.

[13]. Présentation du logiciel Quartus II (Altera) ; Version 0.01; J. WEISS, octobre 2001.pdf.

[14]. Quartus II doc d’utilisation VHDL.pdf.

[15]. Mini-laboratoire 1: Introduction a Quartus II et Modelsim-Altera _ c automne 2011 Sebastien Roy et Isabelle LaRoche.

[16]. Manuel d'utilisation Chaîne de synthèse Quartus II Simulateur Modelsim _Kit CPLD iMaxII /carte DE2.

[17]. Thèse présentée par Fabio Dias Real de Oliveira_ Conception d'une méthodologie d'implémentation d'applications de vision dans une plateforme hétérogène de type Smart Camera.

[18]. Les FPGAs_ Principes innovants et tendances_ Jean-Luc Danger.

[19]. http://www.ni.com/white-paper/8043/fr/.

[20]. Electronique magazine _ janvier 2004 N 143.

[21]. Logique programmable_ université Montpellier II.

[22]. Institut Universitaire de Technologie de CRETEIL-VITRY Département de Génie Électrique et Informatique Industrielle MC-ENSL1 - Composants programmables complexes COURS / TP FPGA.

[23]. Thése_ Thoma_chapitre 2.pdf.

[24]. Mise en œuvre du SoPC sur composants FPGA Altera et Xilinx_Patrice NOUEL Patrice KADIONIK_www.enseirb-matmeca.fr/~kadionik.

[25]. Laboratoire 1 : utilisation du logiciel Quartus II d'altera.pdf.

[26]. http://www.lembarque.com/le-marche-des-fpga-pourrait-croitre-de-8-5p-par-an-jusquen-2019-pour-atteindre-les-9-md$_001151.

[27]. L'enseignement des systèmes numériques complexes Patrice Kadionik, Patrice Nouel, Ahmed Ben Attitalah, Philippe Dondon ENSEIRB-IXL BP 99 33402 TALENCE Cedex kadionik@enseirb.fr.

[28]. PSIM / TP6 .Etude de la structure de puissance d'un variateur pour moteur asynchrone

[29]. http://fr.farnell.com/terasic-technologies/p0037/carte-de-dev-de0-fpga-cyclone iii/dp/2217597.

[30]. Laboratoire 1 : Utilisation du logiciel Quartus II d'Altera.pdf.

[31]. Prototypage rapide a base de fpga d'un algorithme de controle avancé pour le moteur a induction par_boubacar housseini_l'université du québec à trois-rivières.

[32]. Denis Rabasté IUFM d'Aix-Marsielle _ stage de programmation des CPLD et FPGA en VHDL sous Max+plus II mars 2002.

[33]. https://www.google.fr/search?newwindow=1&q=Mémoires+mortes+et+logique+programmable.pdf&oq=Mémoires+mortes+et+logique+programmable.pdf.

[34]. Technologies de logique programmable_ http://jacques.weber.pagesperso orange.fr /telecharge /circ_prog.pdf.

[35]. http://fr.scribd.com/doc/192476088/Introduction-Circuits-Logiques-Programmables-presentation-2010.

[36]. http://lewebdephilou.free.fr/RESEAUX-TELECOM/Cours Telecom/Transmission /Electronique-Numerique_EISTI_Guy-Almouzni.pdf.

[37]. http://stigen.branly.amiens.free.fr/fiches%20elec/Lescircuitslogiques programmables.pdf.

[38]. http://www.xilinx.com/.

[39]. Magazine electronique N 7 juillet_Aut2010.
[40]. http://dumas.ccsd.cnrs.fr/dumas-00574220/document.
[41]. http://perso.univ-st- etienne.fr
[42]. www.techniques-ingenieur.fr/.
[43]. http://www.ni.com/white-paper/8043/fr/.
[44]. http://sti.discip.ac-caen.fr/sites/sti.discip.ac-caen.fr/IMG/pdf/plds.pdf.
[45]. Magazine electronique N 4.
[46]. http://www.yannthoma.com/research/these/These_Thoma_chap2.pdf.
[47]. http://www.altera.com/devices/fpga/cyclone2/cy2-index.jsp.
[48]. http://www.xilinx.com/products/silicon-devices/fpga/spartan-6.html.
[49]. http://www.techniques-ingenieur.fr/base-documentaire/energies-th4/convertisseurs electriques-et-applications-42253210/commande-numerique-a-base-de-composants-fpga-d-une-machine-synchrone-d2902/methodologie-de-developpement-d2902niv 10004.html.
[50]. http://www.schneider-electric.fr/documents/enseignement/intersection-guides/GT-1998-la-commande-de-moteur-asynchrone.pdf.
[51]. Michel Bensoam_régulation2000-2001.pdf.
[52]. http://biblioweb.u-cergy.fr/theses/07CERG0344.pdf.
[53]. http://depot-e.uqtr.ca/1885/1/030166253.pdf.
[54]. http://jacques.weber.pagespersoorange.fr/circuits_numeriques/6VHDL_rev_0703.pdf.
[55]. Cyclone II Device Handbook, Volume 1.pdf.
[56]. Introduction aux FPGA_Mickaël Dardaillon_M2RTS.pdf.
[57]. Simulation sur Matlab/Simulink et implémentation sur DSP/FPGA de la commande vectorielle de la machine synchrone à aimants permanents (PMSM) alimenté par un onduleur de tension a Modulation vectorielle (SVM).pdf.