الجـــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــعـليـم العـــــــالي و البـــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــامعة الدكتـــور الطاهـــــر مـــولاي سعيـــــــدة–

Université Saida Dr Tahar Moulay –
Faculté de TECHNOLOGIE

MEMOIRE

Projet de recherche présenté pour l'obtention du Diplôme de MASTER

En : Électronique

Spécialité : Instrumentation

Sujet

Génération de signaux vidéo VGA avec un FPGA pour les systèmes de vision artificielle

Présenté par : SIRAT Abdelhak & BAHLOULI Hichem

Soutenue publiquement, Juin 2024, devant le jury composé de :

Mr. BELLIL Ameur	MCB	Univ. Saida	Président
Melle. SEGHIER Salima	MCB	Univ. Saida	Encadrant
Melle. MAACHOU Fatima	MCB	Univ. Saida	Examinatrice

Année universitaire 2023/2024

Remerciements

Je remercie d'abord «Allah » qui m'a donné la force
et le courage pour terminer mes études et élaborer
ce modeste travail.

Ce travail n'aurait certainement jamais vu le jour sans l'aide, le
soutien et le dévouement de certaines personnes
que nous tienvivement à remercier.

Nous remercions notre encadreur Dr. SEGHIER Salima, qui m'a
fait bénéficier de son savoir, de ses compétences scientifiques
et de sa passiance pour la recherche.

Nos remerciements les plus vifs s'adressent aux membres du
jury d'avoir accepté d'examiner et d'évaluer notre travail.

Nos remerciements également à Mr A.ABDOUN l'ingénieur de
laboratoire de FPGA.

Un grand remerciement à nos parents qui ont sacrifié
pour nous, car grâce a leurs efforts nous somme là.

Nos remerciements les plus sincères à toutes
les personnes qui nous ont aidé dans ce mémoire même avec
une simple sourire.

Dédicaces

Je Dédié ce travail :

A ma très chère mère

A mon très cher père

Qui ont sacrifié leur vie pour

Ma réussite et m'ont éclairé le chemin par

Leurs conseils judicieux.

J'espère qu'un jour,

Je pourrai leurs rendre un peu de ce qu'ils

Ont fait pour moi, que dieu leurs prête bonheur et longue vie.

A mon grand-père et ma grand-mère

A mon encadreur Dr SGHIER SALIMA

A Mon binôme « BAHLOULI Hichem »avec qui j'ai passé des agréables moments.

A ceux qui me sont chers

A ceux qui ont toujours cru en moi

A ceux qui m'ont toujours encouragé

A ma famille et mes meilleurs amis qui par leurs prières et leurs encouragements, on a pu surmonter tout les obstacles.

A tous ceux qui m'aiment, et tous ceux que j'aime.

SIRAT ABDELHAK

Dédicaces

Je Dédié ce travail :
A ma très chère mère
A mon très cher père
Qui ont sacrifié leur vie pour
Ma réussite et m'ont éclairé le chemin par
Leurs conseils judicieux.
J'espère qu'un jour,
Je pourrai leurs rendre un peu de ce qu'ils
Ont fait pour moi, que dieu leurs prête bonheur et longue vie.
A mon grand-père et ma grand-mère
A mon encadreur Dr SGHEIR SALIMA
A Mon binôme « SIRAT ABDELHAK»avec qui j'ai passé des agréables moments.
A ceux qui me sont chers
A ceux qui ont toujours cru en moi
A ceux qui m'ont toujours encouragé
A ma famille et mes meilleurs amis qui par leurs prières et leurs encouragements, on a pu surmonter tout les obstacles.
A tous ceux qui m'aiment, et tous ceux que j'aime.
À ceux avec qui j'étais uni par des liens familiaux, à ceux avec qui je partageais les douceurs et les amers de la vie, « ma précieuse famille », de la petite à la grande.

À ceux dont vous avez renforcé le soutien à travers moi, mon frère et ma sœur

Aux mains cachées, à celles qui sont loin, aux plus proches en esprit de ceux qui ont été mon aide et mon soutien, que Dieu les bénisse et les protège... À toutes les connaissances qui m'ont encouragé, même avec le sourire.

Que Dieu vous récompense bien et vous récompense avec la meilleure récompense

BAHLOULI HICHEM

RESUME:

Les FPGAs (Field Programmable Gate Arrays) sont des circuits électroniques parallèles qui permettent aujourd'hui de développer des applications de plus en plus performantes en vitesse d'exécution et gourmandes en ressources matérielles. Dans notre projet, nous avons utilisé une caméra CMOS OV7670 pour capturer des images en temps réel et les afficher sur un écran VGA. Bien que notre projet n'implémente pas directement la vision artificielle, l'utilisation de la caméra et l'affichage des images en temps réel posent les bases pour une future intégration de systèmes de vision artificielle. En ajoutant des algorithmes de vision par ordinateur, notre système pourrait être amélioré pour non seulement capturer et afficher des images, mais aussi d'analyser et interpréter les scènes visuelles pour identifier des objets ou des situations spécifiques, augmentant ainsi les capacités de surveillance et d'alerte. Afin de réaliser la conception et l'implémentation de quelques fonctions d'affichages VGA, on a utilisé le langage de description matériel VHDL après avoir expliqué ses notions et ses fonctionnalités. Après, avoir implanté les programmes des différentes fonctions des signaux vidéo VGA sur FPGA dans l'environnement Quartus II, qui est un logiciel développé par la société ALTERA permettant la gestion complète d'un flot de conception, on a finalement généré nos signaux de caractères et des images sur un écran vidéo VGA.

MOTS CLES:

FPGA, VHDL, implémentation, RVB, pixels, VGA, génération des signaux, Cyclone IV, vision artificielle, logiciel Qartus II.

Abstract:

FPGAs (Field Programmable Gate Arrays) are parallel electronic circuits which today make it possible to develop applications that are increasingly efficient in execution speed and demanding in hardware resources. In our project, we used an OV7670 CMOS camera to capture real-time images and display them on a VGA screen. Although our project does not directly implement artificial vision, the use of the camera and the display of images in real time lays the foundation for future integration of artificial vision systems. By adding computer vision algorithms, our system could be enhanced to not only capture and display images, but also analyze and interpret visual scenes to identify specific objects or situations, thereby increasing surveillance and monitoring capabilities. 'alert. In order to design and implement some VGA display functions, we used the VHDL hardware description language after explaining its concepts and functionalities. After having implemented the programs for the different functions of the VGA video signals on FPGA in the Quartus II environment, which is software developed by the company ALTERA allowing the complete management of a design flow, we finally generated our character signals and images on a VGA video display.

Keywords:

FPGA, VHDL, implementation, RGB, pixels, VGA, signal generation, Cyclone IV, artificial vision, Qartus II software.

ملخص:

FPGA (مصفوفات البوابات القابلة للبرمجة) عبارة عن دوائر إلكترونية متوازية تتيح اليوم إمكانية تطوير تطبيقات ذات كفاءة متزايدة في سرعة التنفيذ وتتطلب موارد الأجهزة. في مشروعنا، استخدمنا كاميرا OV7670 CMOS لالتقاط صور في الوقت الفعلي وعرضها على شاشة VGA. على الرغم من أن مشروعنا لا ينفذ الرؤية الآلية بشكل مباشر، إلا أن استخدام الكاميرا وعرض الصور في الوقت الفعلي يضع الأساس للتكامل المستقبلي لأنظمة الرؤية الآلية. ومن خلال إضافة خوارزميات رؤية الكمبيوتر، يمكن تحسين نظامنا ليس فقط لالتقاط الصور وعرضها، ولكن أيضًا لتحليل وتفسير المشاهد المرئية لتحديد أشياء أو مواقف معينة، وبالتالي زيادة قدرات المراقبة والمراقبة. من أجل تصميم وتنفيذ بعض وظائف عرض VGA، استخدمنا لغة وصف الأجهزة VHDL بعد شرح مفاهيمها ووظائفها. بعد تنفيذ البرامج الخاصة بالوظائف المختلفة لإشارات فيديو VGA على FPGA في بيئة Quartus II، وهو برنامج تم تطويره بواسطة شركة ALTERA يسمح بالإدارة الكاملة لتدفق التصميم، قمنا أخيرًا بإنشاء إشارات وصور شخصيتنا على VGA عرض الفيديو.

الكلمات الدالة:

FPGA، VHDL، التنفيذ، RGB، البكسلات، VGA، توليد الإشارة، Cyclone IV، الرؤية الاصطناعية، برنامج Qartus II.

SOMMAIRE

Chapitre II : Présentation de VHDL, signal vidéo VGA et la vision artificielle

Chapitre III : Commande d'un écran VGA par un circuit FPGA Cyclone IV

LISTE DES FIGURES

LISTE DES TABLAUX

LISTE DES ABREVIATIONS

A

ASM: Algorithmic **S**equential **M**achines
ASIC: Application **S**pecific **I**ntegrated **C**ircuit.

C

CAO: Conception **A**ssistée par **O**rdinateur.

CLB: Configurable **L**ogic **B**lock.

CPLD : Complex **P**rogrammable **L**ogic **D**evice.

CMOS: Complementary **M**etal–**O**xide–**S**emiconductor.

D

DC *: Direct* **C**urrent.

DSP: Digital **S**ignal **P**rocessor.

DoD: Department **o**f **D**efense **(**United States**).**

F

FPGA: Field **P**rogrammable **G**ate **A**rrays.

FIFO : First **I**n, **F**irst **O**ut

H

HDL : Hardware **D**escription **L**anguage (langage de description matériel).

HSYNCH : **S**ynchronisation **H**orizontale

I

IEEE: Institut of **E**lectrical and **E**lectronics **E**ngineers.

IOB: Input **O**utput **B**loc.

IEO: **E**lement **D**'entrée/**S**ortie

J

JTAG: Joint **T**est **A**ction **G**roup

L

LAB: Logic **A**rray **B**loc.
LE : Elément **L**ogique**.**
LSI : Large **S**cale **I**ntegration.
LUT : Look **u**p **T**able.
LCD : Liquid **C**rystal **D**isplay
LED : **D**iode **E**lectroluminescente
LC : **C**ellule **L**ogique

P

PAL: Programmable Array Logic

PLD: Programmable Logic Device

PLL : **P**hase-**L**ocked **L**oop

M

MIF : Memory **I**nitialization **F**ile

R

RAM: Random **A**ccess **M**emoy.

ROM: Read-**O**nly **M**emory

RGB : **R**ouge **V**ert **B**leu

RTL: Register **T**ransfer **L**evel.

S

SRAM: Static **R**andom **A**ccess **M**emory.

T

TTL : **T**ransistor-**T**ransistor **L**ogic

V

VHDL: VHSIC **H**ardware **D**escription **L**anguage.

VHSIC: Very **H**igh **S**peed **I**ntegrated **C**ircuit.

VGA : Video **G**raphics **A**dapter

VLSI: Very **L**arge **S**cale **I**ntegration.

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Depuis la commercialisation du premier circuit programmable FPGA en 1985, l'utilisation de ces circuits ne cesse de s'étendre à des domaines et applications variés, parmi lesquels nous pouvons citer le traitement d'image et de vidéos [1], [2], les réseaux de neurones [3], la comparaison de séquences génétiques [4], l'architecture des ordinateurs [5] etc... L'intérêt suscité par les FPGA est dû essentiellement à leurs prix abordables, facilité de mise en œuvre et flexibilité [6].

Dans notre vie quotidienne, l'image joue un rôle de plus en plus important pour nous informer ou nous divertir. En parallèle, le traitement de l'information s'est lui aussi développé grâce à l'évolution de la microélectronique en proposant des systèmes de plus en plus performants pour exécuter des algorithmes très complexes. Les progrès dans la capacité d'intégration des circuits électroniques ont ouvert de nouvelles perspectives pour le traitement d'images en temps réel sur des systèmes embarqués. D'un côté, des processeurs spécifiques peuvent couramment effectuer des milliards d'opérations par seconde et d'un autre côté, des composants reprogrammables comporteront dans un avenir proche plusieurs milliards de portes logiques. Ces circuits permettent de réaliser des applications avec des performances en termes de vitesse de traitement sans cesse croissantes

Pour réaliser une application avec un FPGA il faut décrire le circuit électronique à réaliser avec un langage de description matérielle comme le VHDL. 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.

L’étude que nous présentons dans ce mémoire vise la conception, la génération et l’implémentation des descriptions textuelles (VHDL) sur un FPGA de la famille Cyclone IV de type EP4CE10E22C8 dont les sorties sont directement reliées à un connecteur pour écran VGA par le logiciel Quartus II afin de générer des images et des caractères sur un écran vidéo VGA.

L'écran du moniteur pour un format VGA standard contient 640 colonnes par 480 lignes d'éléments d'image appelé pixel. Une image est affichée sur l'écran en allumant et en éteignant individuellement les pixels. La combinaison de nombreux pixels génère une image.

L’objectif de notre travail est de construire une plate-forme d'imagerie FPGA en utilisant un langage de description matériel le VHDL. Le principe est d'interfacer une caméra CMOS de résolution VGA avec la carte de développement OMDAZZ et de transmettre le flux vidéo en direct acquis sur un moniteur VGA. Les images de la caméra sont diffusées sur un moniteur VGA à 30 images par seconde. Nos images sont affichées sur 160x120 pixels puisque le FPGA cyclone IV quand on a utilisé n'a pas assez de mémoire pour stocker l’image VGA complète.

Ce mémoire se divise en trois chapitres organisés de la manière qui suit:

Dans le premier chapitre nous nous intéressons aux circuits logiques programmables et en particulier les FPGA et la présentation de notre carte FPGA.

Le deuxième chapitre, traite l’outil indispensable pour la programmation d’un FPGA qui est le langage VHDL et les fonctionnalités de base de celui-ci lors des phases de conception ou de synthèse, ensuite nous donnons une généralité sur l’image, le signal vidéo VGA et comment générer l'affichage vidéo VGA à l'aide d’un FPGA, une présentation de la vision artificielle et le montage de la caméra et le module VGA avec la carte FPGA Cyclone IV.

Le troisième chapitre a débuté par une description détaillée sur la méthode de conception et d’implémentation des circuits dans un FPGA on utilisant le logiciel Quartus II, en finira par la présentation de nos programmes VHDL (les schémas

fonctionnels RTL) et les résultats de génération de l'affichage des écrans vidéo des caractères et des images captées par une caméra CMOS OV7670.

Dans un premier temps, nous allons afficher le message "WELCOME JURY" sur l'écran pour l'accueil. Ensuite, nous avons utilisé un capteur ultrasonique pour mesurer la distance et afficher la valeur mesurée sur un afficheur à sept segments. Lorsque quelqu'un entre dans une zone de 30 cm, un buzzer s'active pour alerter de sa présence.

Ensuite, nous allons intégrer une caméra pour capturer une vidéo en temps réel et l'afficher sur un écran VGA avec une résolution de 160x120. Enfin, nous avons combiné les fonctionnalités de la caméra et du capteur ultrasonique pour activer le buzzer lorsqu'une personne entre dans le champ de détection, sans afficher la distance mesurée sur l'afficheur à sept segments.

Enfin, nous terminons notre travail avec une conclusion générale dressera le bilan de cette étude.

REFERENCES BIBLIOGRAPHIQUES -INTRODUCTION-

[1] S. C. Chan, H.O. Ngai and K.L. Ho, "A programmable image processing system using FPGA" International journal of electronics, vol 75, N°4 pp 725-730, 1993.
[2] M. Alves de barros, "Traitement bas niveau d'images en temps réel et circuits reconfigurables" Thèse de doctorat, Université de Paris-Sud, 1994.
[3] J.G. Eldrerge, B.L. Hutchings, "Density enhancement of neural network using FPGAs and run-time reconfiguration" FCCM, 1994.
[4] E. Lemoine, J. Quinqueton and J. Salantin, "High speed pattern matching in genetic data base with reconfigurable hardware" Proceeding of th 2nd INT. Conf. o, Intelligent systems for molecular biology, pp 269-276. AAAI, 1994.
[5] B.Heeb and C. Pfister, "Chameleon, a workstation of a different colour" 2nd International Workshop on Field-Programmable Logic Applications, paper 5.6, Vienna, Austria, 1992.
[6] H.Guermoud, Y.Berviller, E.Tisserand, S.Weber, « Architecture à base de FPGA reconfigurable dynamiquement dédiée au traitement d'image sur flot de données », SEIZIÈME COLLOQUE GRETSI — 15-19 SEPTEMBRE 1997 — GRENOBLE

Chapitre I

Circuits logiques programmables FPGA

I.1 Introduction

FPGA est l'abréviation de Field Programmable Gate Arrays ou "réseaux logiques programmables". Inventés par la société Xilinx en 1985, les FPGA sont des composants logiques de haute densité et reconfigurables qui permettent, après programmation, de réaliser des fonctions logiques, des calculs, et des générations de signaux. Il s'agit d'un circuit intégré qui peut être programmé pour fonctionner selon la conception prévue. Cela signifie qu'il peut fonctionner comme un microprocesseur, ou comme une unité de cryptage, ou une carte graphique, ou même tous ces trois à la fois.

L'avantage de ce genre de circuit est sa grande souplesse qui permet de les réutiliser à volonté dans des algorithmes différents en un temps très court. Les FPGAs peuvent être utilisés pour implémenter n'importe quelle fonction logique que les Circuits intégrés spécifiques ASICs (Application Specific Integrated Circuit) peuvent implémenter. Leur reconfiguration, qui peut être effectuée un nombre arbitraire de fois, représente l'un de leurs avantages majeurs par rapport aux ASICs [I.1], [I.2], [I.3].

Les conceptions fonctionnant sur des FPGA sont généralement créées à l'aide de langages de description de matériel tels que VHDL et Verilog.

Le présent chapitre décrit les FPGAs ainsi que les principaux composants et éléments qui les caractérisent, leurs avantages et différents domaines d'applications. Pour terminer le chapitre, nous décrivons brièvement une rapide présentation de notre carte de développement d'Altera qui contient un circuit FPGA de la famille Cyclone IV.

I.2 Circuits Logiques Programmables du type FPGA

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 [I.4], [I.5], [I.6].

Les FPGAs, sont des composants électroniques programmables de la famille des PLDs (Programmable Logic Device). Un FPGA est un ensemble de blocs logiques élémentaires que l'utilisateur peut interconnecter pour réaliser les fonctions logiques de son choix. La densité des portes est importante et sans cesse en évolution. L'avantage d'un FPGA est leur grande souplesse dans leur technologie permettant une réutilisation à volonté et en un temps très court (quelques millisecondes) dans des algorithmes différents. Le progrès technologique permet de faire des composants toujours plus rapides et à plus haute intégration, autorisant la programmation d'applications importantes [I.7].

Grâce à l'évolution des procédés de fabrication, ces composants peuvent actuellement supporter des applications complexes. Ils sont constitués d'un réseau de blocs logiques, de blocs mémoires, de blocs dédiés et d'entrées/sorties. L'ensemble est relié par un réseau d'interconnexions programmable. Les blocs logiques permettent de réaliser des opérations avec quelques variables à travers une LUT (Look Up Table) [I.8]. Le résultat peut être éventuellement stocké dans un registre, les blocs RAM permettent d'implanter des mémoires adressables et des FIFO (First In, First Out), les blocs dédiés permettent de réaliser facilement de nombreuses opérations de traitement (blocs DSP), de gérer l'horloge, ou des interfaces de communication (Rocket IO, Ethernet, PCI Express). Les nombreux ports permettent également de connecter des périphériques de la plateforme matérielle à base de FPGA.

Les FPGA se programment grâce à leurs LUT et leur réseau d'interconnexion. La programmation se fait avec un langage de programmation hardware tel que le VHDL ou bien le Verilog. L'outil de développement transforme cette description en un fichier de configuration du FPGA en plusieurs étapes : le HDL (Hardware Design Language) doit d'abord être synthétisé (transformé en éléments logiques de base), puis les éléments doivent être placés sur le composant (placement) et enfin interconnectés (routage) [I.9].

La figure I.1 montre la structure interne d'un FPGA de type matrice symétrique. Il s'agit de l'architecture que l'on retrouve dans les FPGA actuels.

L'utilisateur peut programmer la fonction réalisée par chaque cellule (CLB: Configurable Logic Block). On programme aussi les interconnexions entre les cellules. .

Les FPGA les plus récents sont configurables en une centaine de millisecondes. Les FPGA sont utilisés pour un développement rapide et bon marché des ASIC [I.10].

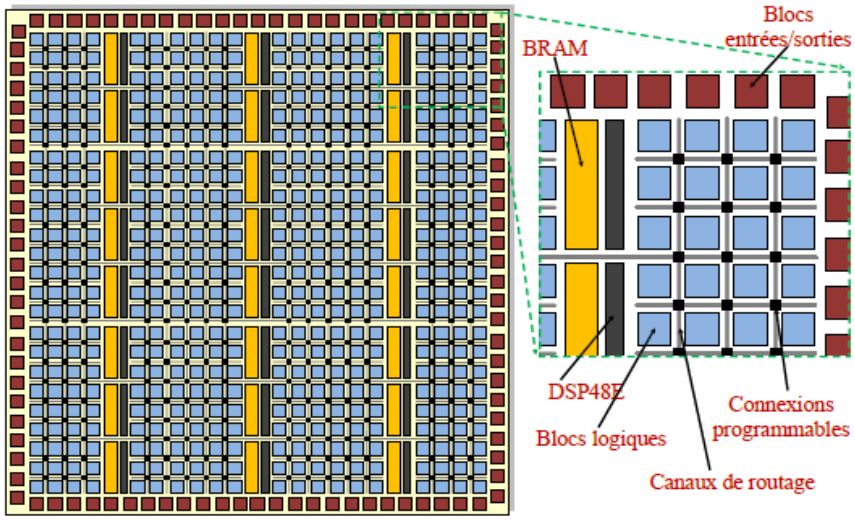


Figure I.1 : Architecture conceptuelle d'un FPGA (Architecture, Interconnexions et CLB).

I.2.1 Architecture [I.10]

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

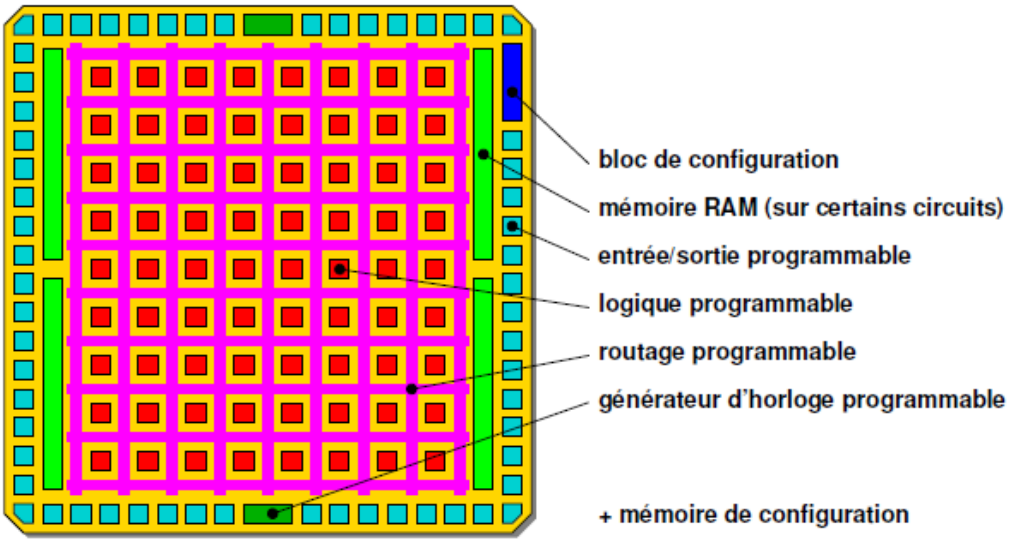


Figure I.2 : Architecture interne des FPGAs

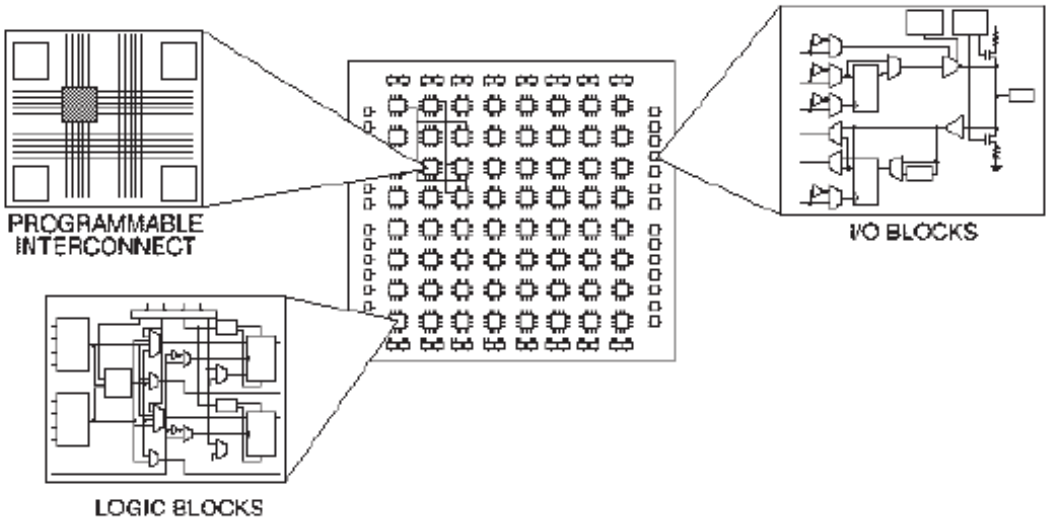


Figure I.3 : Différents secteurs d'un FPGA

La structure du FPGA présentée figure I.2 est composée :

- **De cellules d'entrées sorties modifiables** qui servent d'interfaces entre les broches du circuit et le cœur du FPGA pour adapter les signaux suivants :
  - Alimentation
  - Signaux d'horloge
  - Signaux de configuration du FPGA

- Signaux de test

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

- La partie combinatoire permet de réaliser des fonctions de complexité moyenne avec des portes classiques ET, OU et NON de deux à une dizaine d'entrées.
- La partie séquentielle comporte une ou deux bascules généralement de type D.

Compte tenu du nombre d'éléments logiques et de leur structure, leur association permet de réaliser tous les types de bascule. L'intérêt est de créer des mémoires élémentaires à un bit.

Suivant le fabricant du circuit, ces blocs contiennent un nombre différent de portes logiques et de bascules à l'intérieur d'un bloc comme le montre la figure II.4.

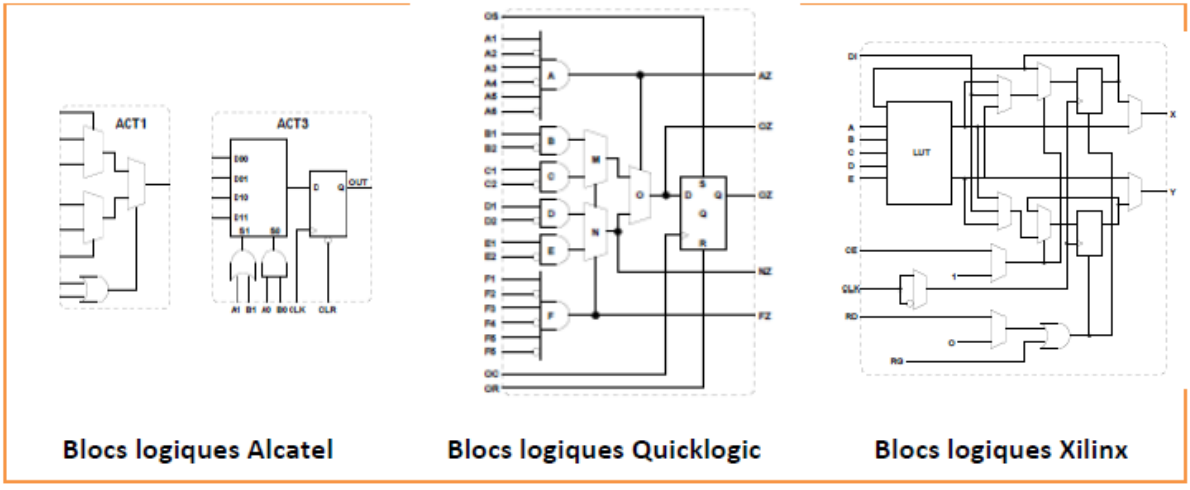


Figure I.4 : Exemple de blocs logiques de différents fabricants

Il existe 4 types de blocs logiques :

Les macro-cellules : Ces cellules logique sont appelé aussi par :

- ✓ Soit CLB (configurable logique bloc), dénomination adoptée par XILINX.
- ✓ Soit LC (cellule logique), le nom choisi par CYPRESS.
- ✓ Soit LE (élément logique), l'appellation d'ALTERA.

- ❖ **De réseaux d'interconnexions** que l'on voit en figure II.5. Ces réseaux relient entre eux les blocs logiques et les blocs d'entrées/sorties. Ces connections peuvent directement relier :
- ▪ **Des éléments internes** dans un bloc grâce à un système de tables logiques appelées LUT. C'est une matrice de connections où les points de routage déterminent le niveau des entrées soit haut soit bas des portes logiques.
- ▪ **Des éléments proches** : on parle de liaisons directes entre les blocs.
- ▪ **Plusieurs blocs présents sur toute la surface** : on parle de liaisons à distance ou générales.

Certains de ses canaux sont spécifiques aux signaux d'horloges.

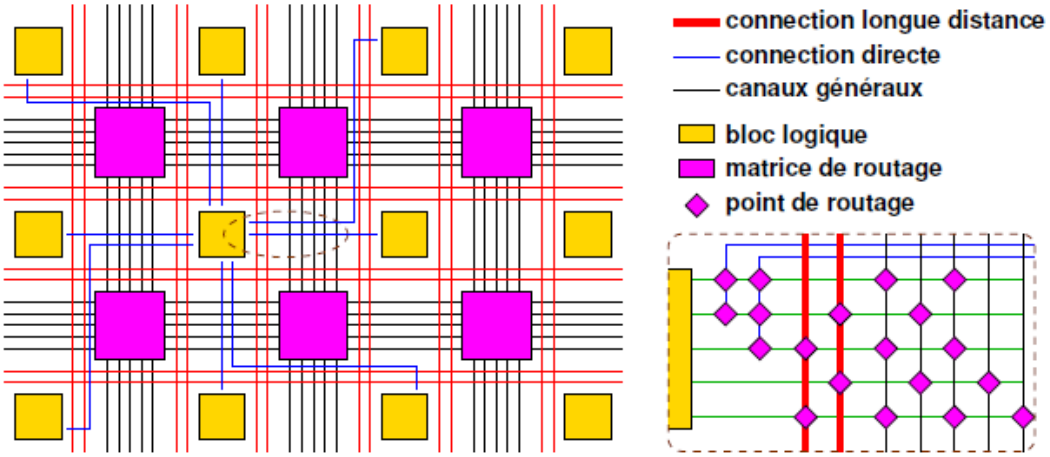


Figure I.5 : Structure générale du routage

Remarque : L'échelle est fausse (blocs logiques < 5% du circuit).

Un microprocesseur : La présence d'un processeur est indispensable pour ordonnancer les commandes reçues par le FPGA. C'est le chef d'orchestre de tout système informatique, où son rôle est de suivre des instructions qui lui ont été préalablement programmées en langage C.

Habituellement, ce processeur se trouve à l'extérieur du FPGA, mais des constructeurs ont intégré ces systèmes directement dans le FPGA. Il s'agit de

processeur « soft-core » (soft pour logiciel et core pour cœur d'exécution) on parle aussi de système sur puce programmable (SOPC) (System On Programmable Chip). Il communique avec le FPGA grâce au langage de description matérielle VHDL. Ce processeur est donc reconfigurable pouvant ainsi d'adapter aux contraintes de chaque utilisation.

ASICs

Dans la littérature, le terme ASIC (Application Specific Integrated Circuit) est employé pour décrire l'ensemble des circuits spécifiques à une application. Or, dans le langage courant, le terme ASIC est presque toujours utilisé pour décrire les circuits réalisés chez un fondeur. On désigne, par le terme générique PLD (Programmable logic Device), l'ensemble des circuits programmables par l'utilisateur.

I.2 .2 FPGAs : illustration avec la famille Cyclone IV d'Altera

S'appuyant sur le succès des FPGA Cyclone et pour accroître son leadership sur les interfaces haut débit, Altera Corporation présente sa nouvelle gamme de FPGA Cyclone IV. Afin de répondre à l'augmentation des besoins en bande passante au moindre coût provoquée par la demande en vidéo mobile, en voix, en accès aux données et en images 3D de qualité, la nouvelle gamme de FPGA Cyclone IV ajoute la prise en charge des protocoles série standards et offre le meilleur compromis en coût, puissance et en fonctionnalités logiques, mémoire et DSP [I.11].

Figure I.6 : Symbole de Cyclone IV d'Altera.

La gamme de FPGA Cyclone IV offre deux variantes. Les circuits Cyclone IV GX possèdent jusqu'à 150K éléments logiques, 6.5Mbits de RAM, 360 multiplieurs et 8 interfaces haut débit. Avec une basse consommation et un boitier de 11x11 mm seulement, ces circuits conviennent aux applications à taille réduite et à faible coût des marchés sans fil, filaire, télédiffusion, industriel et grand public. Les circuits Cyclone IV E associent faible coût et hautes fonctionnalités et diminuent d'au moins de 25% la consommation par rapport à la génération précédente de FPGA Cyclone dans les applications basse consommation comme la radio logicielle portable [I.11].

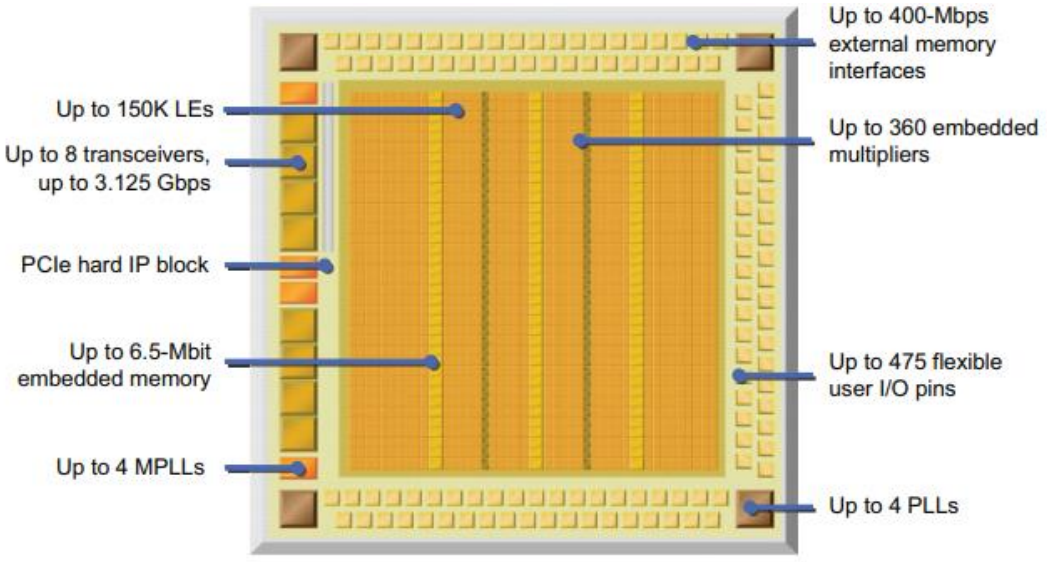


Figure I.7 : Caractéristiques architecturales clés du FPGA Cyclone IV [I.12].

L'architecture de Cyclone IV E comprend jusqu'à 115 K éléments logiques disposés verticalement (LE).

❖ **Elément logique (LE) Cyclone IV**

C'est le plus petit élément de logique dans le Cyclone IV. Sa structure apparaît sur la figure ci-dessous :

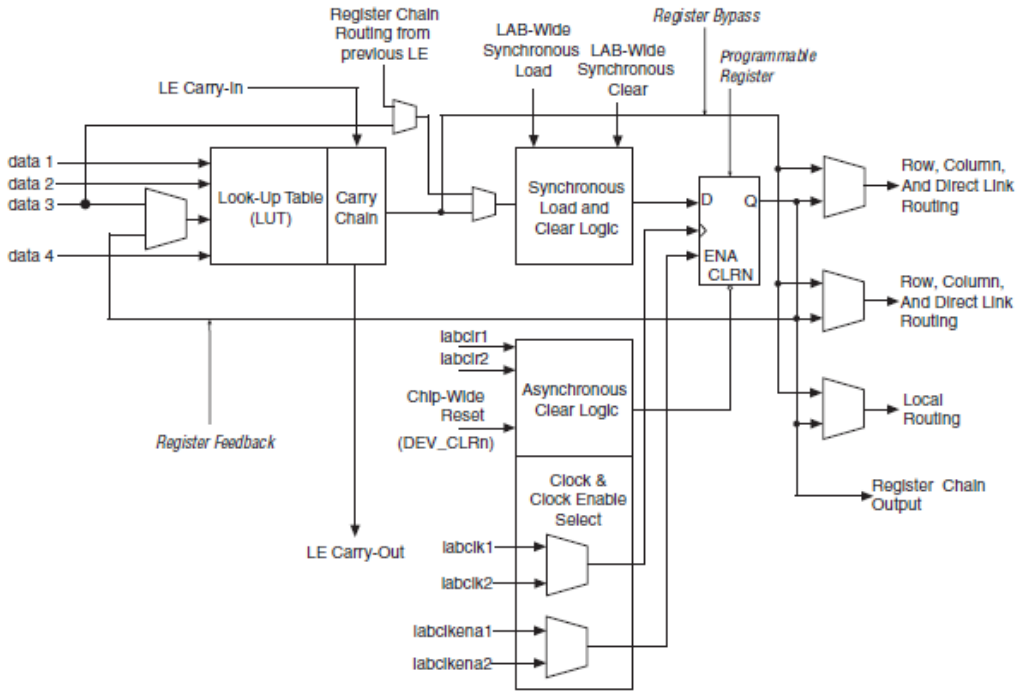


Figure I.8 : Structure d'un Logic Element LE [I.13].

Cette cellule reconfigurable contient divers éléments :

- Une LUT (Look up table) à quatre entrées, permettant le calcul de n'importe quelle fonction de 4 entrées.
- Un registre de sortie programmable
- Une connexion de chaîne de transport
- Une connexion de chaîne de registre
- La possibilité de piloter les interconnexions suivantes :
  - ✓ Locale
  - ✓ Ligne
  - ✓ Colonne
  - ✓ Chaîne d'enregistrement
  - ✓ Lien direct
- Enregistrez le support d'emballage
- Enregistrez le support de rétroaction

❖ **Blocs de réseaux logiques (LAB)**

Un bloc de réseaux logiques contient :

- 16 Eléments Logiques (LE).
- Un réseau d'interconnexion local pour la communication entre LE du même LAB (voir figure I.9).
- Un accès direct aux éléments adjacents du LAB dans la structure du FPGA (voir figure I.9) tels que :
  - ✓ un autre LAB.
  - ✓ un bloc mémoire pour les LAB adjacents aux zones mémoires.
  - ✓ un signal d'horloge d'une PLL.
  - ✓ un multiplexeur.
  - ✓ un IOE (Elément d'entrée/sortie).
  - ✓ un accès aux réseaux d'interconnexions lignes/colonnes pour atteindre n'importe quel point du composant.

L'interconnexion locale LAB est pilotée par des interconnexions de colonne et de ligne et des sorties LE dans le même LAB. Les LAB voisins, les boucles à verrouillage de phase (PLL), les blocs de RAM M9K et les multiplicateurs intégrés de gauche et de droite peuvent également piloter l'interconnexion locale d'un LAB via la connexion de liaison directe. La fonction de connexion directe minimise l'utilisation d'interconnexions de lignes et de colonnes, offrant des performances et une flexibilité supérieures. Chaque LE peut piloter jusqu'à 48 LE via des interconnexions de liaison locales et directes rapides.

Le compilateur Quartus II place la logique associée dans un LAB ou des LAB adjacents, permettant l'utilisation de connexions de chaînes locales et de registres pour les performances et l'efficacité de la zone.

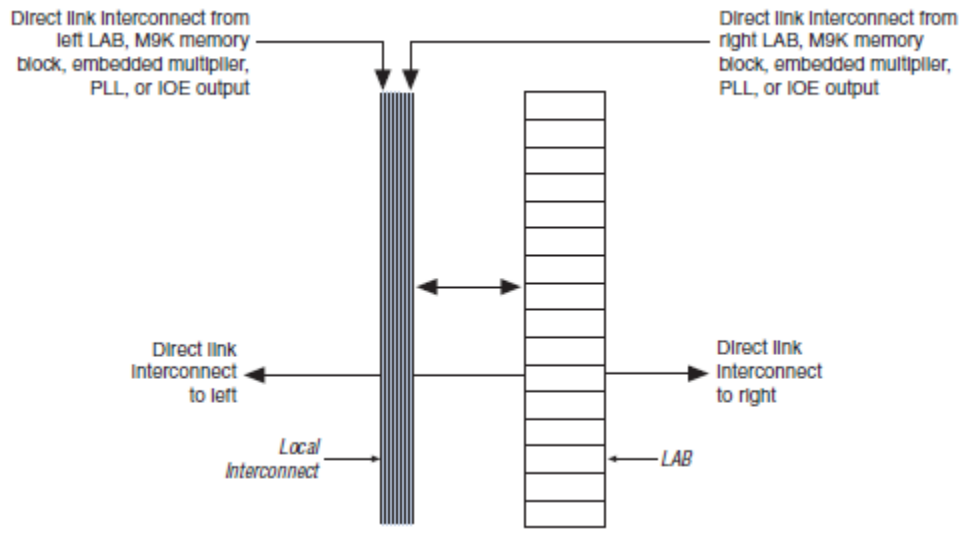


Figure I.9 : Connexions chaînées entre LEs dans un LAB de circuit Cyclone IV [I.13].

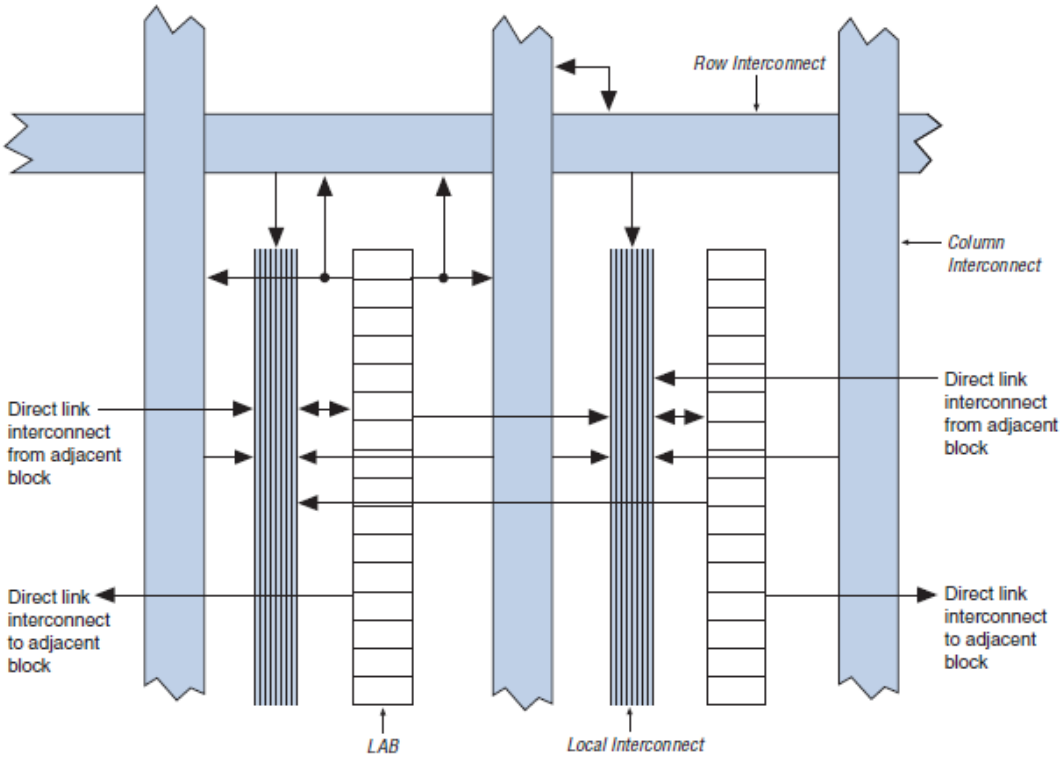


Figure I.10 : Structure LAB de circuit Cyclone IV [I.13].

❖ **Bloc d'entrée/sortie (IOE : Input Output Element)**

Avec l'accroissement constant du nombre de standards d'entrées/sorties en électronique numérique, la conception d'un FPGA a progressivement nécessité de faire apparaître des blocs dédiés capables d'interfaçages s'adaptant à une grande diversité de situation.

La frange supérieure des composants de la famille Cyclone II dispose donc de nombreux blocs d'entrées/sorties (I/O Banks) répartis à la périphérie du composant.

La famille Cyclone IV peut s'interfacer avec des circuits logiques :

- LVTTL et LVCMOS : Interfaçage avec des circuits logiques d'usage général, fonctionnant à des fréquences moins de 100MHz.
- SSTL : Standard mis en place pour la mémoire SDRAM DDR (Double Data Rate).
- HSTL : Signaux des mémoires QDR2 SRAM (Quad Data Rate).
- LVDS (Low Voltage Differential Signaling) : Signaux différentiels (ils garantissent une plus grande immunité au bruit) pour des communications à fort débit (jusqu'à 805Mbps) et faible EMI (émissions électromagnétiques)
- LVPECL (Low Voltage Positive Emitter Coupled Logic) : Signaux différentiels à haute immunité au bruit utilisé en vidéo, télécom, distribution d'horloge.
- PCI et PCI Express Bus locaux des PC utilisés pour la connexion de car te d'extension (Vidéo . . .).

❖ **Blocs de mémoire dans les appareils Cyclone IV**

Les dispositifs Cyclone IV présentent des structures de mémoire intégrées pour répondre aux besoins de mémoire sur puce des conceptions de dispositifs Altera Cyclone IV. La structure de mémoire intégrée se compose de colonnes de blocs de mémoire M9K que vous pouvez configurer pour fournir diverses fonctions de mémoire, telles que la RAM, les registres à décalage, la ROM et les buffers FIFO.

Les blocs M9K prennent en charge les fonctionnalités suivantes :

■ 8 192 bits de mémoire par bloc (9 216 bits par bloc, parité comprise)

■ Signaux d'activation de lecture (rden) et d'activation d'écriture (wren) indépendants pour chaque port

■ Mode compact dans lequel le bloc de mémoire M9K est divisé en deux RAM à port unique de 4,5 Ko

■ Configurations de ports variables

■ Prise en charge des modes simple port et double port pour toutes les largeurs de port

■ Véritable fonctionnement à deux ports (une lecture et une écriture, deux lectures ou deux écritures)

■ L'octet active le masquage d'entrée de données pendant les écritures

■ Deux signaux de contrôle d'activation d'horloge pour chaque port (port A et port B)

■ Fichier d'initialisation pour pré-charger le contenu de la mémoire en modes RAM et ROM

PLL: Les boucles à verrouillage de phase (PLL : Phase Locked Loops) d'un Cyclone II fournit une capacité de synthétiser une horloge principale qui permet la génération de plusieurs horloges internes qui fonctionnent à différentes fréquences issue de l'horloge d'entrée. Chaque PLL peut fournir jusqu'à trois sorties d'horloge pouvant fonctionner à des fréquences différentes. Le PLL est un système qui permet de synchroniser la phase instantanée de deux signaux.

I.2 .2.1 Puissance de circuit Cyclone IV

Comme le montre la figure I.11, Altera a travaillé avec son partenaire de fabrication de longue date TSMC pour optimiser le processus de fabrication afin de produire des FPGA avec une puissance statique et dynamique inférieure, jusqu'à 25 % et 30 % de puissance totale en moins respectivement par rapport aux familles Cyclone précédentes [I.14].

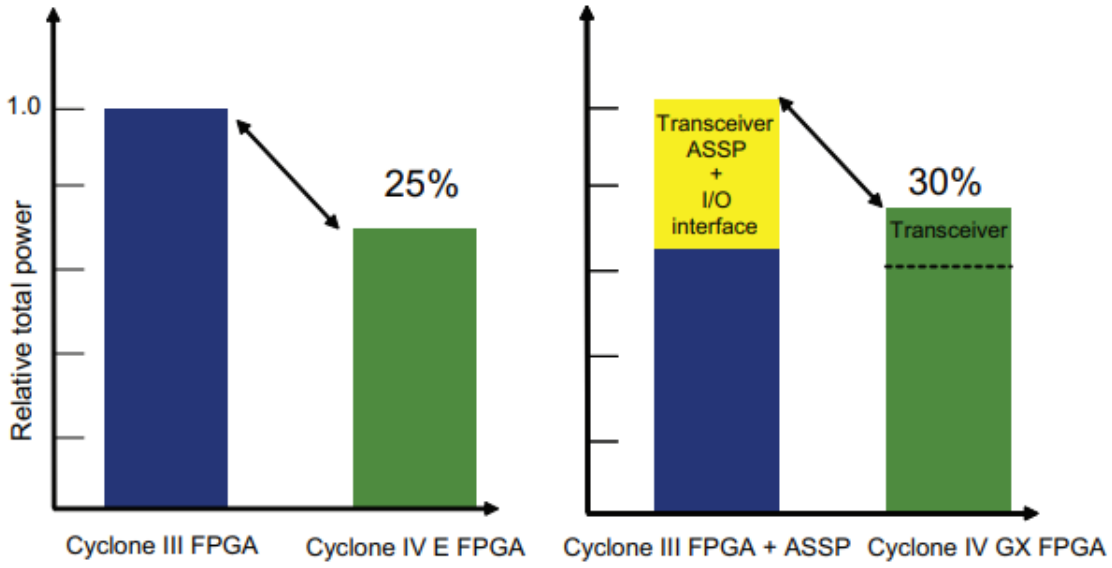


Figure I.11 : Puissance Cyclone IV E (à gauche) et Cyclone IV GX (à droite) [I.14].

Les FPGA sont synonymes de commercialisation rapide des produits en raccourcissant le calendrier de développement des produits.

Les ingénieurs qui choisissent les FPGA Cyclone IV, avec des performances de cœur jusqu'à 25 % plus rapides (par rapport aux FPGA à faible coût des concurrents), consacreront moins d'itérations de conception à la fermeture temporelle, car une plus grande marge temporelle est disponible. De plus, le logiciel de conception Quartus II d'Altera a des temps de compilation jusqu'à 50 % plus rapides par rapport aux produits logiciels concurrents, ce qui rend les ingénieurs plus productifs, chaque jour au bureau [I.14].

I.2 .2.2 Programmation d'un périphérique de configuration série

Le dispositif Cyclone IV E prend en charge la programmation dans le système d'un dispositif de configuration série à l'aide de l'interface JTAG via la conception du chargeur flash série. Le chargeur flash série est une conception de pont pour l'appareil Cyclone IV E qui utilise son interface JTAG pour accéder au fichier EPCS .jic, puis utilise l'interface AS pour programmer l'appareil EPCS. La figure I.12 illustre la méthode de programmation lors de l'adoption d'une solution de chargeur flash série [I.15].

Figure I.12 : Programmation d'un périphérique de configuration série avec une solution de chargeur flash série [I.15].

I.2 .2.3 Caractéristiques de Cyclone IV EP4CE10 FPGA [I.16]

- ✓ éléments logiques (EL) 10000
- ✓ Boucles de structure et d'E/S à phase asservie (PLL) 2
- ✓ Mémoire embarquée maximale 414 Kb
- ✓ Blocs DSP (Digital Signal Processing) 23
- ✓ Format DSP (Digital Signal Processing) Multiply
- ✓ Contrôleurs de mémoire matériels Non
- ✓ Support mémoire externe (EMIF) DDR, DDR2, SDR
- ✓ Configuration E/S Nombre maximal d'utilisateurs des E/S 179

I.3 Avantages du FPGA

Ces notions sur le FPGA donnent des indications quant à l'intérêt de son utilisation dans le cadre du projet. Voici les avantages clés qui ont fait que le laboratoire s'est tourné vers cette solution [I.17].

- ✓ **Un circuit reprogrammable** : L'avantage du FPGA est de pouvoir être reprogrammable contrairement aux circuits intégrés de type ASIC. Ce qui rend cette solution modulable et donne la possibilité de modifier le programme générique de base afin de le rendre spécifique au circuit utilisé. Une solution de validation utilisant le FPGA peut alors convenir à beaucoup de projets et donc diffusée à plusieurs équipes.

- ✓ **Un investissement rentable** dans la durée : Cela est dû à sa reprogrammation, ce qui implique une réutilisation à destination d'autres projets, malgré un prix à l'achat supérieur à un circuit ASIC.
- ✓ **Une Reprogrammation quasi-instantanée** du circuit. Une fois le programme validé cela ne prend que quelques minutes à l'implémenter. A titre de comparaison, la fabrication d'un circuit ASIC peut prendre plusieurs semaines.

Cependant, le FPGA n'est pas le seul composant reprogrammable du marché. Le DSP « Digital Signal Processor », processeur de signal numérique, permet également d'émuler un montage numérique. Le DSP est programmable grâce au langage C, le FPGA utilise quant à lui le VHDL. Le format de description machine est généré automatiquement par les logiciels de développement des concepteurs. Il est possible alors de vérifier les circuits sans avoir à les concevoir par nous-mêmes. L'importation de leur fichier dans notre programme est suffisante. Par ailleurs, le FPGA peut disposer d'un DSP sous forme d'IP incluse dans le système du circuit [I.10].

I.4 Critères de choix du circuit programmable FPGA [I.18], [I.19]

Les FPGA sont développés récemment grâce aux progrès de la technologie VLSI (Very Large Scale Integration), l'apparition de ce type de circuits est une révolution des systèmes digitaux et ouvrants des perspectives de traitement numérique inaccessibles auparavant. La fin des années 80 a vu l'apparition des premiers circuits FPGA qui sont des circuits intégrés que l'on peut configurer en un temps relativement court pour réaliser n'importe quelle fonction logique « câblée » à bas coût par une programmation de ses cellules logiques et ses interconnexions avec une restriction de ne pas épuiser les ressources du FPGA. Typiquement, un circuit FPGA haute densité peut contenir jusqu'à plusieurs millions d'éléments programmables. Pour réussir une application à base de FPGA et afin d'obtenir un système plus performant, consommant un minimum de puissance, il est nécessaire de respecter un certain nombre de règles comme :

- Bien connaître les caractéristiques du FPGA ciblé pour assurer son adéquation avec les besoins du projet.
- Elaborer une méthodologie de conception.
- Maîtriser les outils d'implémentation et de choisir des outils de synthèse de qualité.

La conception sur les circuits FPGA est un challenge dans lequel l'objectif est de trouver le bon compromis entre densité, flexibilité et performances temporelles.

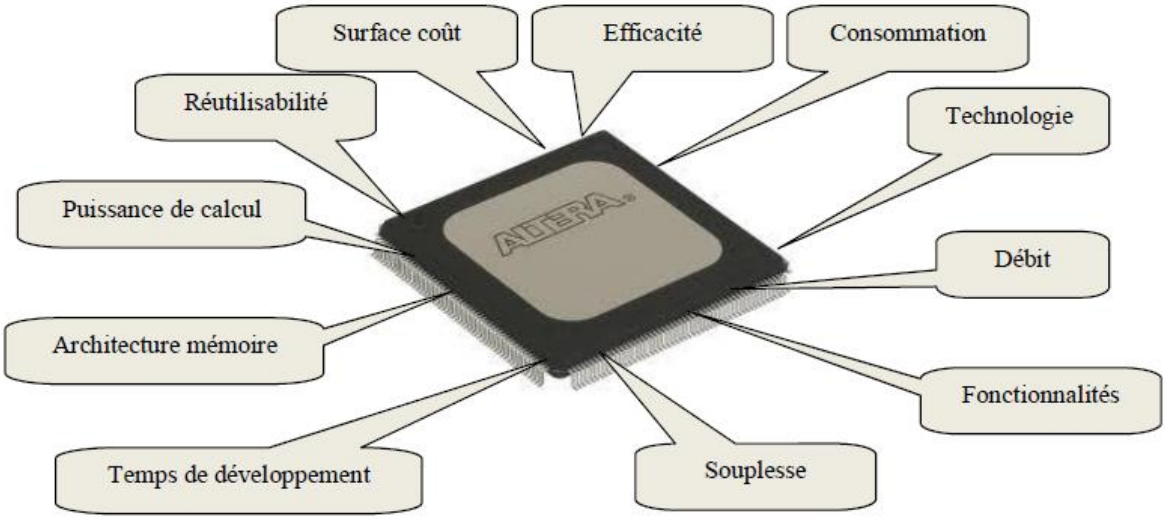


Figure I.13: Critères de choix du circuit logique programmable FPGA.

I.5 Principaux fondeurs d'FPGA

Les fabricants des FPGA ne cessent pas d'améliorer leurs produits par l'efficacité et la puissance. L'ensemble des firmes (Principaux fondeurs) qui conçoivent ce type de circuits sont : **Altera, Actel, Atmel, Cypress, Lattice, Minc, QuicLogic, Xilinx** et d'autres [I.18].

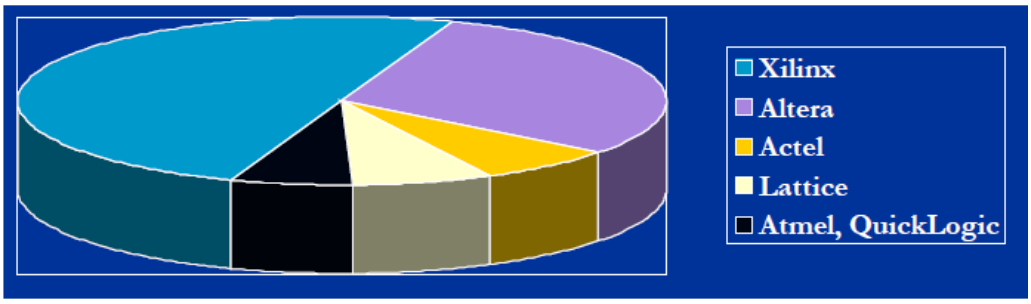


Figure I.14: Statistiques du marché occupé par les vendeurs d'FPGA [I.20].

I.6 Configuration et reconfiguration des FPGA

Un système reconfigurable est un système qui est constitué de composants ou entités à architecture modifiable afin de répondre à un objectif bien déterminé. Ce système reconfigurable dispose d'un mécanisme permettant de choisir une nouvelle configuration et de la mettre en place dans le cadre du processus de reconfiguration. Les circuits FPGA sont un type de ces circuits reconfigurables. Ils sont programmables ou configurables sur les cartes sur quelles ils sont implantés par l'utilisateur. Cette reconfigurabilité est une propriété nécessaire face aux systèmes à charges et contraintes variables [I.18], [I.21].

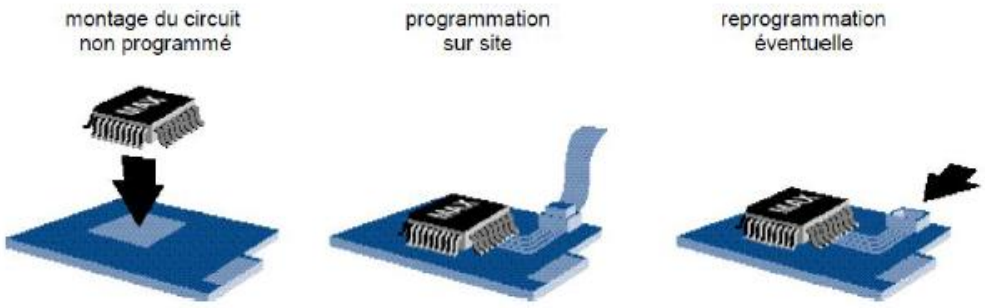


Figure I.15: Reprogrammabilité sur site d'un FPGA

I.7 Méthodologie de conception [I.22]

I.7.1 Outils de CAO (Conception Assistée par Ordinateur) pour la configuration d'un FPGA

Le rôle principal confié aux outils de *CAO* se résume en 4 étapes qui sont : la description, la simulation, la synthèse, le placement et le routage et en dernier la

configuration du FPGA. Un design peut être conçu à l'aide d'un éditeur schématique lors de la conception des circuits simples ou d'un outil de programmation utilisé pour les circuits complexes.

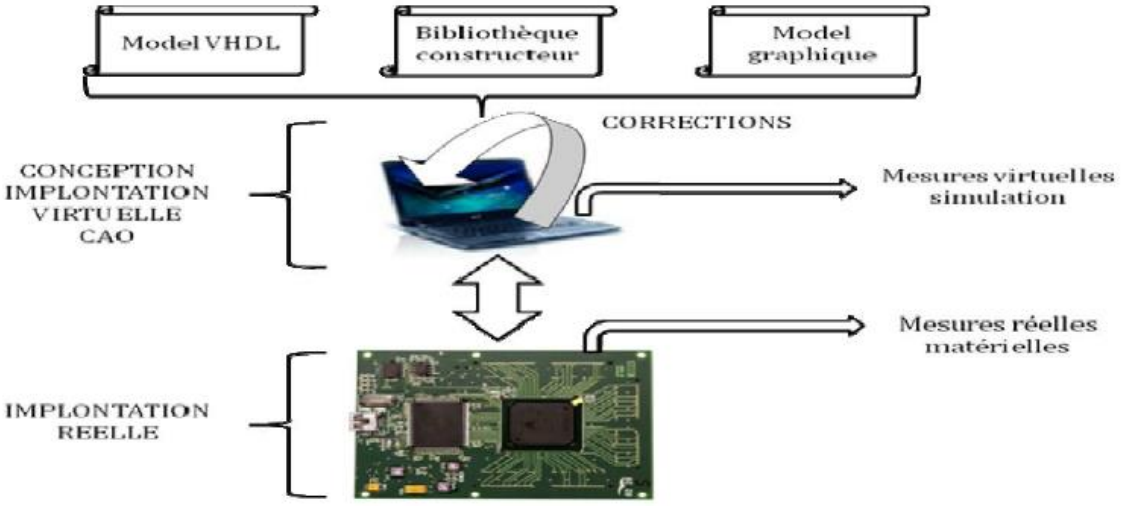


Figure I.16: Mode d'exécution matériel des outils de CAO.

I.7.1.1 Spécification du design

Pour réaliser un circuit, il faut tout d'abord envisager l'architecture globale de ce circuit et spécifier les trois éléments suivants :

- Le nombre de broches d'entrées/sorties et leurs localisations dans le composant FPGA.
- La spécification de la fréquence d'horloge du système.
- La spécification de la mémoire requise pour l'application.

I.7.1.2 Développement du design

- Spécification de la méthodologie de design (Outil de développement utilisé).
- La saisie du circuit Codage RTL (VHDL, Verilog ...)
  - Graphique (Machine à états).
  - Saisie HDL (Hardware Description Language).
- La simulation (Prés et Post synthèse).

I.7.1.3 Synthèse

L'outil de synthèse a pour objectifs de minimiser la surface de silicium, le temps de propagation ainsi que la consommation. Cet outil permet de convertir la représentation du design à partir du code HDL fourni pour produire une représentation au niveau de portes logiques. Cette phase s'occupe de déterminer quelles sont les structures susceptibles pour répondre à un cahier des charges étudié et de produire un code sous forme d'un fichier.

I.7.1.4 Placement et routage

Le placement et le routage sont réalisés en définissant les chemins qui relient l'ensemble des blocs logiques choisies pour notre application à travers un algorithme de routage qui est sensé de faire l'aiguillage des données qu'il reçoit vers leurs destinations par action sur les nœuds de routage.

Plusieurs traitements sont nécessaires pour obtenir un fichier de configuration:

- Partitionnement: Les équations logiques spécifiques de notre application sont regroupées en un autre ensemble équivalent d'équations. Chaque équation de ce nouvel ensemble peut être implémentée dans un seul bloc logique du composant FPGA.

- Placement: Des blocs logiques sont sélectionnés et affectés au calcul des nœuds du réseau booléen.

- Routage: Les ressources d'interconnexion sont affectées à la communication de l'état des nœuds du réseau vers les différents blocs logiques.

-Génération des données numériques de configuration: Les informations abstraites de routage, de placement et les équations implantées dans les blocs sont transformées en un ensemble de valeurs binaires, fournies sous forme d'un fichier appelé « bitstream » qui va être envoyé vers le FPGA via une interface de configuration.

I.7.1.5 Intégration et implémentation

L'implémentation est la réalisation proprement dite qui consiste à mettre en œuvre l'algorithme sur l'architecture du circuit configurable cible, c'est-a-dire à compiler, à charger, puis lancer l'exécution sur un ordinateur ou calculateur. C'est une étape de programmation physique et de tests électriques qui clôturent la réalisation du circuit. La figure suivante résume un peu l'ensemble de ces étapes.

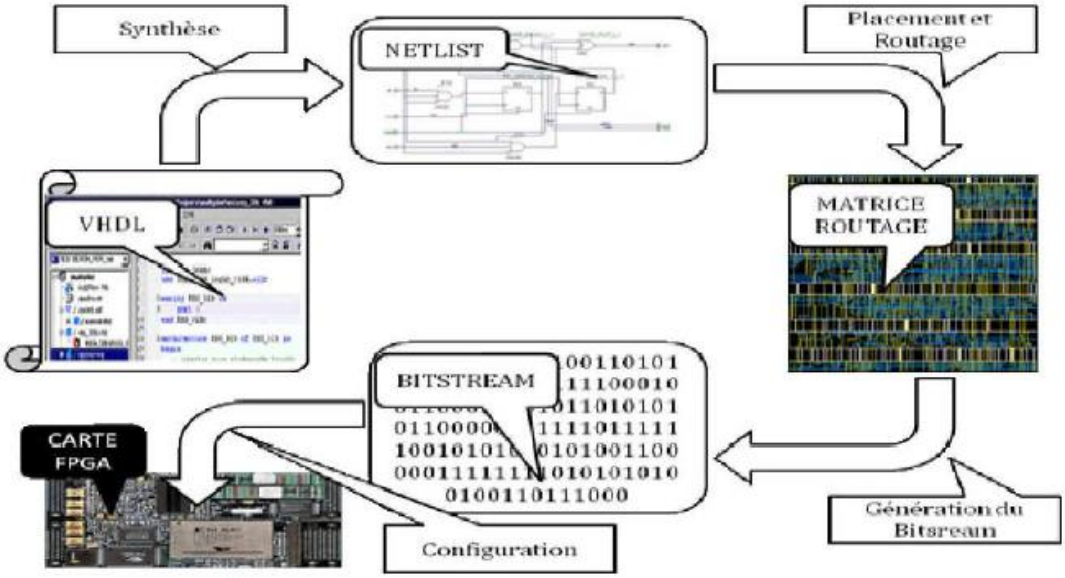


Figure I.17: Cycle de programmation d'un FPGA en utilisant les outils de CAO.

I.8 Principales applications des FPGA [I.23], [I.24]

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

- ✓ Applications Médical
- ✓ Application Militaire
- ✓ Applications Wireline
- ✓ Application Sans fil
- ✓ Véhicules électriques
- ✓ Intelligence artificielle
- ✓ Vision par ordinateur et le prototypage rapide

I.9 Rapide présentation de la carte de développement d'Altera

Pour pouvoir être programmé, le FPGA a besoin d'une carte de développement. C'est un environnement de test et de conception. La figure I.18 présente la carte de développement de la série OMDAZZ.

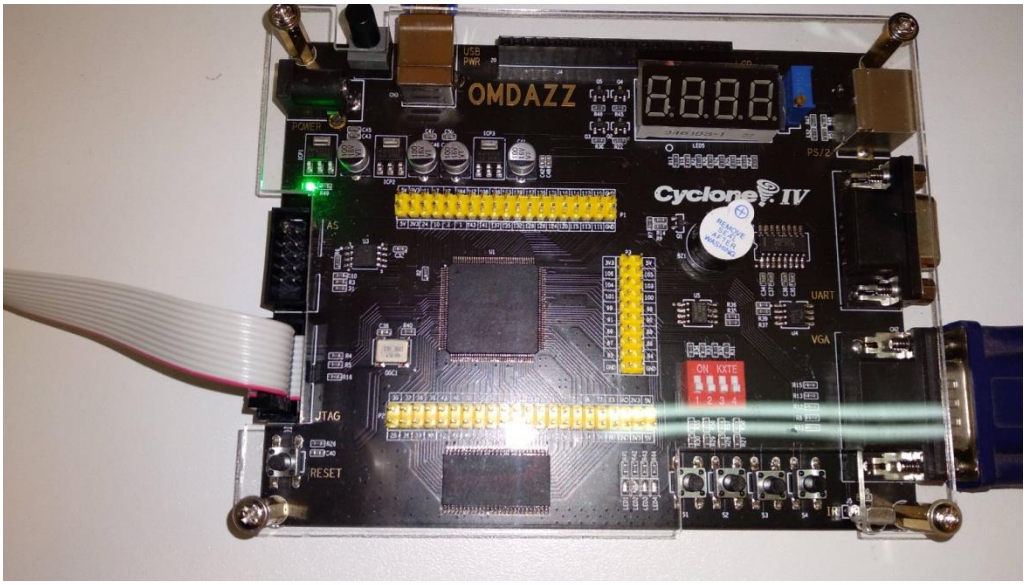


Figure I.18 : Carte de développement FPGA Altera Cyclone IV EP4CE10E22C8.

I.9.1 Caractéristiques de notre composant FPGA EP4CE10E22C8 [I.24]

Le tableau I.1 présente les caractéristiques de notre FPGA.

Tableau I.1 : Caractéristiques de FPGA EP4CE10E22C8.

LE	I /O	Memory bits	PLL	Global clocks
10320	92	423936	2	10

I.9.2 Différentes parties de la carte OMDAZZ [I.24]

Pour permettre aux utilisateurs de disposer de plates-formes expérimentales excellentes, la série OMDAZZ fournit des plates-formes basées sur ALTERA, le composant supporté est EP4CE10E22C8.

La carte (figure I.19), construite autour d'un FPGA ALTERA Cyclone IV (10320 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..).

- 1 interrupteur d'alimentation à bouton autobloquant, 1 clé de réinitialisation, 4 clés utilisateur.
- 4 diodes LED.
- tube à 4 chiffres.
- Commutateur DIP à 4 chiffres.
- 1 buzzer.
- interface ps2.
- port série RS232
- 1 prise LCD à 20 broches, prend en charge LCD1602, LCD12864, LCD TFT.
- Résistance réglable avec précision, rétroéclairage LCD réglable.
- Puce de capteur de température LM754A.
- Interface VGA 8 couleurs.
- Série I2C, pour l'expérience du bus IIC
- Module de réception infrarouge.
- Port série RS232.
- Conduit à toutes les broches de la puce principale, avec un espacement de 2.54mm.

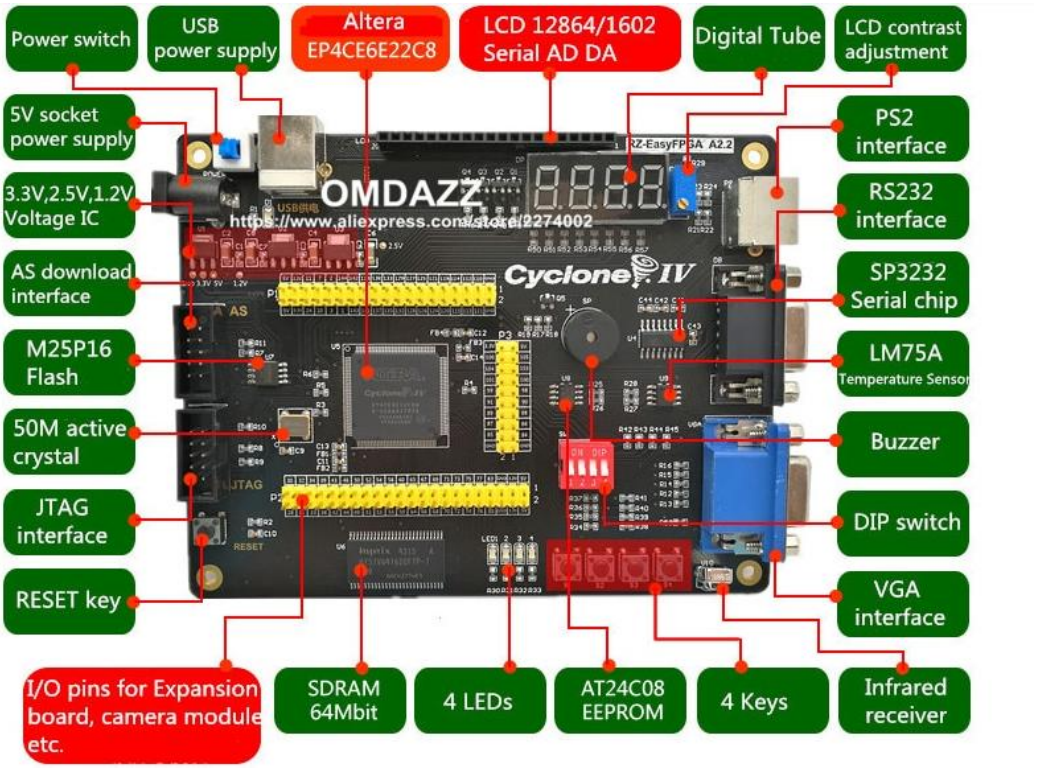


Figure I.19: Constitution de la carte de développement FPGA Altera Cyclone IV.

I.10 Conclusion

Dans ce chapitre nous avons vu une présentation des circuits logique programmables FPGAs ainsi que les principaux composants et éléments qui les caractérisent, leurs avantages et différents domaines d'applications, ce qui nous a permis de conclure que la technologie FPGA s'inscrit au sommet de l'évolution des composants logiques et le besoin croissant de composants plus performants, plus économiques et disponibles en grandes quantités avec un faible coût est les grands axes du progrès qui sont disponibles dans les FPGA. 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.

Pour programmer un FPGA nous avons besoin d'un langage de description matériel, dans le chapitre suivant nous allons présenter le langage VHDL, le signal vidéo VGA et son application pour la vision artificielle.

REFERENCES BIBLIOGRAPHIQUES -CHAPITRE I-

[I.1] AOUICHE Mounir, « Conception et implémentation d'un Microcontrôleur de 64 bits sur FPGA», Mémoire master en Réseaux et télécommunications Université Mohamed Khider Biskra, 2019.
[I.2] http://proxacutor.free.fr/index.htm (site consulté le 25/03/2019)
[I.3] https://www.cder.dz/vlib/bulletin/pdf/bulletin_024_05.pdf Les circuits FPGA : description et applications GUELLAL Amar Attaché de recherche (site consulté le 25/03/2019).
[I.4] MARIANI, Johanna, « Programmation et Utilisation du FPGA pour la validation et la vérification de circuits électroniques », mémoire Présenté en vue d'obtenir Le diplôme d'ingénieur CNAM spécialité : électronique, CENTRE D'ENSEIGNEMENT DE GRENOBLE, 2011.
[I.5] TISSERAND Arnaud : Introduction aux circuits FPGA, Présentation Séminaire MIM, 2003.
http://www.irisa.fr/prive/Arnaud.Tisserand/docs/semmim-at-fpga.pdf
[I.6] ALTERA Corporation : Site d'un constructeur de FPGA, Ensemble de documents techniques concernant les FPGA, 2010.
http://www.altera.com/literature/lit-index.html
[I.7] Mickaël RAULET, «Optimisations Mémoire dans la Méthodologie AAA pour Code Embarqué sur Architectures Parallèles', Thèse, Institut National Des Sciences Appliquées De Rennes, Mai 2006.
[I.8]Michel PAINDAVOINE, «Traitement des images en temps réel - Applications industrielles », Techniques de l'Ingénieur, Mesures et Contrôle, R 6-720, 1-10.
[I.9] Abdelhalim SAMAHI, «Contribution à la mise en œuvre d'une plate-forme de prototypage rapide pour la conception des systèmes sur puce», Thèse, Laboratoire LE2I, Université de Bourgogne, 2007.
[I.10] MESSAOUDI, Kamel « Traitement des signaux et images en temps réel:" implantation de H. 264 sur MPSoC», Thèse de doctorat en science spécialité électronique. Université de Bourgogne – Dijon, 2012.
[I.11] http://www.electronique-mag.com/article2230.html
[I.12] https://www.intel.fr/content/www/fr/fr/products/details/fpga/cyclone/iv.html
[I.13] https://www.ee.ryerson.ca/~courses/coe608/Data-Sheets/Cyclone-IV.pdf
[I.14] file:///C:/Users/a/Downloads/wp-01113-lowest-system-cost.pdf
[I.15] https://www.ti.com/lit/ug/tidu737/tidu737.pdf

[I.16]https://www.intel.fr/content/www/fr/fr/products/details/fpga/cyclone/iv/e.html
[I.17] THOMPSON Mike « FPGAs accelerate time to market for industrials » designs, article de l'EETimes concernant les avantages de l'utilisation du FPGA dans l'industrie, 2004.
http://www.designreuse.com/articles/exit/?id=8190&url=http://www.eetimes.com/showArticle.jhtml;jsessionid=4OJLA20JQAVNSQSNDBGCKHSCJUMEKJVN?articleID=22102798

[I.18] GUETTAT, ABDELGHANI, « Conception et Implémentation d'un Corrélateur Numérique sur FPGA », Thèse de doctorat. USTO, 2012.
[I.19] SEBASTIEN SNAIDERO, « Modélisation multidisciplinaire VHDL-AMS de systèmes complexes vers le prototypage virtuel », Thèse présentée afin d'obtenir le grade de docteur de l'université Louis Pasteur Strasbourg I, décembre 2004.
[I.20] DAVID GUIHAL, «Modélisation en langage VHDL-AMS des systèmes Pluridisciplinaires», Thèse présentée au laboratoire d'analyse et d'architecture des systèmes du CNRS en vue de l'obtention du titre de docteur de l'université Toulouse, Mai 2007.
[I.21] Synopsys, «Saber HDL : language-Independant Mixed –Signal Multi-Technology Simulator », Synposys Inc, Etats-Unis d'Amérique 2003.
[I.22] FANDI Tadj Eddine, « Simulation d'un classifieur neurenal sur FPGA », mémoire de Master en génie biomédical Spécialité : Electronique Biomedical, Université Abou Bakr Belkaïd de Tlemcen, 2013.
[I.23] WWW.ALTERA.COM
[I.24] HOUARI ABBAD, «méthodologie de développement et d'une plantation sur puce FPGA d'algorithme de commande » mémoire présenté en vue de l'obtention du diplôme de magister en électronique option signaux et communication université MOHAMED KHIDER, BISKRA 2016.
[I.25]https://onedrive.live.com/?authkey=%21ABZ%5Fj27B4HZ3Adg&id=B2CDC3A30980D5BD%2182722&cid=B2CDC3A30980D5BD

Chapitre II

Présentation de VHDL, signal vidéo VGA et la vision artificielle

II.1 Introduction

VGA (Video Graphics Adapter) est une norme de signal vidéo que l'on trouve principalement dans les ordinateurs individuels. Ce signal vidéo est constitué de 5 signaux, 3 analogiques avec des niveaux de 0.7 à 1.0 volt pour chacune des couleurs Rouge, Vert, Bleu (RGB) accompagnés de 2 signaux logiques que sont synchronisation horizontale et verticale.

Le VHDL est un langage de description de matériel qui est utilisé pour la spécification (Description du fonctionnement), la simulation et la preuve formelle d'équivalence de circuits. Ensuite il a aussi été utilisé pour la synthèse automatique. L'abréviation VHDL signifie VHSIC (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit) **H**ardware **D**escription **L**angage (langage de description de matériel pour circuits à très haute vitesse d'intégration) [II.1], [II.2]. Au début, ce langage était uniquement destiné à décrire les circuits intégrés déjà conçus et devait permettre de réaliser des documentations techniques facilement interprétables par certaines personnes. Aujourd'hui, la finalité de ce langage a bien changée, puisque il est essentiellement utilisé à concevoir et modéliser les circuits, non plus dans un but de documentation, mais de simulation. Car on la étendu en lui rajoutant des extensions pour permettre la conception (synthèse) de circuits logiques programmables (P.L.D. Programmable Logic Device).

Auparavant pour décrire le fonctionnement d'un circuit électronique programmable les techniciens et les ingénieurs utilisaient des langages de bas niveau (ABEL, PALASM, ORCAD/PLD,..) ou plus simplement un outil de saisie de schémas.

Actuellement la densité de fonctions logiques (portes et bascules) intégrée dans les PLDs est telle (plusieurs milliers de portes voire millions de portes) qu'il n'est plus possible d'utiliser les outils d'hier pour développer les circuits d'aujourd'hui.

Les sociétés de développement ainsi que les ingénieurs ont voulu s'affranchir des contraintes technologiques des circuits PLD, en créant des langages plus faciles qui sont VHDL et VERILOG.

Ces langages permettent au code écrit d'être portable, de façon qu'une description écrite pour un circuit puisse être facilement utilisée pour un autre circuit. Ceci permet de matérialiser les structures électroniques d'un circuit. En effet les instructions écrites dans ces langages se traduisent par une configuration logique de portes et de bascules qui est intégrée à l'intérieur des circuits PLDs. C'est pour cela qu'on préfère parler de description VHDL ou VERILOG que de langage [II.3], [II.4].

Dans ce chapitre nous nous intéresserons seulement au VHDL et aux fonctionnalités de base de celui-ci lors des phases de conception ou synthèse (c'est à dire à la conception de PLD) qui sera présenté dans la deuxième partie de ce chapitre.

Dans la deuxième partie de ce chapitre, nous présentons, une généralité sur l'image, une présentation de signal vidéo VGA et comment générer l'affichage vidéo VGA à l'aide d'un FPGA en suite nous présentons une introduction sur la vision artificielle.

La conception des tels systèmes numériques intégrés a généralement recours aux langages numériques de description de matériel comme le VHDL.

II.2 Langages de description matérielle

II.2.1 Historique [II.2]

Au début des années 80, le département de la défense américaine (DOD) désire standardiser un langage de description et de documentation des systèmes matériels ainsi qu'un langage logiciel afin d'avoir une indépendance vis-à-vis de leurs fournisseurs. C'est pourquoi, le DOD a décidé de définir un langage de spécification. Il a ainsi mandaté des sociétés pour établir un langage. Parmi les langages proposés, le DOD a retenu le langage VHDL qui fut ensuite normalisé par IEEE. Le langage ADA est très Proche, car celui-ci a servit de base pour l'établissement du langage VHDL.

La standardisation du VHDL s'effectuera jusqu'en 1987, époque à laquelle elle sera normalisée par l'IEEE (Institute of Electical and Electronics Engineers). Cette première normalisation a comme objectif:

- La spécification par la description de circuits et de systèmes.
- La simulation afin de vérifier la fonctionnalité du système.
- La conception afin de tester une fonctionnalité identique mais décrite avec des solutions d'implémentations de différents niveaux d'abstraction.

En 1993, une nouvelle normalisation par l'IEEE du VHDL a permis d'étendre le domaine d'utilisation du VHDL vers:

- La synthèse automatique de circuit à partir des descriptions.
- La vérification des contraintes temporelles.
- La preuve formelle d'équivalence de circuits.

II.2.2 Langages HDL [II.5]

Depuis le début des années 90, l'intégration de transistors sur une même puce contribue à l'intégration "*single chip*" dans le domaine de la commande numérique des systèmes de puissance. Cette évolution a également ouvert la voie aux langages de haut niveau de description de matériel, encore appelés HDLs pour "***H**ardware **D**escription **L**anguages*". Deux d'entre eux ont émergé et sont aujourd'hui couramment utilisés : il s'agit de VHDL et de Verilog. Tous les deux sont supportés par un grand nombre de logiciels.

Les intérêts majeurs d'une description basée sur un HDL résident dans sa portabilité et son caractère exécutable. En effet, un modèle fonctionnel numérique décrit à haut niveau par un HDL peut être vérifié par simulation, avant même son conception finale.

La conception des tels systèmes numériques intégrés a généralement recours aux langages numériques de description de matériel comme par exemple VHDL.

II.2.3 Langage de description matérielle VHDL

Le VHDL est basé sur l'assemblage de composants nommés entités. Ces entités peuvent être dénies par le langage sous la forme d'instructions séquentielles et/ou parallèles. Ces instructions peuvent soit correspondre à l'assemblage d'autres entités, soit correspondre à des instructions de base du langage. Certaines variables peuvent être évaluées dans le temps. La communication entre les entités se fait par l'intermédiaire de mémoires partagées représentées par des signaux.

VHDL est un langage qui permet de représenter la plupart des concepts nécessaires à la description des systèmes temps réel. Il n'a cependant pas d'instructions spécifiques aux transitions d'états et permettant de définir des exceptions [II.5], [II.6].

Le langage VHDL exploite quatre objets différents :

- Les signaux.
- Les constantes.
- Les variables.
- Les portes

II.2.3.1 Structure d'un programme VHDL

La structure ou bien l'unité de conception est un ensemble d'éléments VHDL avec les quels nous allons décrire un système numérique. Celui-ci peut-être constitué d'une simple porte logique jusqu'à un système complexe. Nous parlerons aussi de module VHDL. L'unité de conception est constituée d'une entité (définit l'interface), une ou plusieurs architectures (défini le fonctionnement), des bibliothèques et de la configuration. Les bibliothèques regroupent un ensemble de définition, déclarations, fonctions, etc…nécessaire à tous les modules. La configuration est optionnelle [II.4], [II.7].

Donc un programme écrit sous VHDL obéit à la structure suivante:

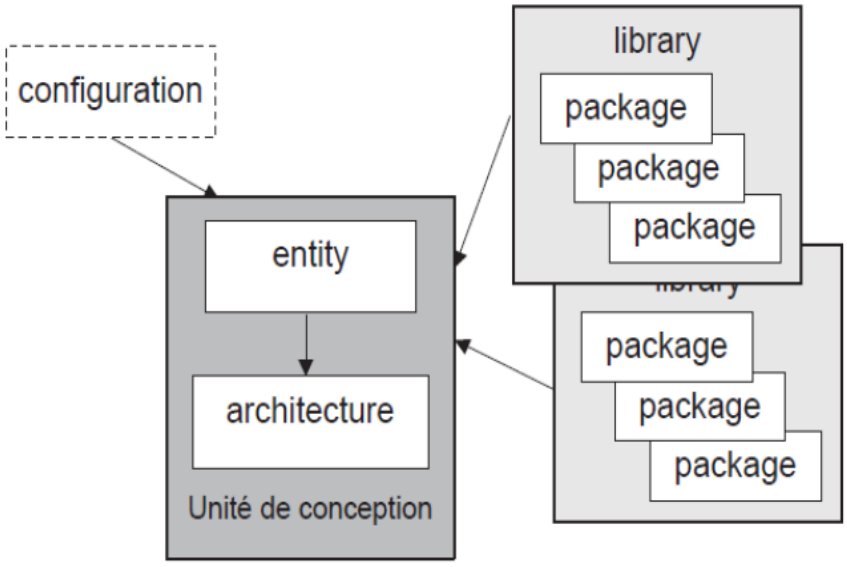


Figure II.1 : Représentation d'une unité de conception [II.7].

II.2.3.1.1 Entête

C'est une partie facultative, elle referme des informations concernant le programmeur, la description du programme en général, la date de rédaction et toute information qui semblera importante pour celui qui rédige le programme [II.3].

II.2.3.1.2 Déclaration des librairies

Toute description VHDL utilisée pour la synthèse a besoin de bibliothèques. L'IEEE (Institut of Electrical and Electronics Engineers) les a normalisées et plus particulièrement la bibliothèque IEEE 1164. Elles contiennent les définitions des types de signaux électroniques, des fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques,... [II.8].
Donc, la première chose à faire est de définir les librairies qui seront utilisées dans le code.

- *Library ieee ;*
- *Use ieee.std_logic_1164.all ;*
- *Use ieee.numeric_std.all ;*
- *Use ieee.std_logic_unsigned.all ;*

-- cette dernière bibliothèque est souvent utilisée pour l'écriture de compteurs
Cela se fait de la manière suivante :
Tout d'abord, **la librairie principale** (en générale, IEEE).

Ensuite, le mot clé « **use** », qui indique quelle **package** de la librairie nous allons utiliser. Après cela, le **nom du package**. Enfin, le **.all** signifie que l'on souhaite utiliser tout ce qui se trouve dans ce package. Lorsque le nom de la librairie est précédé de IEEE, cela signifie que c'est une librairie qui est définie dans la norme IEEE, et que l'on retrouvera donc normalement dans tout logiciel. A l'inverse, il faut se méfier des librairies qui ne sont pas IEEE, car elles sont en générale spécifiques à un logiciel.

Les librairies IEEE principales sont :

- *IEEE.standard*
- *IEEE.std_logic_1164*
- *IEEE.numeric_std*
- *IEEE.std_logic_arith*

Attention, il ne faut pas utiliser les librairies numeric_std et std_logic_arith en même temps : la librairie std_logic_arith est en fait une version améliorée de la numeric_std, développé par synopsys, et qui a été ensuite incorporé dans la norme IEEE. Le fait d'utiliser les 2 librairies en même temps causera un conflit lors de l'utilisation de certaines fonctions.

II.2.3.1.3 Déclaration d'entité

Tout programme en VHDL est défini par une déclaration d'entité. Cette entité permet de décrire l'interface avec l'environnement extérieur. On y retrouve donc un nom d'entité, et la liste des signaux, avec leurs caractéristiques (nom, mode, type).

La déclaration d'entité peut être partagée par plusieurs entités de conceptions, dont chacune possède une architecture différente car elle peut être vue comme une classe d'entité de conception, et présente les mêmes interfaces.

Elle permet de définir le nom de l'entité et aussi les entrées, sorties et entrées/sorties qui sont utilisées, et c'est l'instruction ‹‹**port** ›› qui les a définit.

La syntaxe générale de l'entité :

```
Entity Nom_de_l'entité is
Port (Nom_entrée_1 : in  type _du_signal ;
Nom_entrée_2 : in  type_du_signal ;
.....
Nom_sortie_1 : out  type_du_signal ;
Nom_E_S_1 : inout  type _du_signal
);
End Nom_de_l_entite ;
```

II.2.3.1.3.1 Signal d'entré/sortie

a) Le NOM_DU_SIGNAL

Il est composé de caractères, le premier caractère doit être une lettre, sa longueur est quelconque, mais elle ne doit pas dépasser une ligne de code. **VHDL** n'est pas sensible à la « casse », c'est à dire qu'il ne fait pas la distinction entre les majuscules et les minuscules.

b) Le SENS du signal

- **in** : pour un signal en entrée.
- **out** : pour un signal en sortie.
- **inout** : pour un signal en entrée sortie
- **buffer** : pour un signal en sortie mais utilisé comme entrée dans la description.

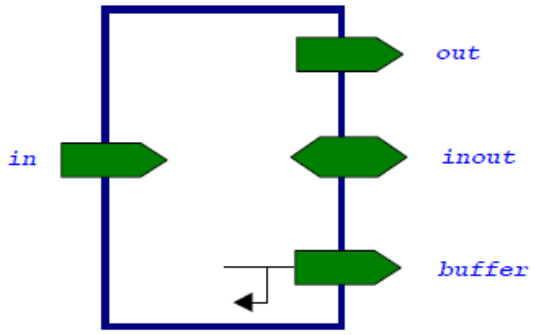


Figure II.2 : Représentation de sens de signal.

c) Le TYPE

Le **TYPE** utilisé pour les signaux d'entrées / sorties est :

- le **std_logic** pour un signal (pour les données à un bit).
- le **std_logic_vector** pour un bus composé de plusieurs signaux.

Les valeurs que peuvent prendre un signal de type **std_logic** sont :

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

II.2.3.1.4 Architectures

L'architecture décrit la vue interne du modèle, elle décrit le fonctionnement souhaité pour un circuit ou une partie du circuit.

En effet le fonctionnement d'un circuit est généralement décrit par plusieurs modules VHDL. Il faut comprendre par module le couple ENTITE/ARCHITECTURE. Dans le cas de simples PLDs on trouve souvent un seul module.

L'architecture est établit à travers les instructions les relations entre les entrées et les sorties. On peut avoir un fonctionnement purement combinatoire, séquentiel ou les deux (séquentiel et combinatoire) en même temps.

En VHDL la boîte noire est nommé entité (entity) car l'entité doit toujours être associée avec au moins une description de son contenu, de son implémentation qui est l'architecture.

Une architecture fait toujours référence à une entité.

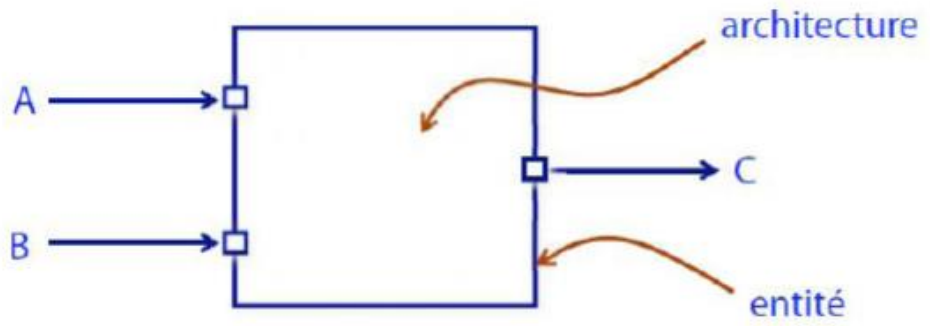


Figure II.3 : Représentation de l'entité et l'architecture.

Une architecture est un ensemble de processus qui s'exécutent en parallèle. Elle est définie par un nom, que l'on pourra choisir pour expliciter la façon dont on code.

A la suite de la déclaration de l'architecture, on définira les déclarations préalables (signaux internes, composants).

Ensuite, viens le mot clé « **begin** ». A sa suite, on trouvera le code qui décrit le fonctionnement de l'architecture.

La description d'une architecture peut prendre trois formes :

- Comportementale
- Structurelle
- Mixte

La description qu'elle soit structurelle, comportementale, ou mixte se fait de la manière suivante :

```
Architacture description of nom_entité is
  {Partie déclarative}
Begin
  {Partie descriptive}
End description ;
```

II.2.3.1.4.1 Description comportementale [II.9]

Ce type de description décrit le fonctionnement du circuit à réaliser et le simuler en fonction des équations logiques qui relient les entrés aux sorties. Cette description est représentée dans la syntaxe suivante :

```
ARCHITECTURE comportementale of circuit is
  -Partie déclarative.
  BEGIN
  -partie descriptive.
END comportementale ;
```

Ce type de description à deux moyennes de représentations qui sont :

a. Sous forme de flow de données (DATA FALOW)

Dans ce type de description comportementale DATA FALOW, on modélise le circuit par un ensemble d'équations logiques et arithmétiques, car elle consiste à décrire chaque sortie par une équation en fonction des entrées.

Example

ARCHITECTURE XOR of circuit is BEGIN
S1 <= IN1 XOR IN2; S2 <= IN1 XOR IN3; END XOR;

b. Sous forme d'instruction séquentielle

Sous cette forme, le contenu est décrit de façon algorithmique en utilisant les structures des langages de programmation, à savoir : les déclarations séquentielles suivantes :

- **La structure alternative**

If <condition> THEN Séq_stat
ELSIF <condition> THEN Séq_stat ;
END IF;

- **La structure d'aiguillage**

CASE <identificateur> IS
WHEN choix= séq_stat ;
WHEN others= séq_stat ;
END CASE;

Remarque:

Séq_stat désigne une ou plusieurs déclarations séquentielles.

II.2.3.1.4.2 Description structurelle

Dans ce type de description, les interconnexions des composants préalablement décrits sont énoncées. Cette description est la transcription directe d'un schéma. Elle se compose de trois rubriques qui sont :

- **Déclaration des signaux internes destinés à interconnecter les composants :**

Cette déclaration se fait par le mot clé <**SIGNAL**> accompagné par le nom et le type du signal utilisé.

- **Déclaration des composants utilisés :**

Cette étape permet de lister les composants utiles pour la construction de l'entité.

Elle se fait de la manière suivante :

COMPONET Composant IS
Port (A: in bit, B: out bit);
END COMPONET;

- **Représentation de la réalisation des différentes interconnexions des composants déclarés dans la partie déclarative :**

Chaque connexion est définie de la manière suivante :

Comp 1 : composant port map (….) ;

Où :

Comp1 : représente l'affectation du mapping.

Composant : est le nom de la cellule qui entre dans la constitution de l'entité.

Port map : est le mot clé qui réalise la connexion entre les différents niveaux, signaux utilisés, internes et externes.

II.2.3.1.4.3 Description mixte [II.9]

Elle regroupe les deux descriptions décrites précédemment. À chaque entité peut être associée à une ou plusieurs architectures mais au moment de l'exécution (Simulation, synthèse…) seulement une architecture et un seule est utilisée.

Cette dernière est spécifiée par le mécanisme de package.

L'architecture dépend implicitement de l'entité à laquelle elle est associée ; tous les objets définis dans l'entité sont connus par l'architecture et ils sont vus comme des signaux qui peuvent être lus ou écrits à cet endroit, cela se fait par le biais d'instructions concurrentes énumérées au niveau du corps de l'architecture. Cette architecture comporte aussi une partie déclarative où peuvent figurer un certain nombre de déclarations (de signaux, de composants …etc.) internes à l'architecture.

II.2.3.1.4 Relation entre une structure VHDL et un circuit numérique

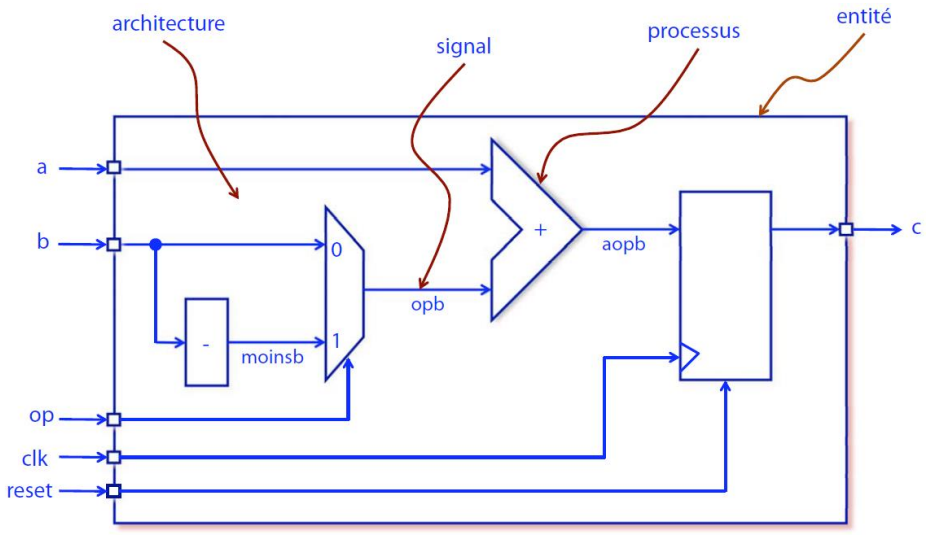


Figure II.4 : Relation entre les composants d'un VHDL et un circuit numérique.

- Les entrées/sorties du système sont les **ports** de l'entité.
- Chaque composant interne du système sera un processus (**process**) de l'architecture.
- Une **architecture** est un ensemble de processus.
- Les processus s'exécutent en parallèle.
- Les processus de l'architecture sont interconnectés par le biais des signaux (**signal**).
- Quelques notes sur la syntaxe d'un programme VHDL:
    - pas de différentiation entre majuscules et minuscules
    - format libre
    - toute phrase termine par un point virgule
    - le début d'un commentaire est signalé par un double trait ("--"). Le commentaire termine avec la fin de ligne

II.2.4 Types d'instructions utilisées en VHDL

II.2.4.1 Instructions concurrentes

L'ordre d'écriture n'a alors pas d'importance, les opérations étant réalisées simultanément. Comme montre la figure II.5 [II.2], [II.10].

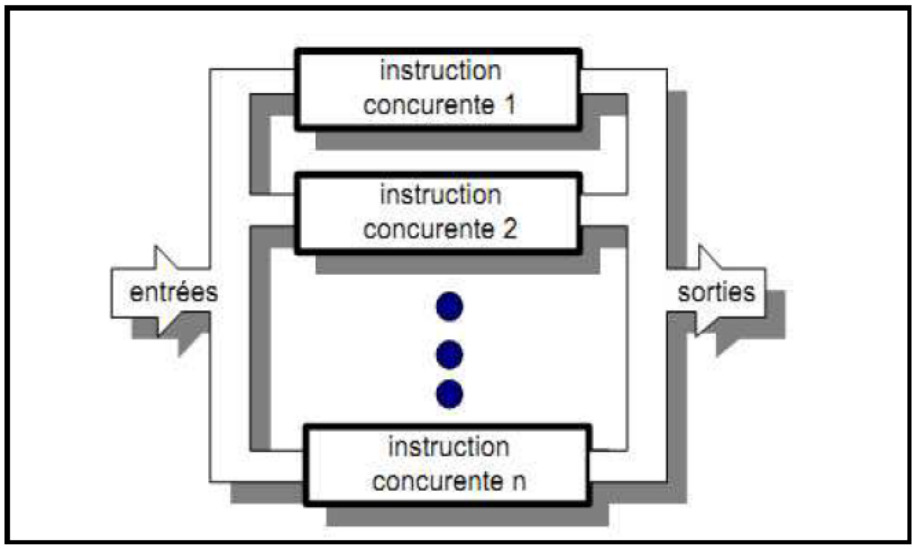


Figure II.5: Instructions en mode concurrent.

II.2.4.1.1 Affectation simple

Dans une description VHDL, c'est certainement l'opérateur le plus utilisé. En effet il permet de modifier l'état d'un signal en fonction d'autres signaux et/ou d'autres opérateurs.

NOM_D'UNE_GRANDEUR<=VALEUR_OU_NOM_D'UNE_GRANDEUR ;

II.2.4.1.2 Affectation conditionnelle

L'interconnexion est cette fois soumise à une ou plusieurs conditions.

NOM_D'UNE_GRANDEUR<=Q1 when CONDITION I else
Q2 when CONDITION 2 else
...
Qn ;

On note l'absence de ponctuation à la fin des lignes intermédiaires.

II.2.4.1.3 Affectation sélective

Cette instruction permet d'affecter différentes valeurs à un signal, selon les valeurs prises par un signal dit de sélection.

With EXPRESSION select
NOM_D'UNE_GRANDEUR<=Q1 when valeur1,
Q2 when valeur 2,
... ,
Qn when others ;

On note la présence d'une virgule (et non un point virgule) à la fin des lignes intermédiaires.

II.2.4.1.4 Instanciation du composant

Consiste à utiliser un sous-ensemble décrit en VHDL comme composant dans un ensemble plus vaste.

L'instanciation d'un composant se fait dans le corps de l'architecture de cette façon :

<nom_instance>:<nom_composant> ***port map*** *(liste des connexions);* [II.3], [II.4] [II.11].

II.2.4.2 Instructions séquentielles

Un ou plusieurs événements prédéfinis déclenchent l'exécution des instructions dans l'ordre où elles sont écrites. Comme indique la figure II.6 [II.12].

II.2.4.2.1 Définition d'un process

Un process est une partie de la description d'un circuit dans laquelle les instructions sont exécutées séquentiellement c'est à dire les unes à la suite des autres.

Un process peut être défini par un label, mais ce n'est pas obligatoire. Il permet d'effectuer des opérations sur les signaux en utilisant l'instruction standard de la programmation structurée comme dans les systèmes à microprocesseurs.

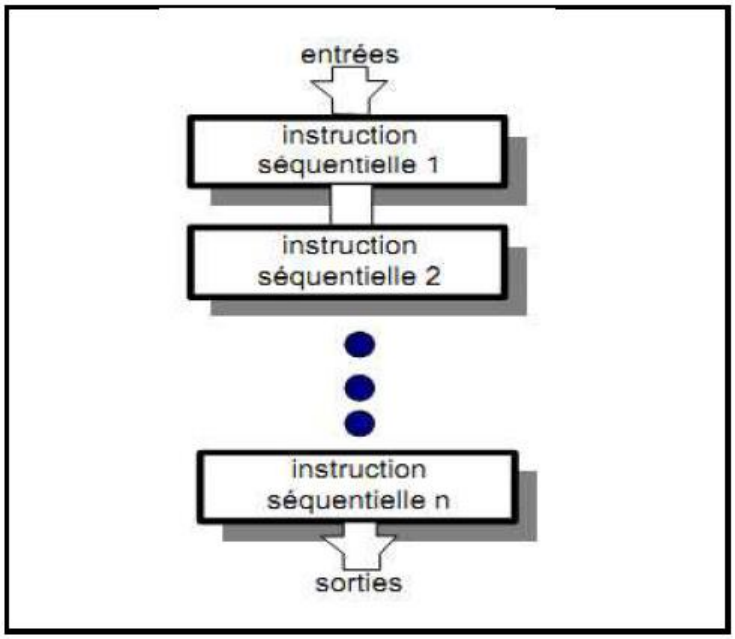


Figure II.6. Instructions en mode séquentiel.

L'exécution d'un process est déclenchée par un ou des changements d'états de signaux logiques. Le nom de ces signaux est défini dans la liste de sensibilité lors de la déclaration du procès.

```
[Nom_du_process:] Process (LISTE_DE_SENSIBILITE)
        NOM_DES_OBJETS_INTERNES :«type» ; --zone facultative
        begin
            INSTRUCTIONS_SEQUENTIELLES ;
End  process ;
```

Remarque:

Le nom du ***process*** entre crochet est facultatif, mais il peut être très utile pour repérer un ***process*** parmi d'autres lors de phases de mise au point ou de simulations.

II.2.4.2.2 Principales instructions utilisées dans un process

II.2.4.2.2.1 Instruction conditionnelle [II.4], [II.9]

Cette instruction est très utile pour décrire à l'aide d'un algorithme le fonctionnement d'un système numérique. La syntaxe générique de l'instruction conditionnelle est la suivante:

```
If Condition_Booléenne_1 then
  --Zone pour instructions séquentielles
   elsif Condition_Booléenne_2 then
  --Zone pour instructions séquentielles
  elsif Condition_Booléenne_3 then
     ...
   else
  --Zone pour instructions séquentielles
  end if ;
```

II.2.4.2.2.2 Instruction de choix

La syntaxe générique d'une instruction de choix est la suivante:

```
case Expression is
     when Valeur_1  =>  --Zone pour instructions séquentielles
     when Valeur_2  =>  --Zone pour instructions séquentielles....
     when others    =>  --Zone pour instructions séquentielles
end case;
```

II.2.4.2.2.3 Instruction wait

L'instruction wait suspend l'exécution d'un process jusqu'à ce qu'un événement, une condition ou une clause de temps écoulé (time out) soit vraie. Si aucune clause de réveil n'est stipulée, le processus s'arrête définitivement [II.4], [II.13].

« wait » peut être utilisé des manières suivantes:

wait on [nom Signal1, nom Signal2…] ;

wait until [expression Booléenne] ;

wait for [expression Temps] ;

II.2.4.2.2.4 Boucles

Les boucles permettent de répéter une séquence d'instructions. Trois catégories de boucles existent en VHDL, suivant le schéma d'itération choisi :

- Les boucles simples, sans schéma d'itération, dont on ne peut sortir que par une instruction « **exit** ».
- Les boucles « **for** », dont le schéma d'itération précise le nombre d'exécution.

- Les boucles « **while** », dont le schéma d'itération précise la condition de maintien dans la boucle [II.14], [II.4].

Dans la partie suivante, nous présentons les différents types des opérateurs que l'on rencontre dans le VHDL.

II.2.5 Opérateurs de base [II.2]

Le langage VHDL comporte six classes d'opérateurs avec un niveau de priorité défini pour chaque classe. Lors de l'évaluation d'une expression, l'opération dont la classe a la plus haute priorité est effectuée en premier. Etudions, l'une après l'autre, chacune de ces six classes en les donnant par ordre de priorité croissante, les opérations logiques ayant donc le niveau de priorité le plus faible [II.15].

II.2.5.1 Opérations logiques

Ce sont les opérations and, or, nand, nor, xor, soit les fonctions logiques: ET, OU, NON-ET, NON-OU, OU exclusif. Les cinq opérateurs logiques ont la même priorité, d'où l'intérêt et quelquefois la nécessité de parenthèses. Ainsi, l'opération A or B and C, est impossible à interpréter sans parenthèses.

Tableau II.1 : Ecriture des opérateurs logique en VHDL.

Opérateur	VHDL
ET	And
NON ET	Nand
OU	Or
NON OU	Nor
OU EXCLUSIF	Xor
NON OU EXCLUSIF	Xnor
NON	Not
DECALAGE A GAUCHE	SII
DECALAGE A DROITE	Srl
ROTTION A GAUCHE	Rol
ROTATION A DROITE	Ror

II.2.5.2 Opérations relationnelles

Ce sont les relations qui expriment la relation entre deux grandeurs: égal, inégal, plus petit, plus petit ou égal, plus grand, plus grand ou é (=, /=, <, <=, », >=). Le résultat de la comparaison est de type booléen, la relation concernée ne pouvant qu'être vrai ou fausse. Pour rendre l'écriture plus lisible, il est utile et conseillé de mettre entre parenthèses l'opération relationnelle.

Tableau II.2: Ecriture des opérateurs relationnels en VHDL.

Opérateur	VHDL
Egal	=
Nom égal	/=
Inférieur	<
Inférieur ou égal	<=
Supérieur	>
Supérieur ou égal	>=

II.2.5.3 Opérations d'addition

Elles sont au nombre de trois: l'addition, la soustraction et la concaténation de symboles respectifs : +, -, et &. La concaténation est définie pour des chaînes de bits et des chaînes de caractères. C'est une simple juxtaposition des valeurs.

II.2.5.4 Opérations de signe

Ce sont les signes" + " ou "-". Ces opérations sont définies, c'est-à-dire valides, pour des objets de type entier ou flottant.

II.2.5.5 Opérations de multiplication

Ce sont les opérations suivantes:

- La multiplication, de symbole "*".
- La division symbole ‘’ / ‘’.
- Le modulo, de symbole ‘’mod’’.
- Le calcul de reste, (remainder) de symbole Il ‘’rem’’. Elles sont définies sur les types entier et flottant.

II.2.5.6 Opérations NOT, ABS et **

Ce sont les trois opérations de plus haute priorité. L'opération NOT prend le complément de la valeur d'un objet de type bit, booléen, bit_vector et également d'une chaîne d'éléments de type booléen.

Exemples

C < = not A ; -- C = complément de A

C < = not (A or B) ; -- C = complément (A+B), la parenthèse est nécessaire ici.

L'opération ABS rend la valeur absolue. Elle est définie sur les entiers et les flottants.

L'opération ** est l'exponentiation, c'est-à-dire l'élévation de l'opération de gauche à la puissance définie par l'opérande de droite. Ce dernier doit être un nombre entier tandis que l'opérande de gauche doit être de type entier ou flottant. Exemple:

A ** B est égal à AB.

Après avoir présenté la syntaxe des déclarations, un paquetage permet de grouper ces déclarations et de les stocker dans une bibliothèque.

II.2.5.7 Sous programmes

Les sous programmes sont le moyen par lequel le programmeur peut se constituer une bibliothèque d'algorithmes séquentiels qu'il pourra inclure dans une description.

Les deux catégories de sous programmes, procédures et fonctions, diffèrent par les mécanismes d'échanges d'informations entre le programme appelant et le sous programme.

II.2.5.7.1 Fonctions

Une fonction retourne au programme appelant une valeur unique, elle a donc un type. Elle peut recevoir des arguments, exclusivement des signaux ou des constantes, dont les valeurs lui sont transmises lors de l'appel.

Une fonction ne peut en aucun cas modifier les valeurs de ses arguments d'appel [II.14].

FUNCTION nom de la fonction (liste des paramètres de la fonction avec leur type)
RETURN type du paramètre de retour IS
zone de déclaration des variables;
BEGIN
instructions séquentielles;
RETURN nom de la variable de retour ou valeur de retour;
END; [II.16]

Le corps d'une fonction ne peut pas contenir d'instruction **wait**, les variables locales, déclarées dans la fonction, cessent d'exister dès que la fonction se termine.

II.2.5.7.2 Procédures

Une procédure, comme une fonction, peut recevoir du programme appelant des arguments : constantes, variables ou signaux. Mais ces arguments peuvent être déclarés de modes « in », « inout » ou « out » (sauf les constantes qui sont toujours de mode « in »), ce qui autorise une procédure à renvoyer un nombre quelconque de valeurs au programme appelant.

Procedure nom de la procédure (liste des paramètres de la procédure avec leur direction et type)
RETURN IS
zone de déclaration des variables;
BEGIN
instructions séquentielles;
END [nom de la procédure]; [II.16]

Le corps d'une procédure peut contenir une instruction **wait**, les variables locales, déclarées dans la procédure, cessent d'exister dès que la procédure se termine.

Une procédure peut être appelée par une instruction concurrente ou par une instruction séquentielle, mais si l'un de ses arguments est une variable, elle ne peut être appelée que par une instruction séquentielle [II.14].

II.2.6 Différences entre VHDL et un langage de programmation [II.9]

La différence majeure avec un langage informatique ("C" ou le "C++") est la simultanéité des actions décrites. Pour maitriser ce langage et faire la différence avec les autres langages, il faut se familiariser avec la notion d'instruction concurrente et d'instruction séquentielle. Par défaut tout ces instructions sont concurrentes, et pour quelles soient séquentielles il faut l'introduire la notion de "process". Le langage VHDL a pour objectif de décrire l'état de la structure matérielle d'un système car ce matérielle a une structure figée dont l'état logique évolue au cours du temps par contre les autres langages de programmation ont un objectif différent est qui est de décrire l'exécution d'un programme. Mais la différence la plus importante réside dans le fait qu'un programme est séquentiel alors qu'une description matériel est parallèle car c'est un problème d'interconnexion du code .il est possible de décrire du matériel avec un langage de programmation en modifiant la sémantique du langage.

Avec ce langage, un système est structuré en composants qui fonctionnent en parallèle et en permanence. Par contre, un autre langage de programmation est structuré en sous programme qui a une durée d'exécution limité dans le temps avec un début en une fin. Il existe aussi une différence qui concerne le support de l'information : car dans un système matériel les composants sont interconnectés avec des signaux qui sont des connexions permanentes et concurrentes, des sorties de canaux par lesquels transitent les informations. La notion de signal est la base de la description matérielle des structures de donnés qui sont le support de l'information en transit dans le système, car un signal est affecté en permanence. Par contre pour les autres langages de programmation les variables supportent l'information et elles sont affectées de manière ponctuelle dans le temps.

II.2.7 Vérification d'une conception VHDL [II.5]

Il existe différentes méthodes pour développer des systèmes numériques incluent les processus d'optimisation, de vérification et de validation des systèmes. Les tests de développement du système sont principalement effectués en utilisant deux méthodes.

Premièrement la fonctionnalité et les performances du système sont observées et réglées à l'aide d'un logiciel outils de simulation. Le niveau de confiance obtenu à partir de la simulation logicielle est spécifique à l'application et dépend de la disponibilité et de la précision du système.

Deuxièmement, la conception est déployé sur une plate-forme cible pour vérifier la fonctionnalité et mesurer la performance dans des conditions plus réalistes. Cette approche est également connue sous le nom de test de matériel.

II.2.8 Développement d'un projet en VHDL [II.16], [II.17]

Les étapes d'un cycle de design sont les suivantes (figure II.7) :

- Spécification fonctionnelles du projet : descriptions (diagrammes blocs), spécifications et caractéristiques techniques du projet à concevoir.
- Conception en VHDL : descriptions textuelle en langage VHDL du matériel qui doit être implémenté dans le FPGA.
- Simulation fonctionnelle : simulation visant à vérifier la fonctionnalité du design. Elle vérifie si le code VHDL réalise les fonctions requises.
- Synthèse logique : opération de transformation d'une description textuelle (VHDL) d'un comportement en schéma ou « netlist » (liste de nœuds) composée d'instances de cellules élémentaires.
- Simulation fonctionnelle après synthèse : simulation logique servant à vérifier si le code VHDL a été correctement synthétisé et s'il réalise encore les fonctions requises.
- Placement routage : disposition des cellules élémentaires nécessaires à la réalisation des fonctions à satisfaire les contraintes d'occupation de surface.
- Génération du fichier « bitstream » : génération du fichier de configuration du circuit FPGA.
- Programmation du FPGA : pour un FPGA de altéra, c'est le chargement du fichier de configuration « bitstream » ; il est habituellement gardé dans une mémoire PROM qui se trouve sur le même circuit imprimé que le FPGA et qui sert à garder le programme de configuration du FPGA après la mise hors tension du système.

- Test du système : test du design dans son vrai environnement (dans le circuit FPGA) pour s'assurer qu'il réalise encore les fonctions requises tout en respectant les contraintes imposées.

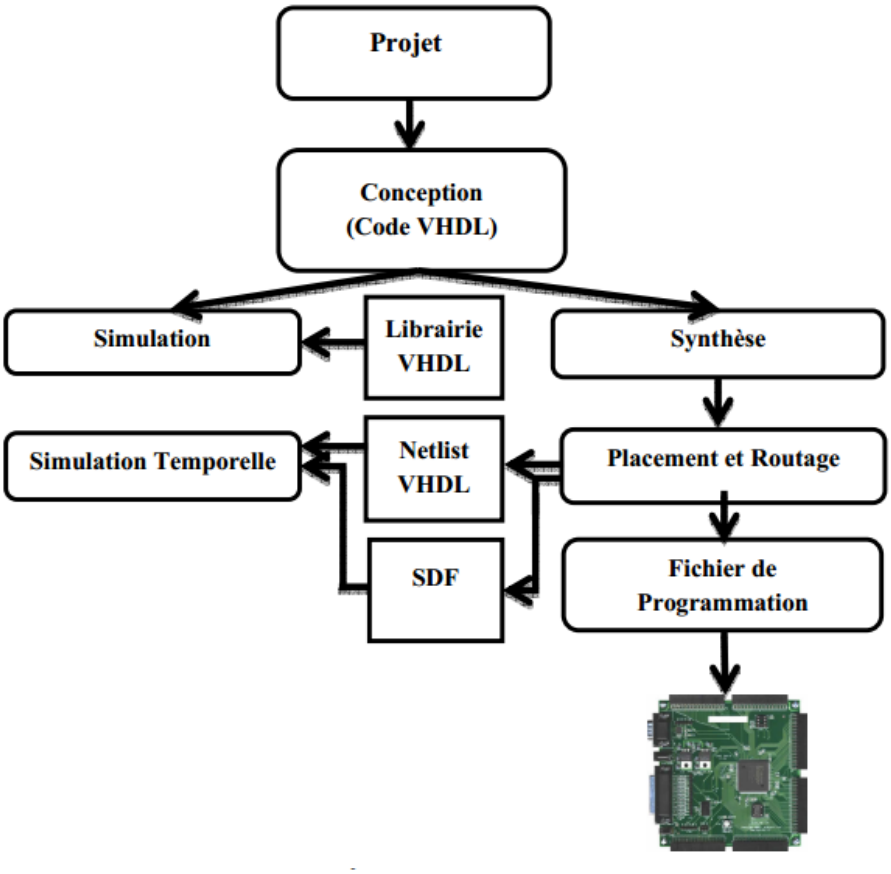


Figure II.7: Différentes étapes de l'implémentation sur FPGA.

II.3 Généralités sur les images

II.3.1 Définition de l'image

Une image est une représentation visuelle voire mentale de quelque chose (objet, être vivant et/ou concept) par la peinture, le dessin, la photographie, le film, etc. [II.18], [II.19], [II.20].

Elle peut être décrite sous la forme d'une fonction I(x, y) de brillance analogique continue, définie dans un domaine borné, où x et y sont les coordonnées spatiales d'un point de l'image et I est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation [II.18], [II.20].

II.3.2 Image numérique

Quand on parle d'image numérique, on parle de pixels. Pixel est la Contraction de l'expression anglaise "Picture éléments": éléments d'image. C'est tout simplement le plus petit élément d'une image, et on peut le figurer comme un carré élémentaire [II.18], [II.20].

Chaque pixel est caractérisé par sa position qui peut être exprimée par deux coordonnées sur l'axe horizontal X et l'axe vertical Y et sa couleur qui représente sa valeur (Pixel (i, j)) (valeur unique, puisque le pixel est considéré comme indivisible) comme le montre la figure ci-dessous.

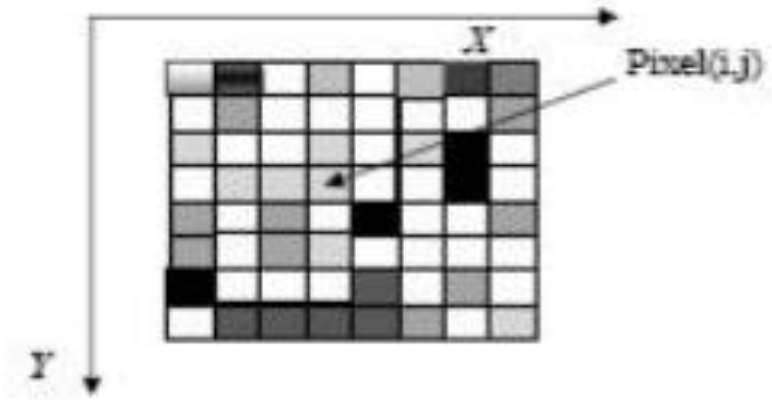


Figure II.8 : Elément d'une image (le pixel).

Donc, Une image numérique est une collection de pixels qui représentent chacun un point de l'image, codé par des valeurs numériques, ces valeurs peuvent être scalaires (images en niveaux de gris), ou bien vectorielles (images couleurs) [II.20], [II.21].

Pour des raisons de commodité de représentation pour l'affichage et l'adressage, les images sont généralement rangées sous forme de tableau (noté I par exemple) de n lignes et p colonnes. Chaque élément I (x, y) représente un pixel de l'image et sa valeur est associé à un niveau de gris codé sur m bits. Dans le cas d'une image couleur, on code celle-ci le plus souvent sur 3couleurs : rouge, vert, bleu (8 bits pour chaque couleur).

II.3.3 Définition de l'espace de couleurs

Un espace colorimétrique ou espace de couleurs associe des nombres aux couleurs visibles. Compte tenu des limites de la vision humaine, ces nombres se présentent généralement sous la forme de triplets [II.22]. Chaque couleur de lumière peut donc être caractérisée par un point dans un espace à trois dimensions. Il y a plusieurs espaces de couleurs qui sont utilisés dans différents domaines. L'espace CIEXYZ sert en photométrie, CIE LUV en visualisation scientifique, CIELAB dans les textiles, RGB pour les moniteurs, CMY pour l'impression, YIQ pour la télévision, HSV, HSI et HLS sont utilisés pour la sélection de couleurs, Munsell en psychologie et Ostwald en peinture [II.23]. Il existe d'autres espaces encore, mais ceux-ci sont les principaux. Par conséquent, certains offrent une palette de couleurs très réduite (l'espace RAL par exemple utilisé en peinture ne dispose que d'environ 1900 couleurs), alors que les plus complets présentent plusieurs millions de couleurs [II.24].

II.3.3.1 L'espace RGB

L'espace RGB (Red, Green, Blue, pour Rouge Vert Bleu, en français RVB), mis au pointent 1931 par la Commission Internationale de l'Eclairage (CIE) consiste à représenter l'espace des couleurs à partir de trois rayonnements monochromatiques de couleurs (rouge, vert et bleu) [II.22], [II.25].

Cet espace de couleurs correspond à la façon dont les couleurs sont généralement codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs [II.22].

Le RGB est un espace de couleur additif (figure II.9) c-à-d, ses trois couleurs primaires (rouge, vert, et bleu) se combinent d'une manière additive pour produire toutes les couleurs désirées [II.24]. Chaque couleur est représentée par un groupe de trois valeurs : une pour le rouge, une pour le vert et une pour le bleu. Chacune de ces valeurs varie dans l'intervalle [0 – 255]. Si ces trois valeurs sont égales à 0, la couleur correspondante est le noir, et le blanc si les trois valeurs sont égales à 255. Lorsque toutes les composantes ont une valeur égale, on obtient une nuance de gris neutre. Si vous vous représentez ce modèle comme des projecteurs, il est facile de

comprendre que des couleurs faibles (valeurs basses) signifie couleurs plus sombres, et couleurs fortes (valeurs élevées) signifie couleurs plus brillantes [II.24]. RGB peut représenter plus de 16 millions de couleurs (Ce qu'on appelle souvent couleurs vraies (True colors)). Ceci concerne les images dont la profondeur est de 24 bits par pixel (8 bits pour chaque composante : 256 intensités). Toutefois, cette valeur n'est que théorique car elle dépend fortement du matériel d'affichage utilisé.

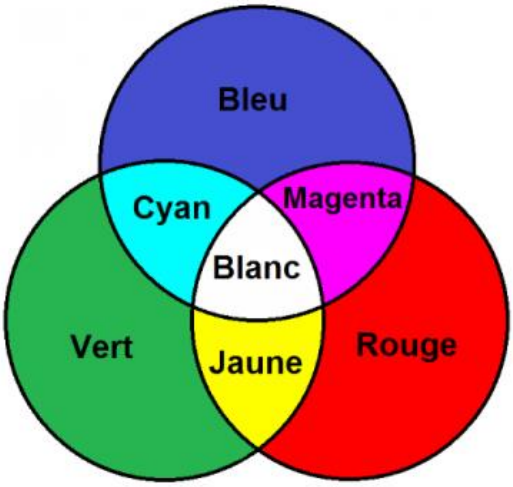


Figure II.9 : Composition additive des couleurs

Le codage RGB repose sur trois composantes proposant la même gamme de valeurs, on le représente généralement graphiquement par un cube dont chacun des axes correspond à une couleur primaire.

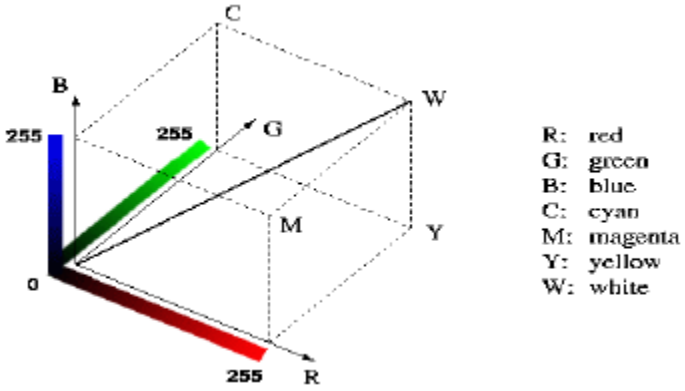


Figure II.10 : Présentation des couleurs dans l'espace RGB.

II.3.4 Mécanique d'affichage d'une image

II.3.4.1 Le signal vidéo VGA

Le nom complet de VGA est Video Graphics Array, qui est une norme pour la transmission vidéo utilisant des signaux analogiques. Les premiers moniteurs CRT ne peuvent recevoir qu'une entrée de signal analogique pour des raisons de conception et de fabrication, de sorte que la carte graphique à l'intérieur de l'ordinateur est responsable de la conversion numérique-analogique, et l'interface VGA est l'interface de sortie des signaux analogiques sur la carte graphique. Bien que les moniteurs LCD puissent recevoir directement des signaux numériques aujourd'hui, afin d'être compatibles avec l'interface VGA de la carte graphique, la plupart d'entre eux prennent également en charge la norme VGA.

II.3.4.2 L'affichage VGA et la génération des signaux de synchronisation de l'image [II.26]

Afficher une image sur un écran consiste à imposer une couleur particulière sur chaque point élémentaire (ou pixel) de l'écran, tandis qu'afficher une séquence vidéo consiste à enchaîner des images rapidement. Pour une image, plutôt que d'afficher tous les points en même temps, il est plus simple de faire un balayage de l'écran à vitesse importante, la persistance des impressions lumineuses sur la rétine de l'œil donnant l'impression d'une image stable.

Le balayage de l'écran sera composé d'un balayage ligne (donc horizontal) associé à un balayage colonne (donc vertical). Ainsi, après avoir balayé toutes les colonnes, de gauche à droite, d'une ligne, on passe à la ligne suivant jusqu'à la dernière ligne en bas, puis on remonte à la première en haut. Pour permettre la synchronisation, après le balayage de chaque ligne il existe un temps mort, et le balayage dure réellement 31,77 µs soit 800 pixels sur une ligne. De même, le balayage complet de l'écran dura 16,67 ms soit 525 lignes au total. On aura donc un rafraîchissement de l'écran à la fréquence de 60 Hz (1/16,67 ms).

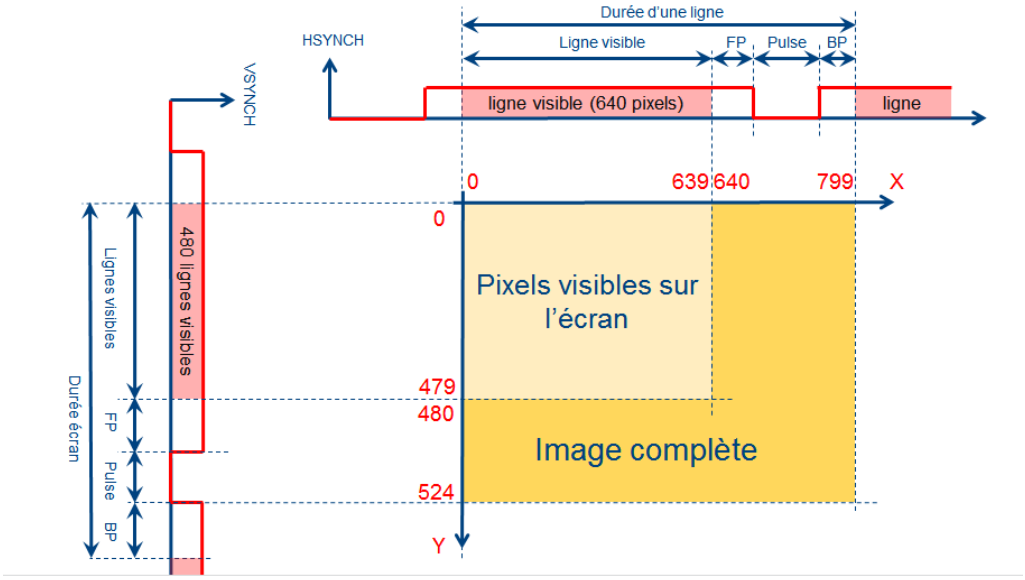


Figure II.11 : Signaux de synchronisation de l'affichage VGA.

La norme VGA 640x480 pixels à 60Hz impose une fréquence d'horloge pour les pixels de 25,175 MHz (cela signifie qu'un pixel a une durée de 1/25175000 s).

La figure ci-dessous donne les différentes durées en nombre de pixels pour la définition d'une ligne, donc du signal HSYNCH, et des durées en nombre de lignes pour la définition du signal VSYNCH.

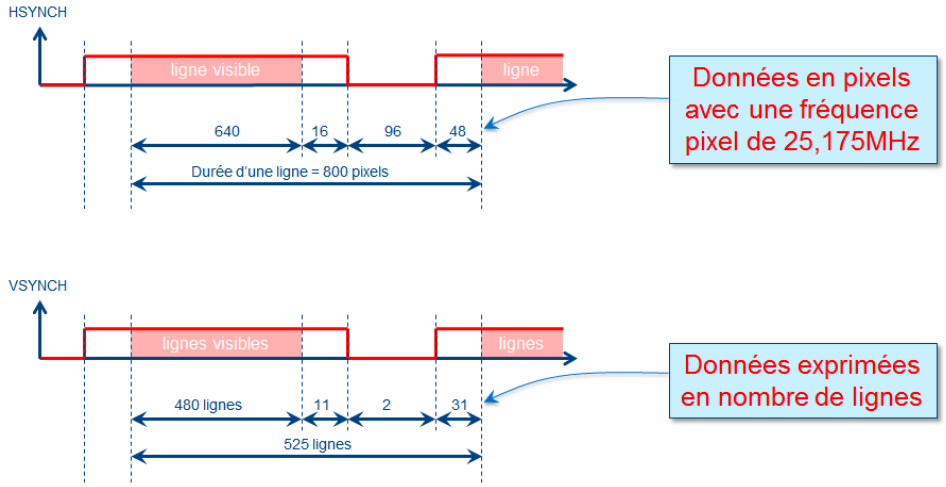


Figure II.12 : Signaux de synchronisation horizontale et verticale de l'affichage VGA.

Dans notre application, une ligne comprendra 640 pixels visibles à l'écran et sera balayée en 24,43 µs. Il y aura 480 lignes (donc 480 pixels) affichées à l'écran, d'où la dénomination 640x480. La couleur de chaque pixel sera la somme pondérée

de trois couleurs élémentaires : rouge, vert et bleu (il s'agit ici de trichromie additive, différente de celle de la palette du peintre qui est sous tractive avec comme couleurs primaires magenta, cyan et jaune). Pour afficher un point blanc, les trois couleurs auront l'intensité maximale, et l'intensité minimale pour afficher un point noir (ou éteindre le pixel).

Les signaux VGA_HSYNC et VGA_VSYNC sont les deux signaux qui vont définir le balayage de l'écran à une fréquence de 60Hz. Le premier signal correspond à la synchronisation horizontale et le deuxième à la synchronisation verticale.

II.3.4.3 Génération d'affichage vidéo VGA à l'aide de FPGA [II.27]

Pour comprendre comment il est possible de générer une image vidéo à l'aide d'une carte FPGA, il faut d'abord comprendre les différentes composantes d'un signal vidéo. Un signal vidéo VGA contient 5 signaux actifs. Deux signaux compatibles avec les niveaux logiques TTL, synchronisation horizontale et synchronisation verticale, sont utilisés pour la synchronisation de la vidéo. Trois signaux analogiques avec des niveaux crête à crête de 0,7 à 1,0 volt sont utilisés pour contrôler la couleur. Les signaux de couleur sont le rouge, le vert et le bleu. Ils sont souvent appelés collectivement les signaux RVB. En changeant les niveaux analogiques des trois signaux RVB, toutes les autres couleurs sont produites.

II.3.4.3.1 Technologie d'affichage vidéo

La première technologie utilisée pour afficher des images vidéo dictait la nature des signaux vidéo. Même si les moniteurs LCD sont maintenant d'usage courant, le composant principal des premiers moniteurs d'ordinateur VGA était le tube cathodique couleur ou tube cathodique illustré à la figure II.13. Le faisceau d'électrons doit être balayé sur l'écran de visualisation dans une séquence de lignes horizontales pour générer une image. La culasse de déviation utilise des champs magnétiques ou électrostatiques pour dévier le faisceau d'électrons vers la position appropriée sur la face du tube cathodique. Les informations de couleur RVB dans le signal vidéo sont utilisées pour contrôler la force du faisceau d'électrons. La lumière

est générée lorsque le faisceau est allumé par un signal vidéo et qu'il frappe un point ou une ligne de phosphore de couleur sur la face du tube cathodique. La face d'un tube cathodique couleur contient une série de rangées avec trois luminophores différents. Un type de phosphore est utilisé pour chacune des couleurs primaires rouge, vert et bleu.

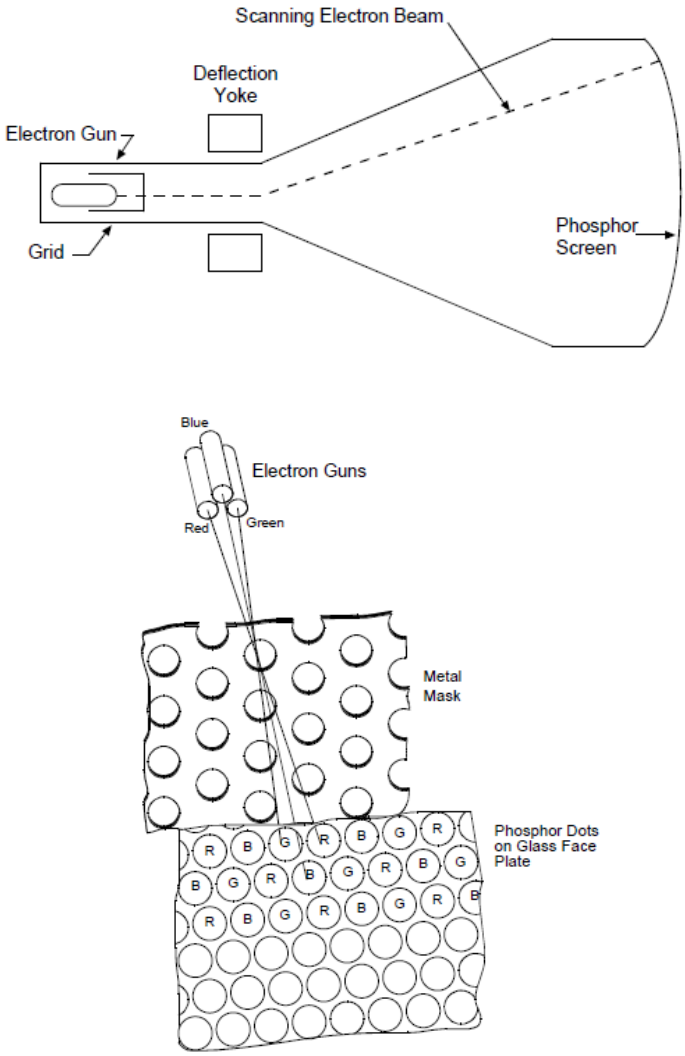


Figure II.13: CRT (Cathode Ray Tube) couleur et points phosphoreux sur la face de l'écran.

Au format VGA standard, comme le montre la figure II.14, l'écran contient 640 x 480 éléments d'image ou pixels. Le signal vidéo doit redessiner l'ensemble de l'écran 60 fois par seconde pour permettre le mouvement dans l'image et réduire le scintillement. Cette période est appelée le taux de rafraîchissement. L'œil humain peut détecter le scintillement à des taux de rafraîchissement inférieurs à 30 à 60 Hz.

Pour réduire le scintillement dû aux interférences des sources d'éclairage fluorescent, des taux de rafraîchissement supérieurs à 60 Hz à environ 70 Hz sont parfois utilisés dans les moniteurs de PC.

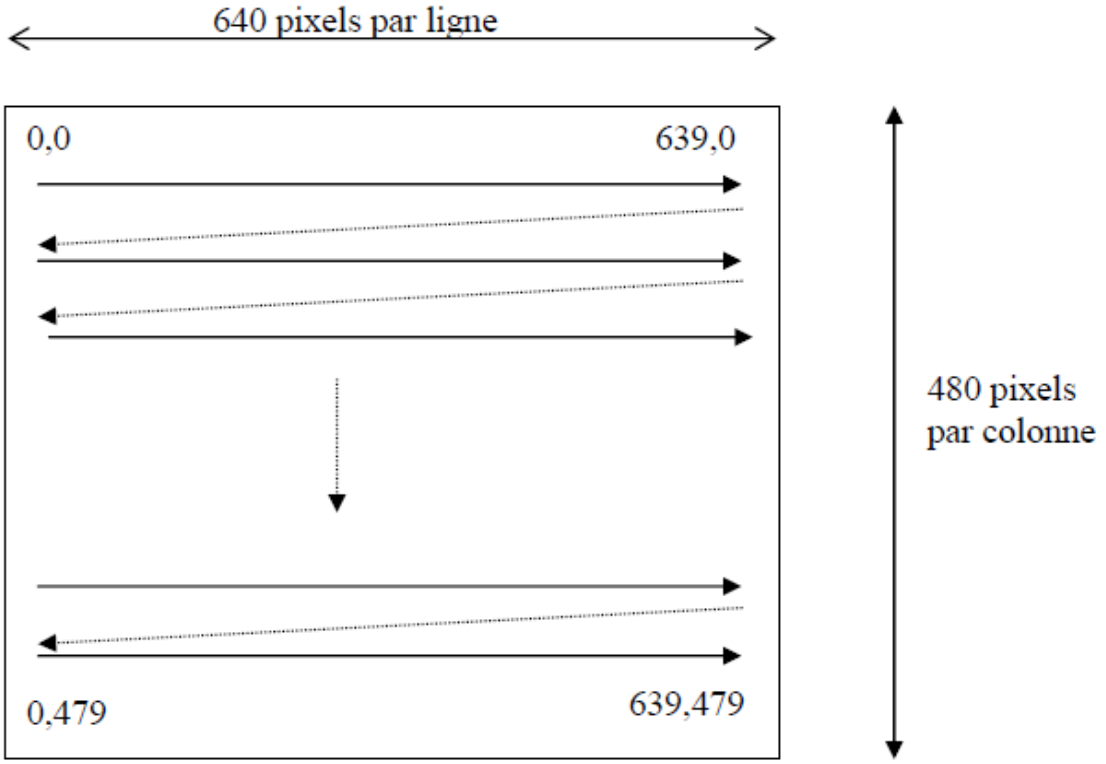


Figure II.14 : Image VGA - Disposition 640 par 480 pixels.

La couleur de chaque pixel est déterminée par la valeur des signaux RVB lorsque le signal balaye chaque pixel. En mode 640 x 480 pixels, avec un taux de rafraîchissement de 60 Hz, cela représente environ 40 ns par pixel. Une horloge à 25 MHz a une période de 40 ns. Une fréquence d'horloge légèrement plus élevée produira un taux de rafraîchissement plus élevé.

II.3.4.3.2 Rafraîchissement de la vidéo

Le processus de rafraîchissement de l'écran illustré à la figure II.14 commence dans le coin supérieur gauche et peint 1 pixel à la fois de gauche à droite. A la fin de la première ligne, la ligne s'incrémente et l'adresse de la colonne est réinitialisée à la première colonne. Chaque ligne est peinte jusqu'à ce que tous les pixels aient été affichés. Une fois que tout l'écran a été peint, le processus de rafraîchissement recommence.

Le signal vidéo peint ou rafraîchit l'image à l'aide du processus suivant. Le signal de synchronisation verticale, comme illustré à la figure II.15, indique au moniteur de commencer à afficher une nouvelle image ou une nouvelle trame, et le moniteur démarre dans le coin supérieur gauche avec le pixel 0,0. Le signal de synchronisation horizontale, comme illustré à la figure II.16, indique au moniteur de rafraîchir une autre ligne de 640 pixels.

Une fois que 480 rangées de pixels ont été rafraîchies avec 480 signaux de synchronisation horizontale, un signal de synchronisation verticale réinitialise le moniteur dans le coin supérieur gauche et le processus se poursuit. Pendant le temps où les données de pixels ne sont pas affichées et que le faisceau revient dans la colonne de gauche pour démarrer un autre balayage horizontal, les signaux RVB doivent tous être réglés sur la couleur noire (tous des zéros).

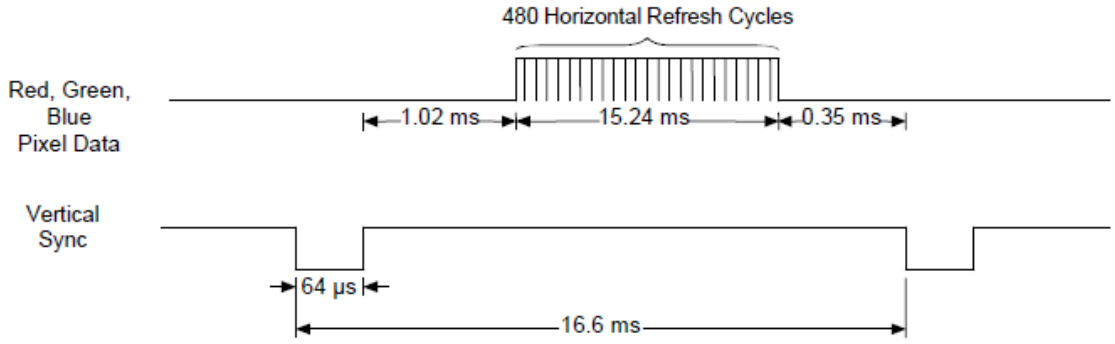


Figure II.15 : Signal de synchronisation verticale de l'affichage VGA pour 640 par 480 à 60 Hz**.**

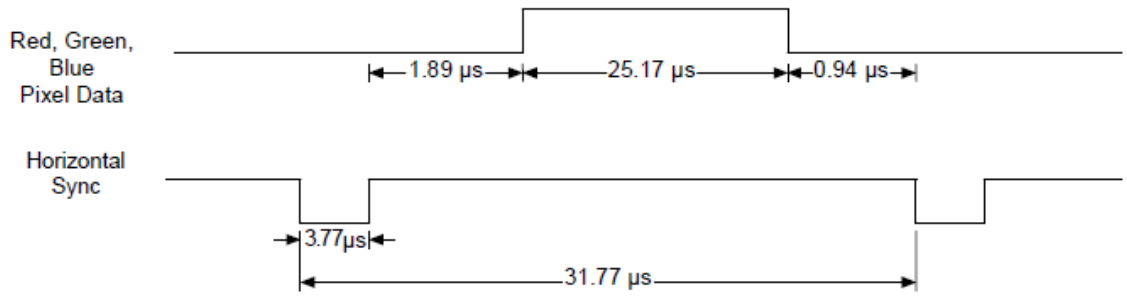


Figure II.16 : Signal de synchronisation horizontale de l'affichage VGA pour 640 par 480 à 60 Hz**.**

De nombreux moniteurs VGA s'éteindront si les deux signaux de synchronisation ne sont pas les valeurs correctes. La plupart des moniteurs PC ont une LED qui est verte lorsqu'elle détecte des signaux de synchronisation valides et jaune lorsqu'elle ne se verrouille pas avec les signaux de synchronisation. Les moniteurs modernes se synchroniseront jusqu'à une plage presque continue de taux de rafraîchissement jusqu'au maximum de leur conception. Dans une carte graphique PC, un emplacement de mémoire vidéo dédié est utilisé pour stocker la valeur de couleur de chaque pixel de l'écran. Cette mémoire est lue pendant que le faisceau balaye l'écran pour produire les signaux RVB. Il n'y a pas assez de mémoire dans les puces FPGA de génération actuelle pour cette approche, donc d'autres techniques seront développées qui nécessitent moins de mémoire.

II.4 La Vision artificielle

II.4.1 Vision et système de vision [II.28]

L'espace qui nous entoure a une structure basé principalement sur les objets. Lorsque l'on demande à une personne de décrire ce qu'elle voit, elle n'éprouve aucune difficulté à discerner et nommer les objets qui l'entourent : téléphone, table, livre, etc. Et pourtant, l'information qui est réellement disponible sur la rétine de ses yeux n'est ni plus ni moins, une collection de points (environ un million !). En chaque point ou pixel (picture element) il y a tout simplement une information qui donne une indication quant à la quantité de lumière et la couleur qui proviennent de l'espace environnant et qui ont été projetées à cet endroit de la rétine. Le téléphone, la table ou le livre n'existent pas sur la rétine. Guidé à la fois par l'information codée dans l'image (ou la rétine) et par ses propres connaissances, le processus visuel construit des percepts. Le téléphone ou le livre sont les réponses finales, résultant d'un processus d'interprétation qui fait partie intégrante du système de vision (figure II.17). De plus, il n'y a pas de correspondance terme à terme entre l'information sensorielle (la lumière et la couleur) et la réponse finale (des objets 3D) [II.28].

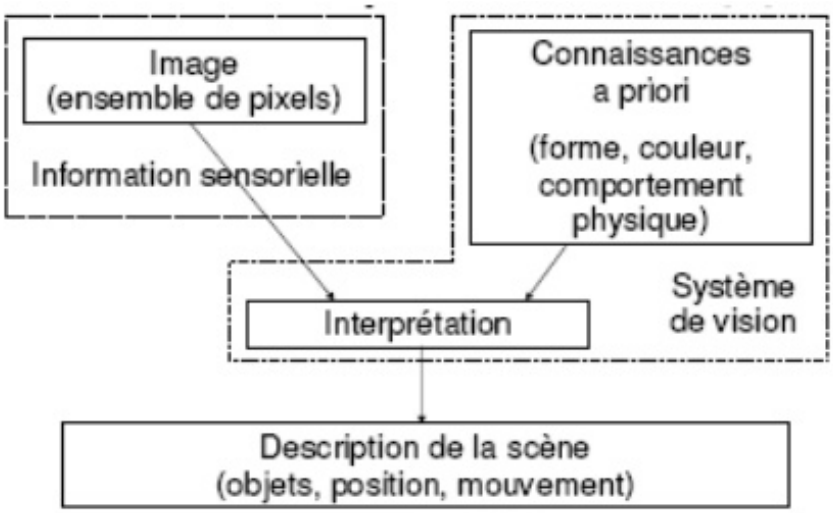


Figure II.17 : Fonctionnement d'un système de vision [II.28].

II.4.2 Définition de la vision artificielle [II.28]

En utilisant un système de vision, la vision artificielle (ou « vision par ordinateur »**)** est cette discipline qui consiste à convertir une image en données objets ou, plus explicitement, à identifier les objets contenus dans l'image par l'extraction et l'analyse de caractéristiques abstraites (features) à partir des pixels, suivant un processus de reconnaissance de forme similaire à celui opéré par l'humain. En effet, ce domaine est une branche de l'intelligence artificielle dont le but est de permettre à une machine de comprendre ce qu'elle «voit» lorsqu'on la connecte à une ou plusieurs caméras.

II.5 Montage de la caméra et le module VGA avec la carte FPGA Cyclone IV

La figure suivante montre comment relier le FPGA de la de la famille cyclone IV de la carte OMDAZZ avec le module de la caméra OV7670 et le module VGA qui sont utilisés pour l'affichage vidéo VGA de notre travail.

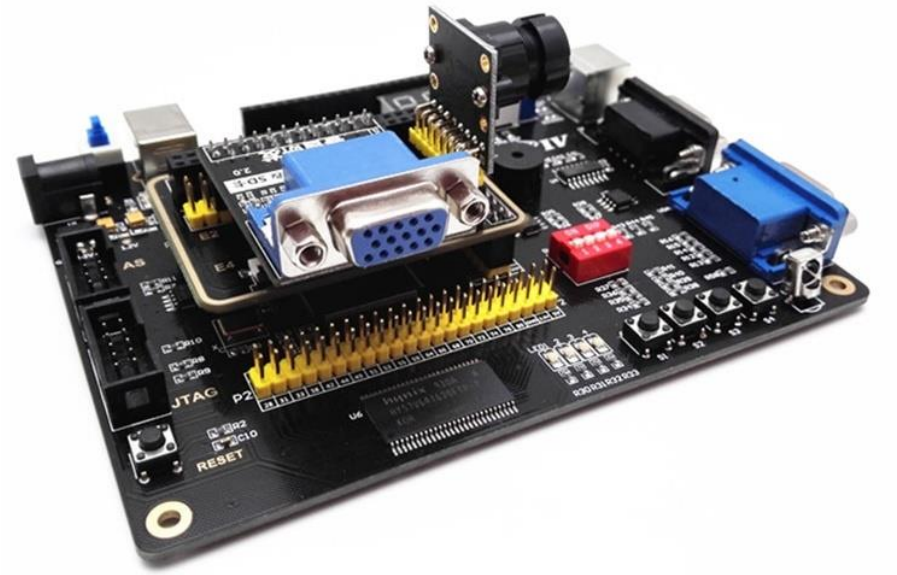


Figure II.18 : Carte FPGA avec les modules OV7670 et VGA.

II.5.1 Carte de connexion

Pour connecter une caméra et un VGA à un FPGA de notre carte, nous aurons besoin d'une carte de connexion, comme le montre la figure II.19.

Figure II.19 : La carte de connexion

II.5.2 Module VGA

Il permet de relier le système générateur de vidéo (le plus souvent l'ordinateur) au moniteur.

- 16 couleurs vraies, 65536 couleurs disponibles
- Le module VGA avec prise de carte SD

Figure II.20 : Connecteur VGA.

Le schéma de VGA et comment le connecter au FPGA est donné sur la figure suivante :

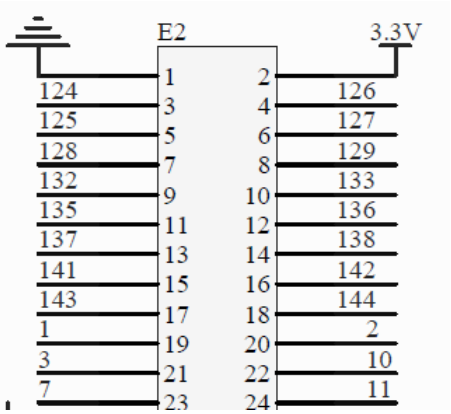


Figure II.21 : Connectique VGA d'une FPGA Cyclone IV [II.29].

Ce module VGA peut être connecté à l'écran VGA, via la caméra pour capturer des images, afficher des images sur l'écran.

II.5.3 Module caméra OV7670 [II.30]

La caméra OV7670 est une petite caméra CMOS qui offre des images en VGA. Parmi ses caractéristiques, on compte sa facilité de connexion et sa faible consommation d'énergie, permettant ainsi l'affichage des images capturées sur un écran VGA. Elle offre toutes les fonctions de la caméra mono-puce VGA de 30 images/s et du processor d'image. Ce projet implique l'analyse des signaux visuels

de la caméra OV7670, leur conversion dans un format approprié pour l'affichage sur un écran VGA. Pour ce faire, nous avons besoin de logiciels et de circuits nécessaires dans le FPGA pour atteindre cet objectif. De plus, ce processus nécessitera l'identification du protocole de communication approprié entre la caméra et le FPGA. Parmi ses caractéristiques nous pouvons citer :

- Taille de la planche: W27mm x L41mm
- Réseau photosensible 640X480
- Tension IO 2.5V à 3.3V (LDO interne 1.8V)
- Consommation d'énergie 60mW/15fps VGA YUV
- Température de fonctionnement-30 ° C à 70 ° C
- Fonctionnement stable 0 ° C à 50 ° C
- Format de sortie (8 bits) YUV/YCbCr4:2:2 RGB565/555/444 GRB4:2:2 Données RVB brutes
- Taille optique 1/6"
- Angle de vue 25 °
- Taux de conversion maximal 30fps VGA
- Rapport signal/bruit 46 dB
- Parcourir le mode ligne par ligne
- Exposition électronique 1 ligne à 510 lignes
- Zone de pixels 3.6 µm x 3.6 µm

Figure II.22 : Caméra OV7670 utilisé dans notre réalisation pratique.

Le schéma de la caméra et comment la connecter au FPGA est illustré sur la figure II.23.

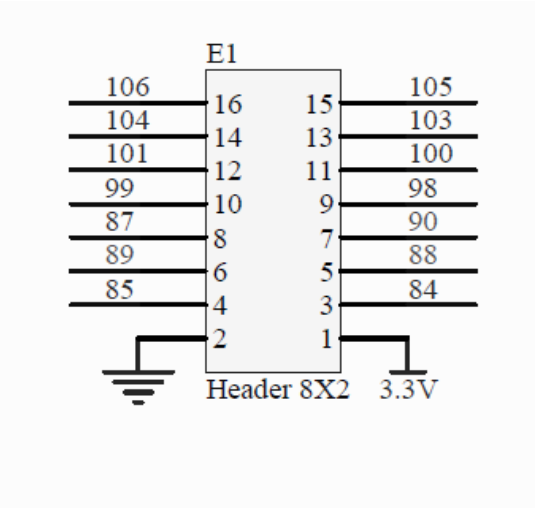


Figure II.23 : Schéma de la caméra et comment la connecter au FPGA [II.29].

De plus, il y a RST, que nous connectons au PIN 3V3, et PWDN au PIN GND.

Le tableau II.3 représente les pins et leurs fonctions de la caméra OV7670.

Tableau II.3: Brochage de la caméra OV7670 [II.30].

PIN	Fonction
3V3	Positive power supply pin
GND	Ground pin
SDIOC	Serial clock
SDIOD	Serial Data
VSYNC	Vertical Sync
HREF	Horizontal Sync
PCLK	Pixel clock output
XCLK	System clock input
D7	Video parallel output bit 7
D6	Video parallel output bit 6
D5	Video parallel output bit 5
D4	Video parallel output bit 4
D3	Video parallel output bit 3
D2	Video parallel output bit 2
D1	Video parallel output bit 1
D0	Video parallel output bit 0
RESET	Reset (active low) pin
PWDN	Power down (active high) pin

II.6 Conclusion

Dans ce deuxième chapitre on a pu faire une présentation détaillée du langage VHDL, ainsi qu'une petite comparaison avec les langages informatiques, et ces avantages , caractéristique , élément de base….etc. et pour enfin conclure cette première partie de ce chapitre on fini avec les étapes développement d'un projet en VHDL pour enfin donné l'ouverture et passer a la phase suivante (pratique) qui consiste a réaliser des programmes VHDL fonctionnel qui nous permis d'afficher des signaux vidéo VGA, en utilisant l'outil de développement Quartus II .

La deuxième partie de ce chapitre a été consacré à quelques concepts sur l'image et à la présentation de signal vidéo VGA et la vision artificielle.

En effet, l'implémentation des FPGA se fait au moyen de langages de description matérielle, dans le chapitre suivant nous allons programmer un FPGA en VHDL afin de lui permettre de commander un écran VGA pour obtenir un affichage en 640 par 480 pixels.

REFERENCES BIBLIOGRAPHIQUES -CHAPITRE II-

[II.1] Céline Guilleminot, « étude et intégration Numérique d'un système multicapteurs amrc de Télécommunication basé sur un prototype virtuel utilisant Le langage de haut niveau vhdl-ams » thèse Doctorat, Université de Toulouse II 01 décembre 2005.

[II.2] Zerari Houssam, « Analyse et synthèse d'un UART en VHDL » Mémoire de fin d'études en vue de l'obtention du diplôme master en Electronique Option Signaux et communication, université Mohamed Khider –Biskra, 2013.

[II.3] NACHEF TOUFIK, «Implantation d'une instruction sur un FPGA», Pour obtenir le diplôme Magister en Electronique, Université MOULOUD MAMMERI DE TIZI OUZOU, 2011.

[II.4] DAHOUN MOHAMED NABIL ET BOUFADINA ILYES, « conception et implémentation d'un micro contrôleur de 16 bits sur un FPGA » mémoire de fin d'étude pour l'obtention du diplôme de master en électronique instrumentation université Dr TAHAR MOULAY, SAIDA 2020.

[II.5] MADANI MAROUA, « Implémentation d'un algorithme MPPT dans une FPGA utilisant la carte ML605 » en vue l'obtention du diplôme de master en électronique option électronique de système embraqué université BOUDIAF, MSSILA 2018.

[II.6] J. Madsen, J. Grode, p. Knudsen, m. Petersen et a. Haxthausen { lycos: then lyngby co-synthesis system , design automation for embedded Systems, vol. 2, depatment of information technology, technical university of Denmark, 1997.

[II.7] Messerli, Etienne, "Manuel VHDL", 2007.

[II.8] https://sti.discip.ac-caen.fr/IMG/pdf/vhdl.pdfSite consulté le (17/09/2020).

[II.9] AFETTOUCHE Malik, « Commande d'un robot en position à base d'une arte FPGA», Mémoire master, université Mouloud Mammeri de tizi ouzou2013.

[II.10] Denis Rabasté,IUFM d'Aix-Marseille,« programmation des CPLD et FPGA en VHDL avec Quartus II »
http://genelaix.free.tf/IMG/pdf/aide_VHDL_quartus.pdf

[II.11] http://hdl.telecom-paristech.fr/vhdl_structurel.html. COURS EN LIGNE VHDL (site consulté le 16/02/2019).

[II.12] AOUICHE Mounir, « Conception et implémentation d'un Microcontrôleur de 64 bits sur FPGA», Mémoire master en Réseaux et télécommunications, Université Mohamed Khider Biskra, 2019.

[II.13] Philippe LARCHER, « VHDL Introduction à la synthèse logique », 2000 .

[II.14] WEBER, Jacques et MEAUDRE, Maurice, « *Circuits numériques et synthèse logique: un outil: VHDL* », Jacques Weber, 1995.

[II.15] MICHEL AUMIAUX « Initiation au langage VHDL » 2ème édition, Docteur ès sciences Professeur à l'école supérieure d'électronique de l'ouest (Angers), Dunod, Paris, 1999.

[II.16] BenlakehalImen, « Implantation des réseaux de neurones sur un circuit reconfigurable de type FPGA», mémoire Présenté pour l'Obtention du Diplôme de Master Académique Option Informatique Industrielle, Université Larbi Ben Mhidi D'Oum El Bouaghi, 2017.

[II.17] Khalid BENSADEK, « développement d'un modèle VHDL Synthétisable d'un décodeur de VITERBI », mémoire de master en Génie Electrique, école de technologie supérieur, université du QUEBEC, 2004.
[II.18] A. Abdelaziz, "Compression d'images par fractale", Thèse de magister en informatique, Université de Batna, 2002.
[II.19] D. Zeroual, "Implémentation d'un environnement parallèle pour la compression d'images a l'aide des fractales", Thèse de magister en informatique, Université de Batna, 2006
[II.20]BOUCETTA ALDJIA,"Etude de l'effet des Transformées de Décorrélation en Compression des Images Couleurs RGB" mémoire de fin d'étude pour l'obtention du diplôme de Magister en informatique OPTION Ingénierie des systèmes informatique, UNIVERSITE DE BATNA, 2010.
[II.21] A. Manzanera, " Les images numériques", cours traitement et reconnaissance d'image, Master IAD, Ecole Nationale Supérieure de Techniques Avancées/Unité d'Électronique et d'Informatique, Université Pierre et Marie CURIE, Paris 2005.
[II.22] C. Bencheriet, A. Boualleg, H. Tebbikh, « Segmentation de la Couleur de Peau par Seuillage Selon Différents Espaces de Couleur ", JIG'2007-3èmes Journées Internationales sur l'Informatique Graphique, Université 8 Mai 45de Guelma, 2007.
[II.23] J. C. Russ, "The image processing –Handbook-", Third Edition. CRC Press, CRC Press LLC, 1998.
[II.24] V. Risson, "Application de la Morphologie Mathématique à l'Analyse des Conditions d'Éclairage des Images Couleur", Thèse de doctorat, Ecole des Mines de Paris, 17 Décembre 2001.
[II.25] R. Heus, "Approches virtuelles dédiées à la technologie des puces à tissus «Tissue Micro Arrays » TMA : Application à l'étude de la transformation tumorale du tissu colorectal", Thèse de doctorat, Université Joseph Fourier, 28 Septembre 2009
[II.26] http://denis.rabaste.free.fr/ressources/FPGA/9_cde_vga_quartus.pdf

[II.27] J.O. HAMBLEN, T.S. HALL and M.D. FURMAN, «Rapid prototyping of digital systèmes», SOPC EDITION, 2007.
[II.28]Meziani Imane, «Un système de Navigation 3D» Mémoire master en Génie Informatique Université IBN KHALDOUN, Tiaret 2018.
[II.29]https://github.com/jvitkauskas/Altera-Cyclone-IV-board-V3.0/blob/master/Hardware%20documentation/expansion%20board%20schematics.pdf
[II.30] BOUTIOUTA Mohammed Islam, «Nœud de capteur d'images sans fil à faible coût énergétique» Mémoire master en Télécommunications Université BADJI MOKHTAR ANNABA, 2019.

Chapitre III

Commande d'un écran VGA par un circuit FPGA Cyclone IV

III.1 Introduction

Video Graphics Array (**VGA**) est un standard d'affichage pour ordinateurs lancé en avril 1987 par IBM avec la mise en marché de la gamme PS/2, en même temps que le MCGA, en tant qu'amélioration des standards EGA (640 × 350) et CGA. VGA appartient à une famille de standards d'IBM et reste compatible avec les précédents formats [III.1].

Comme d'autres réalisations d'IBM, VGA a été très largement cloné par d'autres fabricants. Bien que de moindre résolution que le standard XGA (1024 × 768) lancé en même temps par ce fabricant, c'est le dernier standard IBM que la majorité des constructeurs a décidé de suivre pour les architectures PC. Un consensus se fit rapidement ensuite sur le standard SVGA 800 × 600 [III.1].

Le terme VGA désigne aussi bien un *mode d'affichage* (640 × 480, etc.) qu'une *connectique* (connecteur VGA).

Nous allons au cours de ce chapitre programmer un FPGA en VHDL afin de lui permettre de commander un écran VGA pour obtenir des affichages des messages caractères (en 640 par 480 pixels) et des images en temps réel (en 160 par 120 pixels).

Dans un premier temps nous allons fais une conception vidéo basée sur des caractères, nous nous contenterons d'afficher un message WELCOME JURY sur un moniteur VGA, puis nous allons programmer notre FPGA en utilisant toujours le langage de description matériel le VHDL pour mesurer et afficher la distance capté par un capteur ultrasonique d'un obstacle.

Dans les deux parties de ce travail nous allons interfacer une caméra CMOS OV7670 de résolution VGA avec la carte de développement OMDAZZ pour transmettre un flux des images en direct acquis sur un moniteur VGA.

La carte de développement OMDAZZ que nous utiliserons contient un FPGA de la famille Cyclone IV d'Altera, le circuit est EP4CE10E22C8 dont les sorties sont reliées à un connecteur pour l'écran VGA.

III.2 Plateforme de développement QuartusII

Quartus est un logiciel développé par la société Altera, permettent 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 afin de réaliser une simulation, une synthèse et une implémentation sur cible reprogrammable.

En ce qui suit un tutorial des principales fonctionnalités nécessaires à la création d'un projet, son analyse et en fin son implémentation sur l'FPGA. Ce tutorial est construit avec des tables explicatives contenant des captures d'écran ainsi que des instructions à suivre.

III.2.1 Déroulement de la conception

L'implémentation d'un circuit FPGA sous Quartus suit les étapes suivantes :

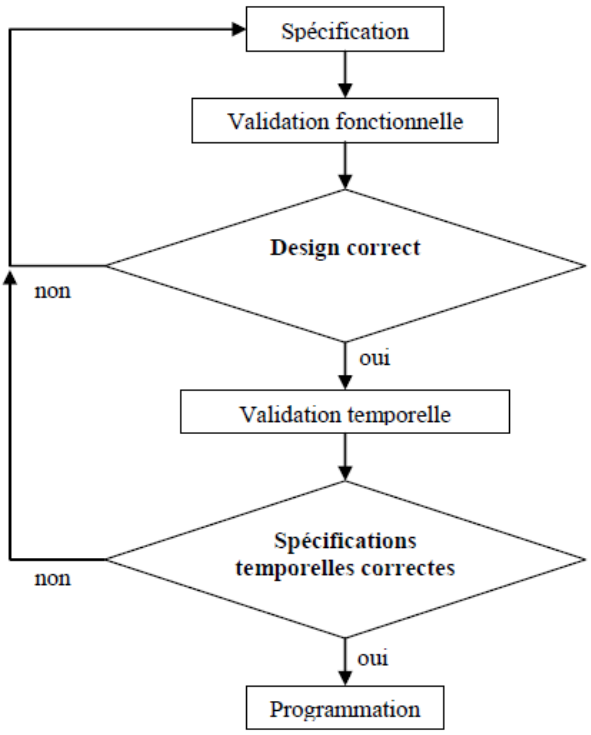


Figure III.1: Déroulement de la conception.

- ✓ **Spécification :** saisie du circuit logique (spécification syntaxique : VHDL, Verilog, mode graphique, etc.).
- ✓ **Validation fonctionnelle :** simulation fonctionnelle du concept (problèmes des sorites/sorties, boucles, etc.). Vérification ne prenant pas en compte les aspects temporels du circuit.

- ✓ **Validation temporelle :** simulation temporelle (et fonctionnelle) du circuit (temps de propagation, recouvrement de signaux, etc.).
- ✓ **Implémentation :** le programme est porté physiquement sur le circuit FPGA en fonction des spécifications précisé par le programmeur (pins, etc.)

III.2.2 Présentation [III.2]

L'environnement de Quartus II est le suivant **:**

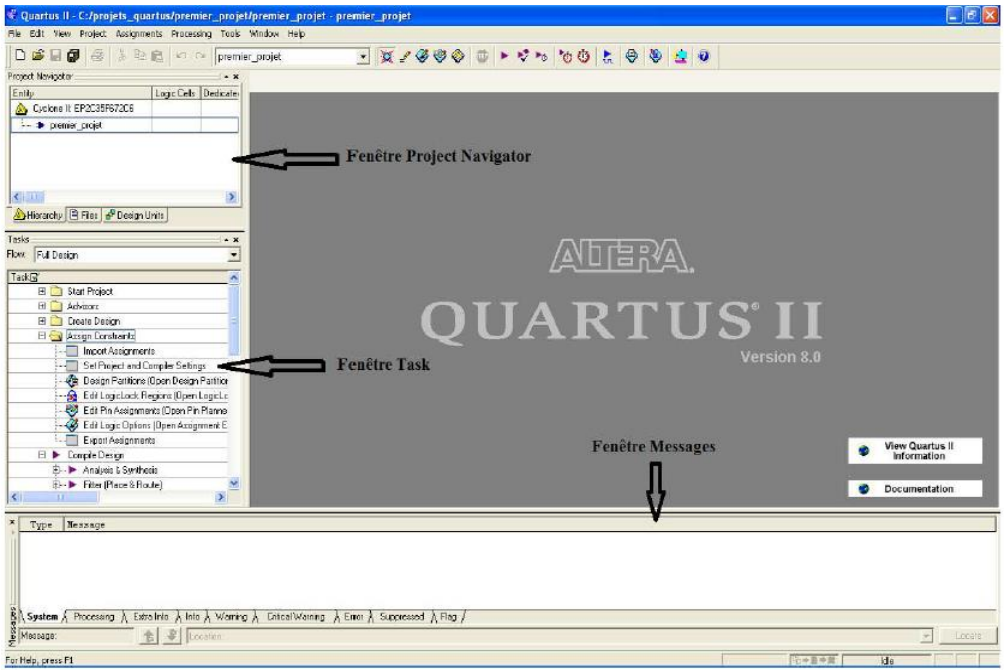


Figure III.2 : Environnement de développement Quartus II.

L'environnement se découd en 4 parties, une pour l'éditeur de code qui apparait après avoir créé un nouveau projet ; 3 sont visibles après lancement de Quartus.

- ➢ **une fenêtre projet Navigator** : qui contient tous les informations relatives au projet dont les détails sur la hiérarchie du projet, les fichiers le constituant, les différents unités définies (entités, architecture, schémas, machines d'états …..).
- ➢ **la fenêtre Tasks :** présente les différentes tâches du design Flow avec la possibilité d'accéder à l'ensemble des processus de traitement de Quartus et les comptes rendus associés.
- ➢ **la fenêtre Messages :** permet d'être informé en continu des informations, avertissements et erreurs apparaissant lors de l'exécution des diverses taches.

III.2.3 Création d'un projet

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).

Après avoir démarré Quartus, on crée un nouveau projet **:*File → New Projet…***

- La première boite de dialogue qui apparait permet de fixer le nom du projet et son lieu de stockage :

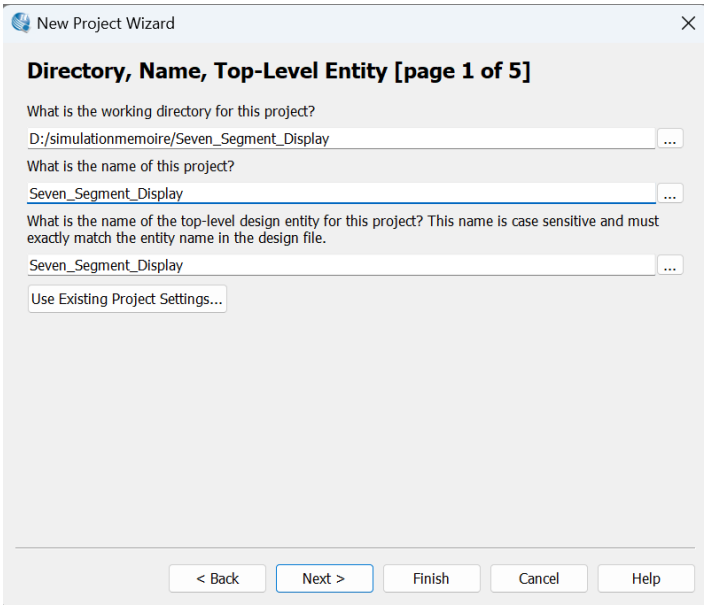


Figure III.3 : Création d'un projet sous quartus II.

- dans la seconde, il est possible d'ajouter des fichiers déjà existants au projet. Ne rie ajouter et cliquer sur le bouton suivant.
- la troisième fenêtre de dialogue permet de choisir le FPGA ciblé.
- Une fois choisi, on en termine avec l'assistant de création de projet en cliquant sur Finish.

La réalisation sous Quartus amène à l'écriture de 3 sources VHDL et d'un schéma hiérarchique.

Dans notre cas, nous choisirons un FPGA de la famille **Cyclone IV** (figure III.4) à savoir le circuit **EP4CE10E22C8** qui est intégré dans la carte de développement OMDAZZ.

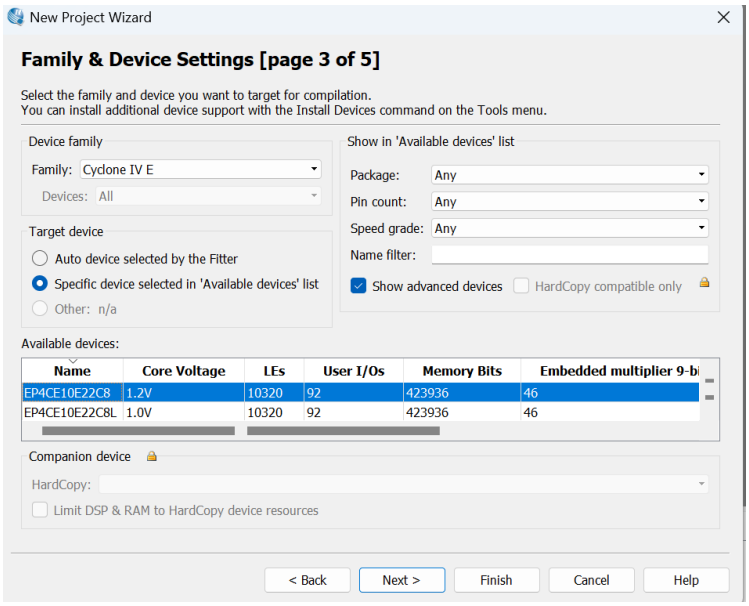


Figure III.4 : Fenêtre de choix du circuit.

Quand la fenêtre ***EDA→Tool→Settings*** apparait cliquer sur ***Next.*** Une fenêtre récapitulative apparait :

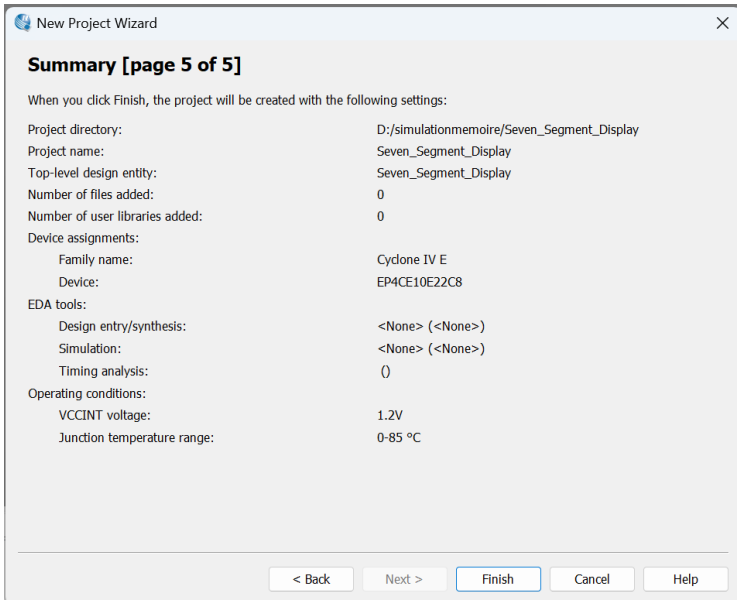


Figure III.5 : Fenêtre de création du projet finie.

Valider les choix par ***Finish*** ou bien faire ***Back*** pour des modifications éventuelles. Dans le navigateur de projet, un onglet avec le type composant et l'entité maitre apparait :

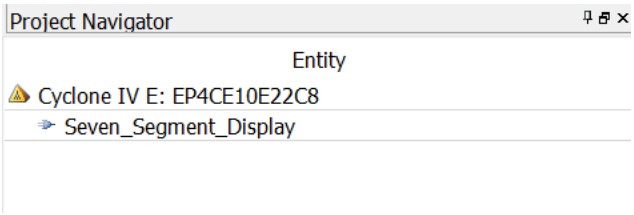


Figure III. 6 : Fenêtre de type de projet.

III.2.4 Ajout d'une source VHDL

Pour ajouter une nouvelle source vhdl : ***file → New*** … eton choisit une source de type VHDL.

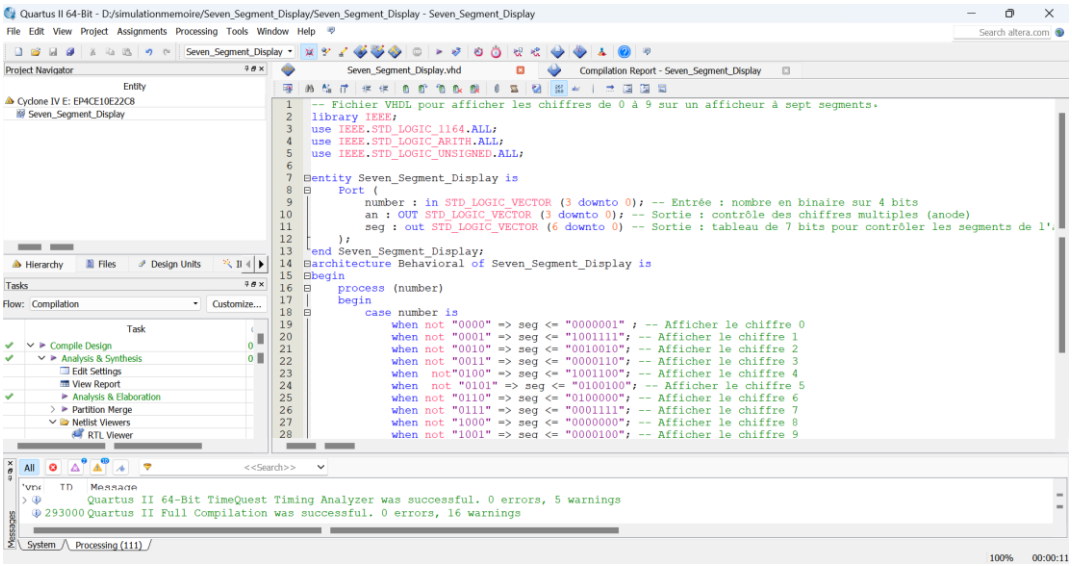


Figure III.7 : Fenêtre de Seven_Segment_Display.vhdl.

III.2.5 Sauvegarde d'un projet

Après avoir tapé le code, la sauvegarde se fait comme suite : ***file→ Save As*** …. Puis entrer nom **Seven_Segment_Display** avec l'extension. **vhd**).

III.2.6 Compilation

Après la création du projet avec l'entité « **Seven_Segment_Display** » et création du fichier « **Seven_Segment_Display.vhdl** » on effectue la compilation afin de vérifier que la description donnée est correcte syntaxiquement et élaborer le circuit. Pour ce faire, aller à ***Processing→Start compilation***.

En fin de compilation on a reçu un message qui confirme le succès de l'opération à compilé comme montre la figure III.8.

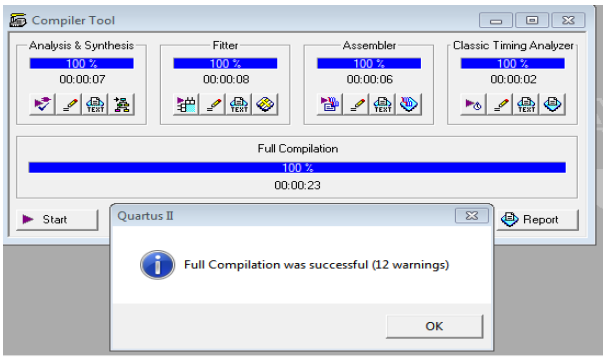


Figure III.8 *:* Fenêtre de compilation projet « Seven_Segment_Display ».

Le succès de compilation permet de savoir qu'il n'ya pas d'erreurs dans la description VHDL mais le fonctionnement doit être certifié par la simulation pour conformer le fonctionnement exact de circuit.

III.2.7 Schéma fonctionnel (RTL : Register Transfer Level)

Le schéma fonctionnel définit les entrées et les sorties du module de décodeur décrit en VHDL.

S'il n'y a pas d'erreur, il faut vérifier le circuit élaboré en consultant son schéma à travers le menu ***Tool → netlist viewer → RTL viewer*** comme la montre la figure ci-après**.**

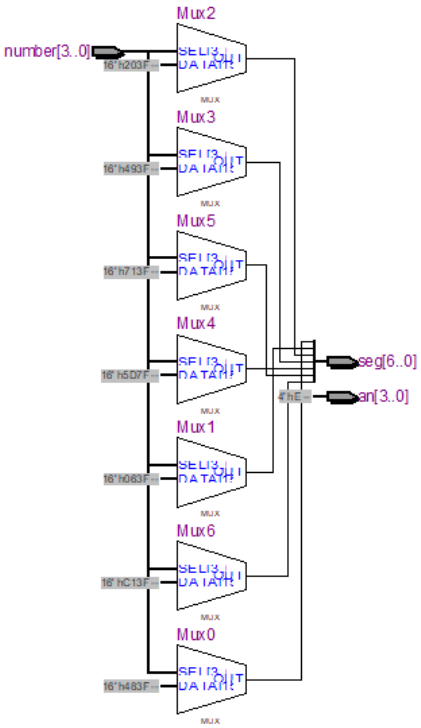


Figure III.9 : Schéma fonctionnel RTL de Seven_Segment_Display.

III.2.8 Création d'un symbole graphique du VHDL

Pour créer un symbole graphique pour la description VHDL du décodeur en peut suivre la figure III.10, ensuite nous ouvrons une feuille graphique (« *Block diagram /Schématic File est* insérer dans la description générale le décodeur qui est maintenant un symbole du répertoire projet. La figure III.11 représente le symbole graphique de décodeur.

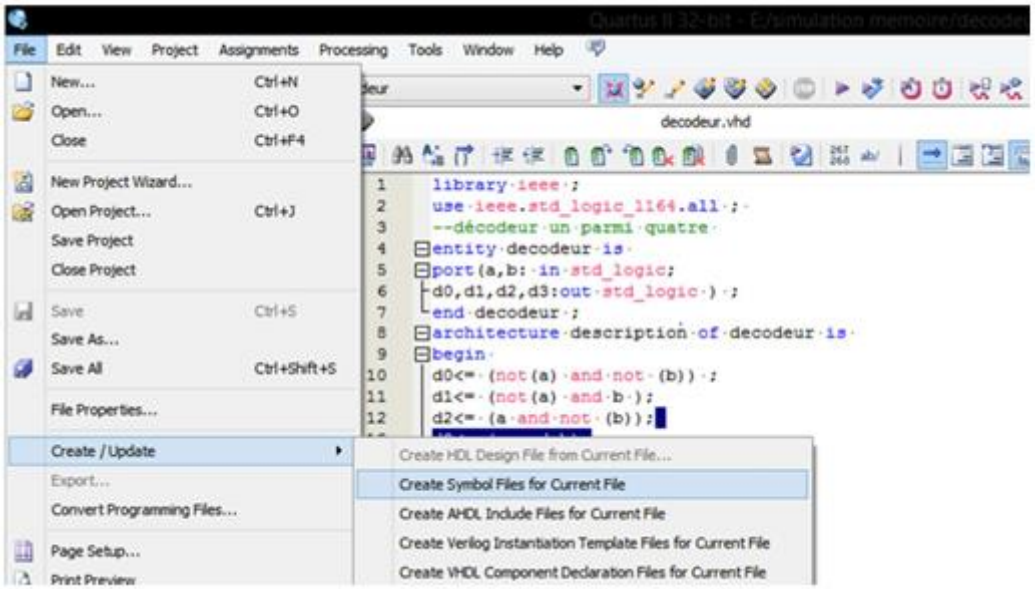


Figure III.10 : Création d'un symbole graphique de Seven_Segment_Display.

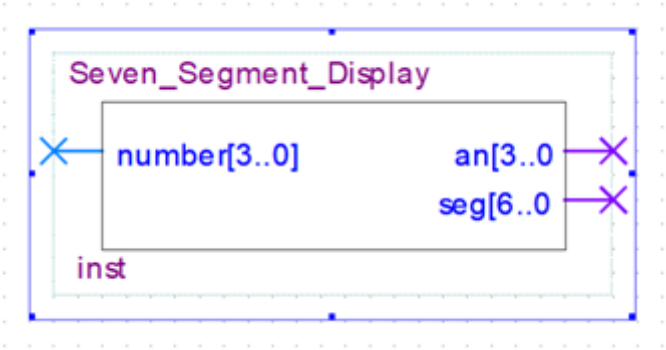


Figure III.11 : Symbole graphique de Seven_Segment_Display.

III.3 Programmation de la maquette Cyclone IV

C'est l'étape utime. Pour cela, il faut assigner les pins d'entrée/sortie du design aux broches du circuit physique.

III.3.1 Affectation des pins

Afin de choisir quelle broche physique du circuit doit etre connectée, lancer l'outil d'assignement de pins par : ***Assignement → Pins***

Dans la fenetre correspondante, il est possible de choisir différents types d'assignement.

La liste des broches utilisables pour le FPGA et sortant sur les connecteurs est donnée dans le manuel de la carte Cyclone IV [III.3] de la série OMDAZZ qui contient un circuit FPGA d'Altera. Les numéros des broches d'entrées, sortie et d'horloge sont données dans le tableau III.1.

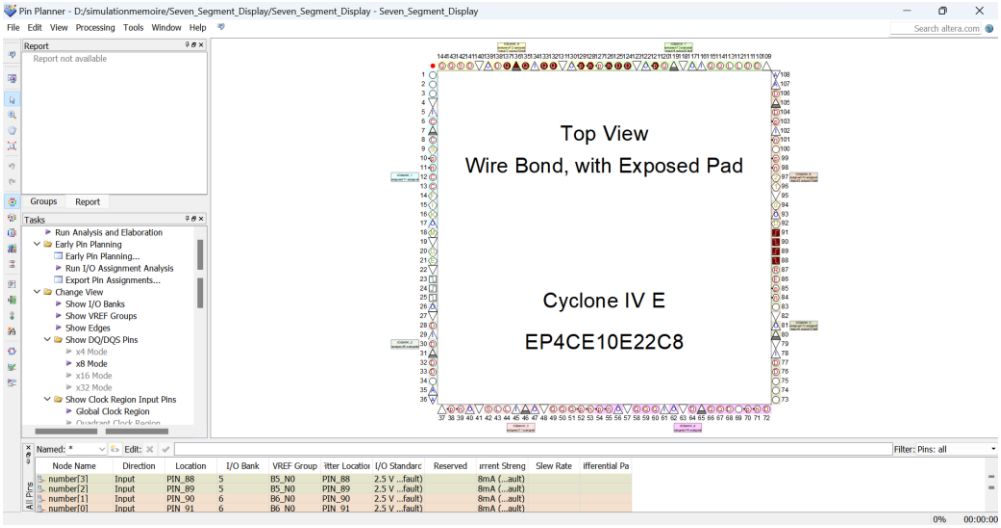


Figure III.12 : Fenêtre de type d'assignement des pins.

Tableau III.1 : Numéros des pins entrées/sorties utilisés [III.3].

Entrées	Numéro des pins	Sorties	Numéro des pins	Sorties	Numéro des pins
number [3]	Pin_88	an[3]	PIN_137	seg[4]	PIN_125
number [2]	Pin_89	an[2]	PIN_136	seg[3]	PIN_129
number [1]	Pin_90	an[1]	PIN_135	seg[2]	PIN_132
number [0]	Pin_91	an[0]	PIN_133	seg[1]	PIN_126
		seg[6]	PIN_128	seg[0]	PIN_124
		seg[5]	PIN_121		

III.3.2 Etude de Floorplan

Floorplan Editor donne accès à l'éditeur de Floorplan qui permet de visualiser l'implantation physique des fonctions dans la structure du circuit ALTERA choisi.

Cet éditeur se présente comme le montre la figure III.13 :

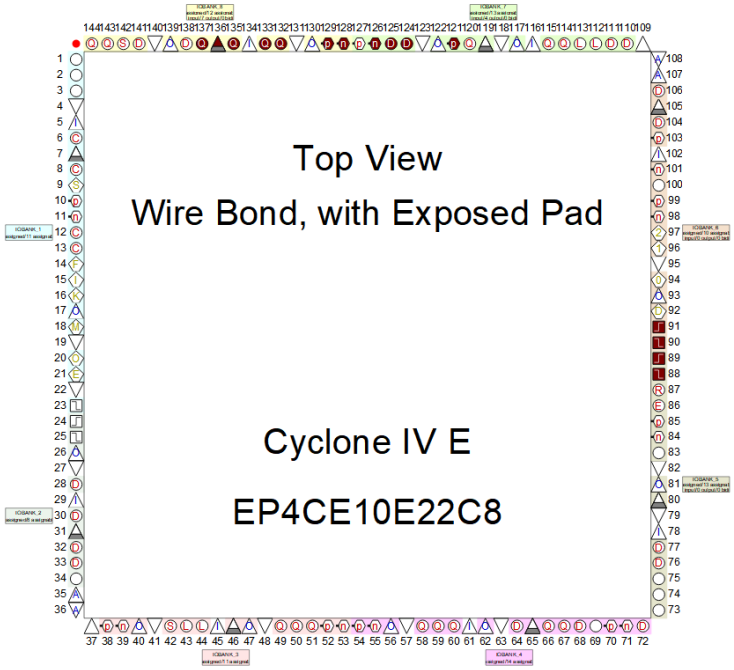


Figure III.13 : Plan de cablage du FPGA.

III.3.3 Programmation du circuit

Il est alors possible de programmer le FPGA de la maquette. Le câble de programmation associé doit être relié au port USB de l'ordinateur par son entrée BLASTER et la maquette doit être alimentée. Elle doit aussi être configurée pour permettre la programmation du composant souhaité. La programmation du circuit se fait via le protocole JTAG (Joint Test Action Group). Pour cela vérifier que les connections entre le PC (port USB) et la carte via le module USB-Blaster sont opérationnelles.

Lors de la compilation un fichier binaire **decodeur.sof** a été généré (le format SOF pour les FPGA et le format POF pour les CPLD). C'est ce fichier que nous allons envoyer dans le circuit EP4CE10E22C8 de la famille Cyclone IV de notre FPGA via le port USB du PC. Il suffit alors de lancer le programmateur ***Tools***

→Programmer, puis cliquer sur le bouton ***AUTODECTECT.*** Vérifier que le **. Sof** est bien la et que la case ***Program→Configure*** est cochée, puis cliquer sur ***Start.***

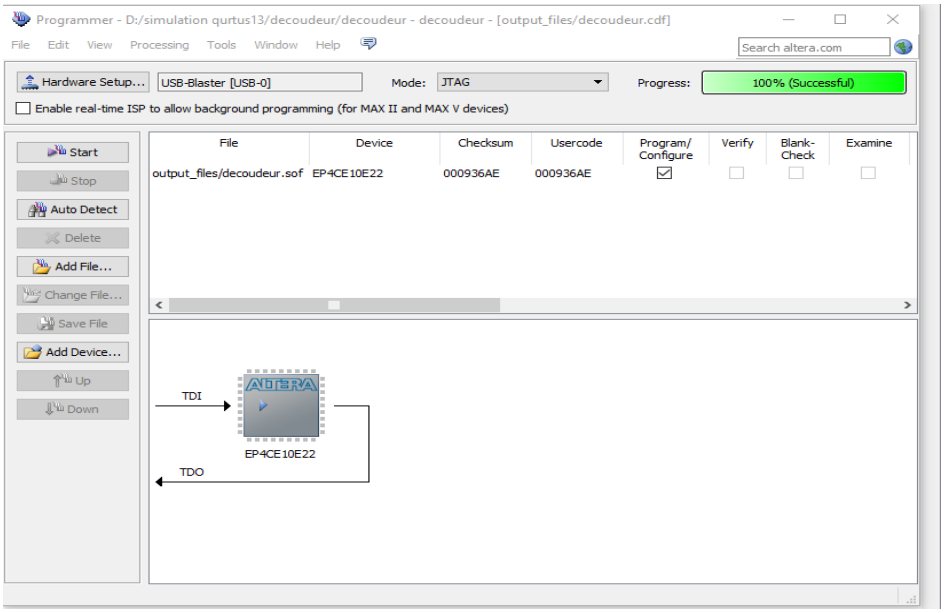


Figure III.14: Boite de dialogue de programmation.

Lorsque le fichier binaire de notre circuit est implémenté sur le FPGA de la carte de développement Cyclone IV le témoin de la touche **OK** s'allume en vert indique que notre FPGA est programmévoir la figure suivante :

Figure III.15: Plateforme Cyclone IV programmée.

III.4 Utilisation d'un FPGA pour la génération de signaux vidéo VGA

Pour fournir des options de sortie intéressantes dans des conceptions complexes, la sortie vidéo peut être développée à l'aide de matériel à l'intérieur du FPGA. Seuls cinq signaux ou broches sont nécessaires, deux signaux de synchronisation et trois signaux de couleur RVB. Un simple circuit de résistance et de diode est utilisé pour convertir les signaux de broche de sortie TTL du FPGA en signaux RVB analogiques basse tension pour le signal vidéo. Cela prend en charge deux niveaux pour chaque signal dans les données RVB et produit ainsi un total de huit couleurs. Ce circuit et un connecteur VGA pour un moniteur sont déjà installés sur la carte Altera. La boucle à verrouillage de phase (PLL) du FPGA peut être utilisée pour générer des horloges pour une grande variété de résolutions vidéo et de taux de rafraîchissement.

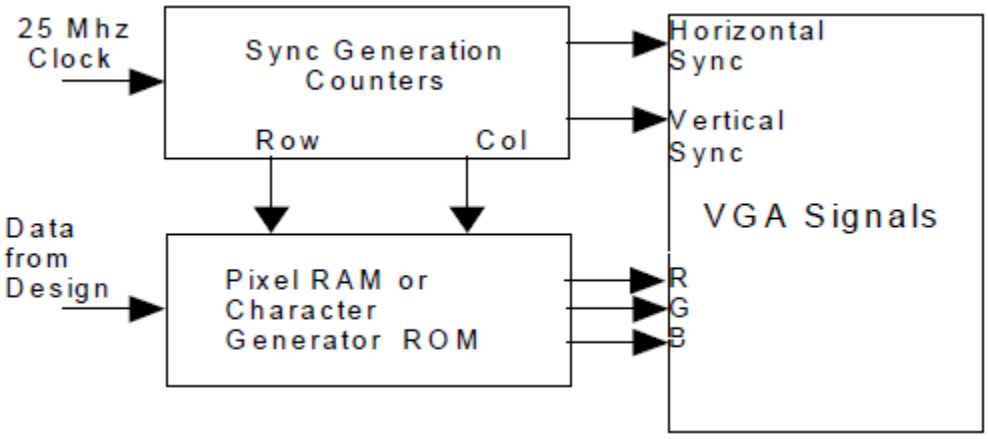


Figure III.16 : Génération FPGA de signaux vidéo VGA.

Comme le montre la figure III.16, une horloge de 25,175 MHz, qui est le 640 par 480 VGA débit de données de pixel d'environ 40ns est utilisé pour conduire les compteurs qui génèrent les signaux de synchronisation horizontale et verticale. Les compteurs supplémentaires génèrent des adresses de lignes et de colonnes. Dans certaines conceptions, la résolution des pixels sera réduite de 640 par 480 à une résolution inférieure en utilisant une opération de division d'horloge sur les compteurs de ligne et de colonne. Les adresses des lignes et des colonnes alimentent une RAM de pixels pour les données graphiques ou une ROM de générateur de

caractères lorsqu'elle est utilisée pour afficher du texte. La RAM ou la ROM requise est également implémentée dans la puce FPGA.

III.5 Conception vidéo basée sur des caractères : Affichage du message "WELCOME JURY"

L'affichage de textes et de graphismes sur un écran VGA nécessite la maîtrise des signaux de synchronisation et des données de pixel. Le code VHDL présenté par son RTL (figure III.18) illustre comment créer un contrôleur VGA pour afficher le message "WELCOME JURY". Ce contrôleur utilise des compteurs pour générer les signaux de synchronisation horizontale et verticale, ainsi qu'une ROM de caractères pour dessiner les lettres du message. En modifiant les couleurs des pixels, nous pouvons afficher le texte souhaité sur l'écran. Ce projet est une démonstration classique de l'utilisation de VHDL pour le traitement de graphismes simples en utilisant un contrôleur VGA.

La synthèse finale de ce premier projet devrait ressembler au schéma de la figure III.18 (menu *Tools → Netlist Viewers → RTL Viewer*).

III.5.1 Description du fonctionnement de ce message

Pour débuter, voir le schéma du circuit (figure III.18) global dans le FPGA qui nous permettra de mieux situer les différents blocs de fonctionnalité. Ce dessin nous permettra de faciliter l'explication des relations entre les différents signaux qui constituent le système.

Le code VHDL implémente un contrôleur VGA capable d'afficher le message "WELCOME JURY" sur un écran 640x480 à 60Hz.

III.5.1.1 Déclarations et paramètres VGA

Les paramètres pour la synchronisation horizontale et verticale sont définis pour une résolution de 640x480 pixels à 60 Hz. Ces paramètres incluent le nombre de pixels visibles (h_display, v_display), les périodes de front porch (h_front_porch, v_front_porch), de sync pulse (h_sync_pulse, v_sync_pulse) et de back porch (h_back_porch, v_back_porch).

Pour illustrer le protocole, on donne la représentation ci-dessous :

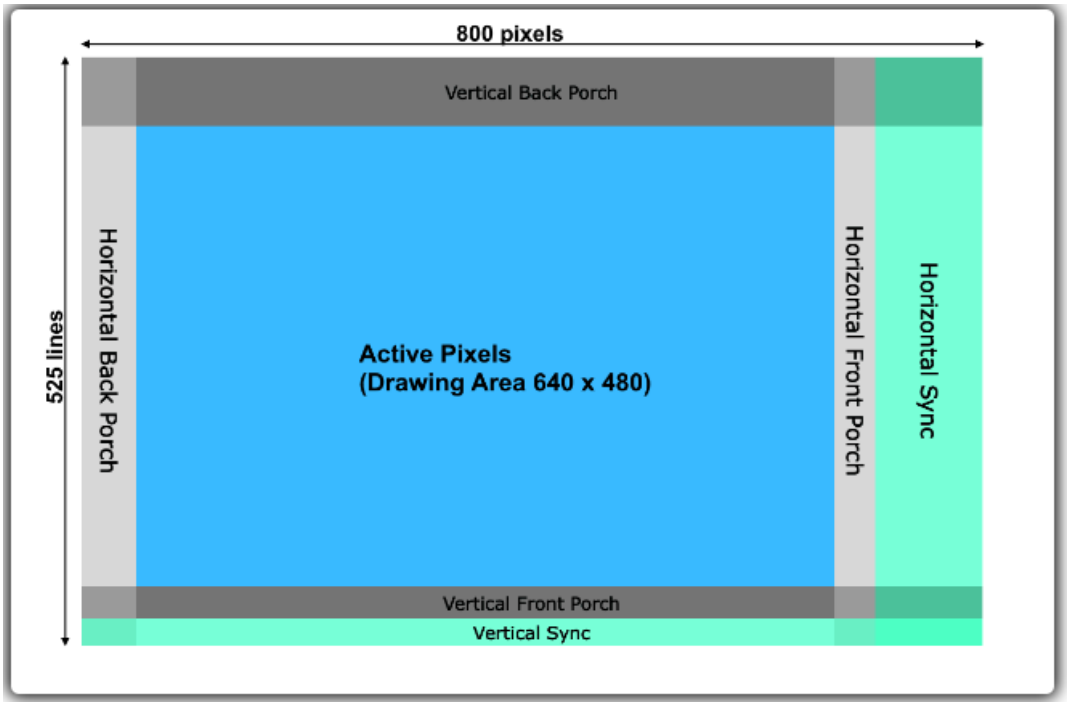


Figure III.17 : Protocole de synchronisation [III.4].

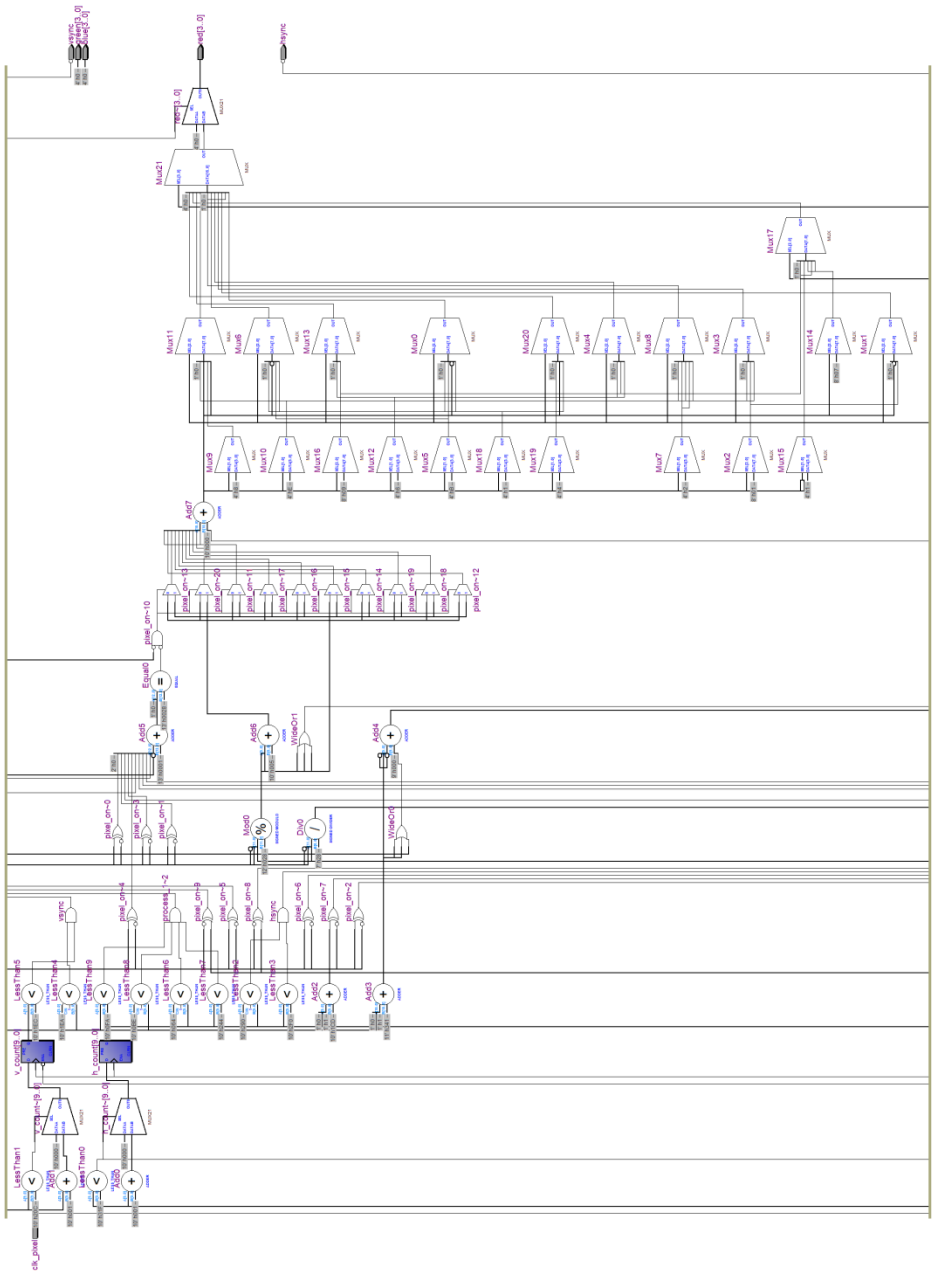


Figure III.18 : Schéma fonctionnel RTL de message WELCOME JURY.

III.5.1.2 Signaux de comptage

h_count et v_count sont des signaux utilisés pour suivre la position actuelle dans le cadre en cours d'affichage. Ils sont incrémentés à chaque cycle d'horloge de pixel pour parcourir chaque ligne et colonne de l'écran.

III.5.1.3 ROM de caractères

Un type char_rom est défini pour stocker les bits de chaque caractère dans le message "WELCOME JURY". Chaque caractère est représenté par une matrice 5x8

de bits (std_logic), où chaque bit indique si le pixel correspondant est allumé ('1') ou éteint ('0').

III.5.1.4 Processus de comptage

Un processus basé sur l'horloge de pixel (clk_pixel) incrémente h_count et v_count pour traverser l'écran de gauche à droite et de haut en bas. Lorsque h_count atteint la fin d'une ligne, il est réinitialisé et v_count est incrémenté pour passer à la ligne suivante. À la fin de la dernière ligne, v_count est également réinitialisé.

III.5.1.5 Génération des signaux de synchronisation

Les signaux de synchronisation horizontale (hsync) et verticale (vsync) sont générés en fonction de la position actuelle des compteurs horizontaux et verticaux. Ces signaux indiquent au moniteur VGA quand commencer une nouvelle ligne ou un nouveau cadre.

III.5.1.6 Génération du signal vidéo

Un autre processus vérifie si la position actuelle se trouve dans la zone où le message "WELCOME JURY" doit être affiché. Si c'est le cas, il détermine le caractère et le pixel spécifique à afficher et définit la couleur du pixel sur rouge si le bit correspondant dans la ROM de caractères est activé. Sinon, la couleur de fond par défaut (noir) est affichée.

Le code VHDL permet ainsi de contrôler un écran VGA pour afficher un message spécifique en utilisant la logique matérielle décrite en VHDL.

III.5.1.7 Affectation des pins

Le tableau suivant représente le nom des signaux utilisés et leurs numéros des pins sur le FPGA **EP4CE10E22C8**.

Tableau III.2 : Numéros des pins entrées/sorties utilisés de l'affichage du message WELCOME JURY [III.3].

Nom	Numéro des pins	Nom	Numéro des pins	Nom	Numéro des pins
clk_pixel	PIN_23	green[3]	PIN_136	red[3]	PIN_128
blue[3]	PIN_144	green[2]	PIN_137	red[2]	PIN_129
blue[2]	PIN_1	green[1]	PIN_138	red[1]	PIN_132
blue[1]	PIN_2	green[0]	PIN_141	red[0]	PIN_133
blue[0]	PIN_3	hsync	PIN_11	vsync	PIN_7

La figure III.19 montre le résultat obtenu après la programmation de notre FPGA avec le programme VHDL qui s'affiche le message WELCOME JURY.

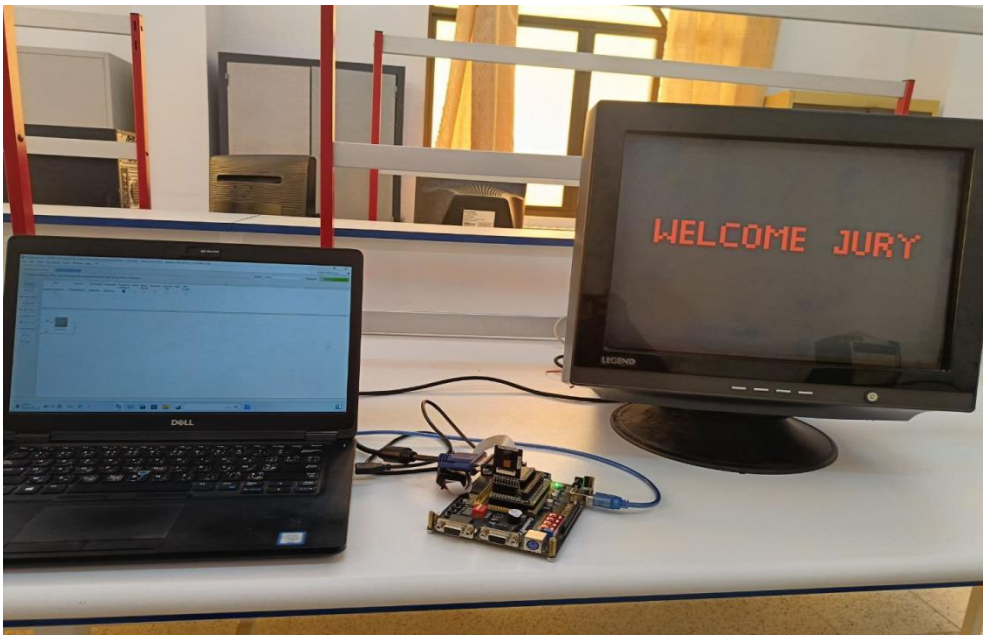


Figure III.19 : Affichage de message WELCOME JURY sur l'écran VGA.

III.6 Utilisation d'un FPGA/VHDL pour la détection des obstacles et la mesure de la distance en utilisant un capteur ultrasonique

III.6.1 Module de détecteur HC-SR04

Le capteur HC-SR04 utilise les ultrasons pour déterminer la distance d'un objet. Il offre une excellente plage de détection sans contact, avec des mesures de haute précision et stables. Son fonctionnement n'est pas influencé par la lumière du soleil ou des matériaux sombres, bien que des matériaux comme les vêtements puissent être difficiles à détecter [III.5].

Figure III.20 : Capteur ultrason HC-SR04.

III.6.1.1 Caractéristiques

- Dimensions : 45 mm x 20 mm x 15 mm
- Plage de mesure : 2 cm à 400 cm
- Résolution de la mesure : 0.3 cm
- Angle de mesure efficace : 15 °
- Largeur d'impulsion sur l'entrée de déclenchement : 10 µs (Trigger Input Pulse width)

III.6.1.2 Broches de connexion

- Vcc = Alimentation +5 V DC
- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation

III.6.1.3 Fonctionnement

Pour déclencher une mesure, il faut présenter une impulsion "high" (5 V) d'au moins 10 µs sur l'entrée "Trig". Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi. Lorsque celui-ci est détecté, il envoie un signal "high" sur la sortie "Echo", dont la durée est proportionnelle à la distance mesurée.

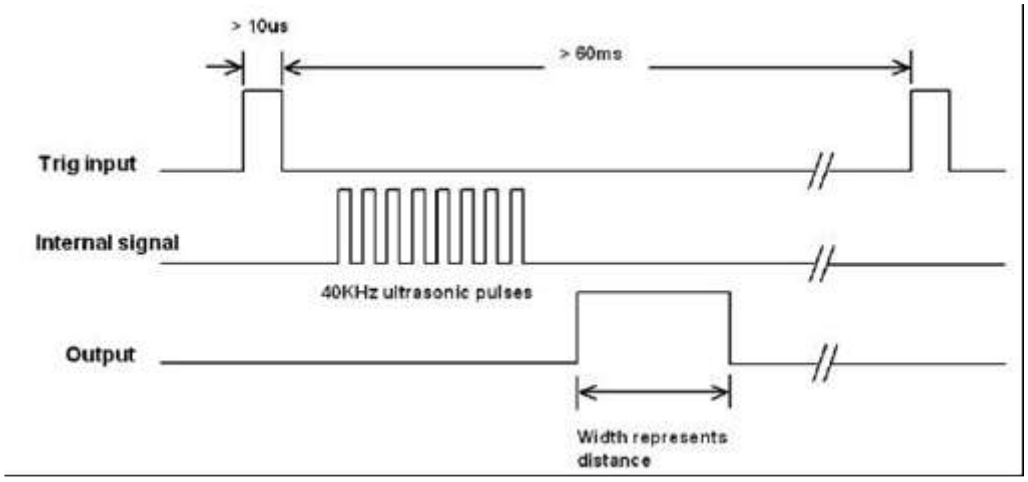


Figure III.21 : Diagramme de fonctionnement du capteur ultrason [III.5].

III.6.1.4 Spécifications et limites

Tableau III.3 : Spécifications et limites du capteur ultrason [III.5].

Paramètre	Min	Type	Max	Unité
Tension d'alimentation	4.5	5.0	5.5	V
Courant de repos	1.5	2.0	2.5	mA
Courant de fonctionnement	10	15	20	mA
Fréquence des ultrasons	-	40	-	kHz

Remarque : la borne GND doit être connectée en premier, avant l'alimentation sur Vcc.

III.6.1.5 Communication

La connexion entre FPGA et le détecteur a ultrason HC-SR 04 est illustrée sur la figure suivante :

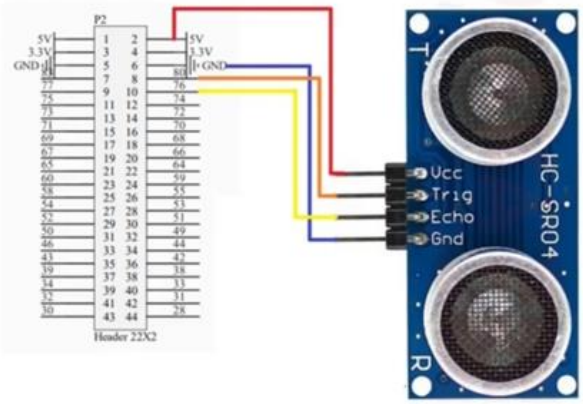


Figure III.22 : Connexion entre notre FPGA et HC-SR 04.

Le branchement du capteur HC-SR04 sur le FPGA est comme suit :

- VCC : relier au 5v de FPGA.
- GND : relier au GND de FPGA.
- TRIG : relier à l'entré 80 de FPGA.
- ECHO : relier à l'entré 76 de FPGA

III.6.2 Utilisation du capteur ultrasonique HC-SR04 pour la mesure de la distance et l'affichage sur 7 segments

Le système utilise un capteur ultrasonique HC-SR04 pour mesurer des distances et afficher les résultats sur un afficheur à 7 segments. Un buzzer est activé

lorsque la distance mesurée dépasse un certain seuil. Le capteur envoie une impulsion ultrasonique et mesure le temps écoulé jusqu'à ce que l'écho revienne, permettant ainsi de calculer la distance. Les données mesurées sont ensuite traitées et affichées, et un signal sonore est déclenché si la distance dépasse une limite prédéfinie. Voici une explication des différents codes VHDL utilisés pour implémenter cette fonctionnalité.

III.6.2.1 Générateur de Trigger Ultrasonique

L'entité TriggerGen est responsable de la génération d'un signal de déclenchement pour le capteur. Utilisant une horloge (clk), elle produit un signal trigger périodique. À chaque front montant de l'horloge, un compteur interne tick est incrémenté jusqu'à atteindre une valeur prédéfinie (nclks). Lorsque tick est inférieur à 500, le signal de déclenchement est actif ('1'), sinon, il est inactif ('0'). Cela génère des impulsions régulières nécessaires pour activer le capteur et envoyer des ondes ultrasonores.

Le code VHDL génère un signal de déclenchement (trigger) avec une période de 15 ms et une durée active de 10 µs, en utilisant une horloge de 50 MHz.

.

III.6.2.2 Compteur « counter »

L'entité « counter » est responsable de compter les impulsions d'horloge pour mesurer le temps écoulé entre l'envoi du signal ultrasonique et la réception de son écho. Le compteur est activé par le signal de déclenchement généré par l'entité « TriggerGen ». Le temps écoulé est ensuite utilisé pour calculer la distance jusqu'à l'objet.

Ce code définit un compteur qui compte le nombre d'impulsions d'horloge. Lorsque le signal « enable » est actif et qu'un front montant d'horloge est détecté, le compteur « tick » est incrémenté. Le compteur est réinitialisé lorsque le signal « reset » est actif. La sortie `q` représente la valeur du compteur, qui est utilisée pour mesurer le temps écoulé entre l'envoi et la réception des signaux ultrasoniques.

Le code VHDL implémente un compteur avec un signal de réinitialisation (reset), un signal d'activation (enable), et une sortie (q) qui représente la valeur actuelle du compteur.

III.6.2.3 Calcul de la Distance

Dans l'entité « distance_calculation », le temps d'écho mesuré « echo_count « est utilisé pour estimer la distance jusqu'à l'objet. Ce code convertit ce temps d'écho en une distance en centimètres en utilisant une série de conditions. Chaque condition vérifie une plage de valeurs de temps d'écho et attribue une valeur de distance spécifique en fonction de cette plage. Par exemple, si « echo_count » est compris entre 2900 et 4350, la distance est définie comme 1-1.5 cm, et ainsi de suite. La dernière condition attribue une distance par défaut si aucune des plages n'est satisfaite.

Le code VHDL implémente un calcul de distance basé sur une valeur de compteur d'écho (echo_count). La sortie (distance) représente la distance calculée en centimètres.

III.6.2.4 L'affichage de la valeur de la distance sur l'afficheur 7 segments

Ce code VHDL utilise une machine d'états pour alterner entre l'affichage des unités et des dizaines d'une distance mesurée, affichée sur un afficheur à 7 segments. Les valeurs de distance sont décodées et affichées de manière cyclique grâce à un compteur de commutation et une logique de décodage. Les calculs de délai et de cycles sont basés sur la fréquence d'horloge pour garantir une transition fluide et lisible.

III.6.2.5 Code principal

Dans ce code, l'entité « ultrasonic » coordonne les différentes parties du système. Elle utilise le composant « TriggerGen » pour générer le signal de déclenchement, le composant « counter » pour mesurer la durée de l'écho ultrasonique, le composant « distance_calculation » pour calculer la distance correspondante, et le composant « display_decoder » pour afficher la distance sur un afficheur à 7 segments. De plus, un processus est utilisé pour activer le buzzer lorsque la distance mesurée dépasse un seuil spécifique.

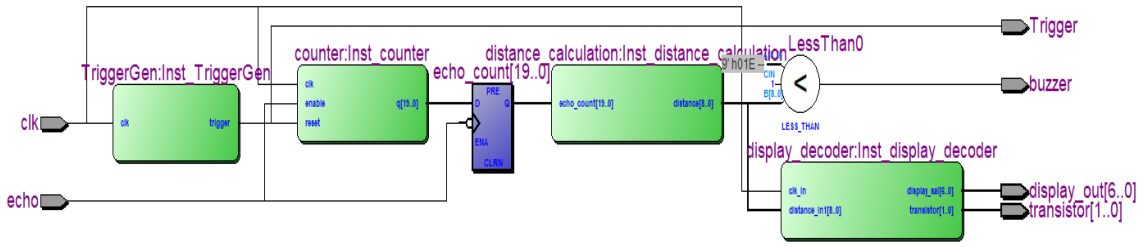


Figure III.23: Schéma fonctionnel RTL principal de programme VHDL « ultrasonic».

Ce code VHDL (présenté par le schéma RTL) implémente un système de mesure de distance utilisant un capteur ultrasonique. Le signal Trigger est généré pour déclencher le capteur, et la durée du signal echo est mesurée par un compteur. La distance est ensuite calculée à partir de cette durée et affichée sur un afficheur à 7 segments. Si la distance dépasse un certain seuil, un buzzer est activé.

III.6.2.6 Affectation des pins

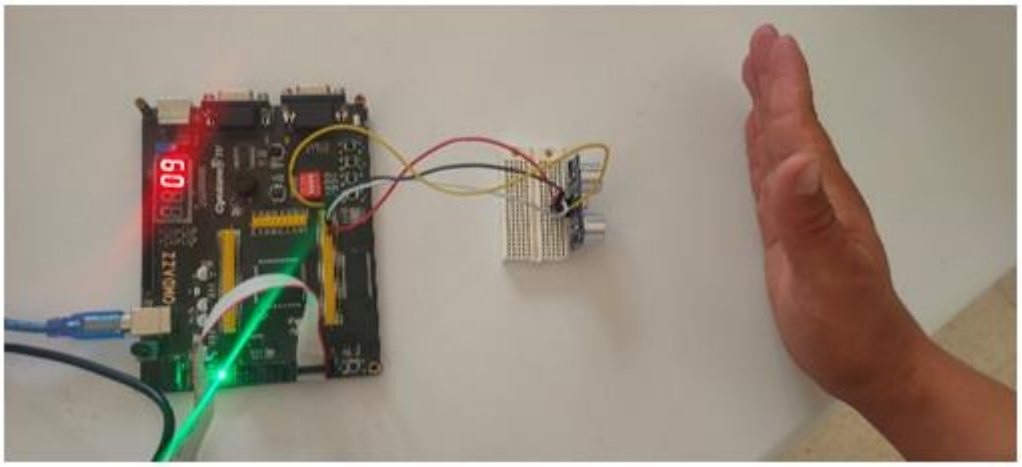
Le tableau III.4 résume les différents numéros des pins utilisés dans le brochage de FPGA avec le capteur ultrasonique **HC-SR04.**

Tableau III.4 : Numéros des pins entrées/sorties utilisés par le capteur ultrasonique [III.3].

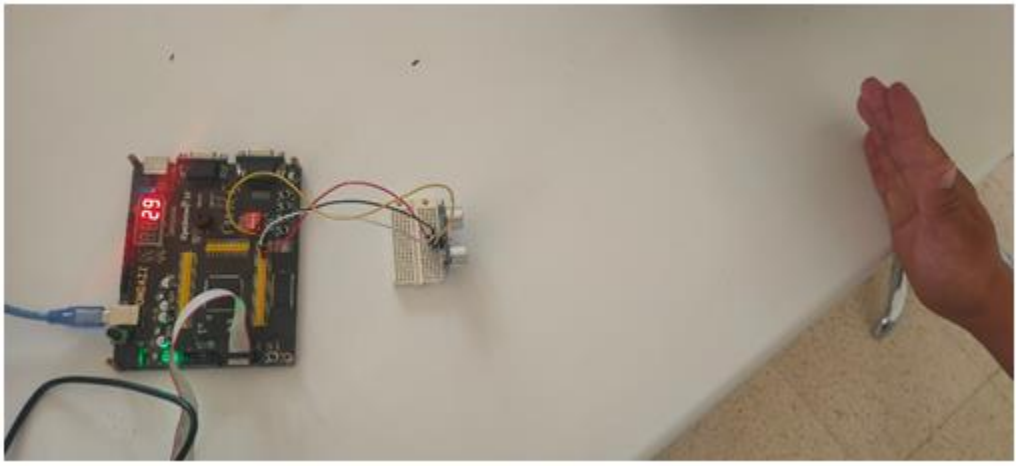
Entrées	Numéro des pins	Sorties	Numéro des pins	Sorties	Numéro des pins	Sorties	Numéro des pins
clk	PIN_23	buzzer	Pin_110	display_out[3]	PIN_129	Transistor [1]	PIN_133
echo	PIN_76	display_out[6]	PIN_124	display_out[2]	PIN_125	Transistor [0]	PIN_135
		display_out[5]	PIN_126	display_out[1]	PIN_121	Trigger	PIN_80
		display_out[4]	PIN_132	display_out[0]	PIN_128		

III.6.2.7 Résultats d'affichage des différentes distances

Les deux figures suivantes représentent respectivement l'affichage d'une distance de 9 cm et 29 cm.



(a)



(b)

Figure III.24 : Des exemples de la distance mesurée: (a) 09 cm, (b) 29 cm.

III.7 Utilisation du FPGA cyclone IV pour la génération de signaux vidéo VGA captés par la camera et l'afficher sur un moniteur VGA

III.7.1 Connection de la caméra OV7670 et le VGA avec le FPGA

Interfacez une caméra CMOS OV7670 avec un FPGA série Cyclone IV pour diffuser un flux vidéo en direct sur un écran VGA.

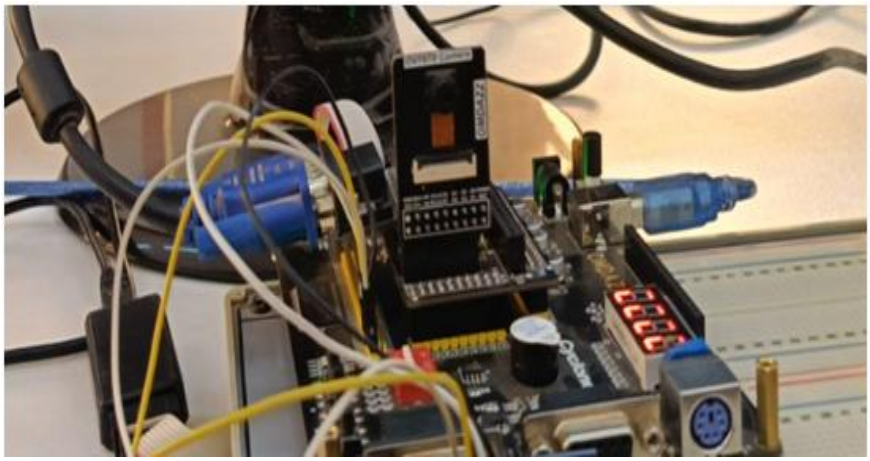


Figure III.25 : Schéma principal pour connecter la caméra OV7670 et VGA avec FPGA.

Pour interfacer avec la caméra OV7670, voici les étapes et les modules les plus critiques :

III.7.1.1 Configuration de la caméra (OV7670_controller.vhd)

Pour configurer correctement la caméra OV7670, il est nécessaire d'utiliser le protocole SSCB de type I2C pour communiquer avec elle, la configurer et obtenir des données d'image. Cela se fait via le fichier i2c_sender.vhd. Il est important de consulter la liste des registres de contrôle pour définir les paramètres tels que le format de sortie (RGB, QVGA, QCIF), le format de couleur (RGB565, RGB555), ainsi que les signaux de synchronisation et les coefficients matriciels qui influencent la qualité de l'image finale. Des détails supplémentaires sur les registres de contrôle sont disponibles dans le fichier OV7670_registers.vhd.

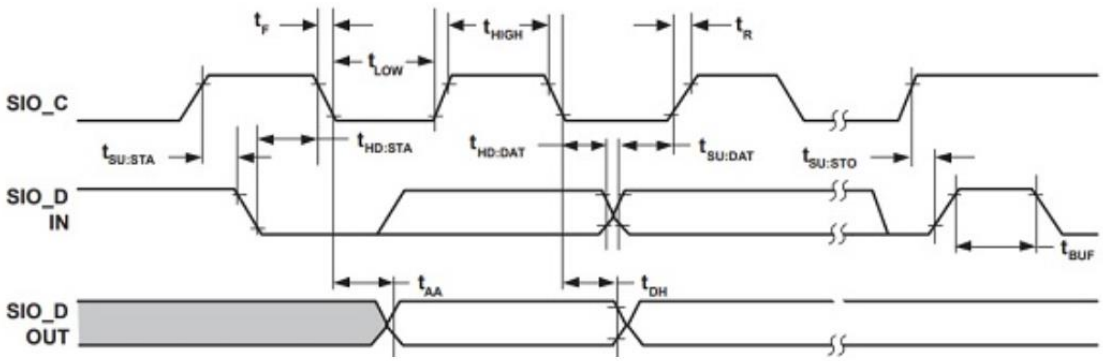


Figure III.26 : Interface SCCB de type I2C pour communiquer avec la caméra OV7670 [III.6].

La synthèse de code VHDL OV7670_controller.vhd de devrait ressembler au schéma de la figure suivante :

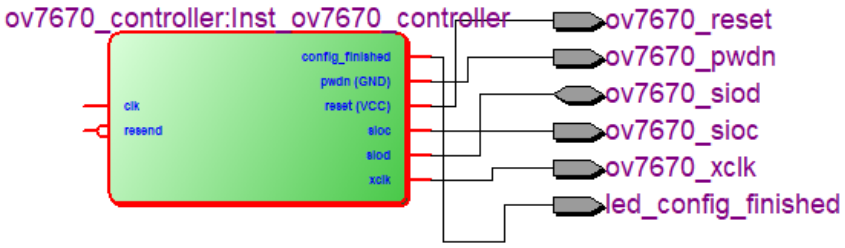


Figure III.27: Schéma RTL du programme VHDL « OV7670_controller».

III.7.1.2 Capture des données d'image (OV7670_capture.vhd)

Après avoir configuré la caméra, l'étape suivante consiste à capturer les données d'image avec le code VHDL OV7670_capture.vhd qui présente le schéma RTL de la figure III.28. Une fois de plus, consultez la fiche technique [III.6] de la caméra pour voir le chronogramme de sortie du format de sortie sélectionné. Par exemple, dans ce projet, le format RGB333 est choisi afin que le chronogramme de sortie pour RGB333 soit utilisé pour capturer correctement les données RVB des signaux de sortie de la caméra.

Figure III.28: RTL du programme VHDL « OV7670_capture».

III.7.1.3 Sauvegarde des données d'image (frame_buffer.vhd)

Après avoir pu configurer la caméra et capturer correctement les données d'image, les données d'image sont stockées dans une mémoire intermédiaire dans notre cas on a utilisé une RAM. Le problème avec le FPGA Cyclone IV EP4CE10E22C8 est que la taille de la mémoire n'est pas suffisante pour une taille d'image de 640 x 480. Pour utiliser les mêmes paramètres pour la caméra et le contrôleur VGA avec une taille d'image complète de 640x480 sans dépasser la

RAM de l'EP4CE10E22C8, l'astuce consiste à enregistrer seulement un pixel tous les 4 pixels pour la taille de 640x480. Ensuite, nous pouvons réduire la taille du frame buffer de 4 fois pour s'adapter au FPGA EP4CE10E22C8 tout en ayant une taille d'image de 160x120 sur un moniteur VGA.

Les entrées et les sorties utilisés par le VHDL frame_buffer.vhd sont représentées par le schéma fonctionnel de la figure III.29.

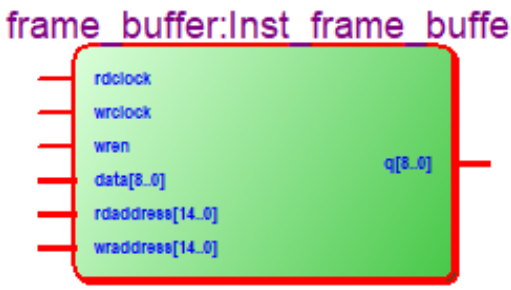


Figure III.29: RTL du programme VHDL « frame_buffer».

III.7.1.4 Générateur d'Adresses (Address_Generator.vhd)

Ce code VHDL Address_Generator.vhd présenté par le RTL de la figure III.30 implémente un générateur d'adresses qui incrémente une adresse sur 15 bits à chaque cycle d'horloge de 25 MHz, si le signal d'activation enable est actif. L'adresse est réinitialisée à zéro lorsque le signal de synchronisation verticale vsync est bas. L'adresse générée est utilisée comme sortie pour d'autres modules ou composants dans un système plus large.

Figure III.30: RTL du programme VHDL « Address_Generator».

III.7.1.5 Module RGB (RGB.vhd)

Ce code VHDL implémente un module de conversion de données d'entrée en composantes RGB (figure III.31). L'entrée Din est un vecteur de 9 bits, où chaque

groupe de bits est mappé aux composantes rouge, verte et bleue respectivement. Le signal Nblank contrôle si les valeurs RGB sont prises à partir de Din ou si elles sont réinitialisées à zéro. Lorsque Nblank est à '1', les bits de Din sont assignés aux sorties RGB. Sinon, les sorties RGB sont toutes mises à zéro.

Figure III.31: RTL du programme VHDL « RGB».

III.7.1.6 Phase-Locked Loop "PLL" (my_altpll.vhd)

Une PLL (Phase-Locked Loop)**,** ou boucle à verrouillage de phase, est un circuit électronique utilisé pour générer des signaux d'horloge synchronisés et stables. Elle est utilisée dans une variété d'applications telles que les communications, le traitement numérique des signaux, les systèmes de contrôle et les appareils électroniques, en particulier dans la conception de circuits numériques et de processeurs.

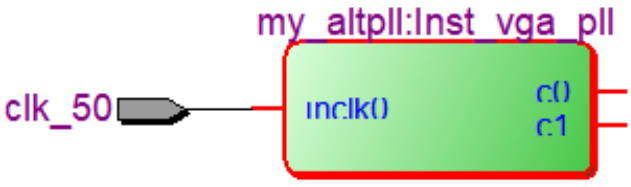


Figure III.32: RTL du programme VHDL « my_altpll».

III.7.1.7 Affichage de la vidéo/image en temps réel sur un moniteur VGA (VGA.vhd)

La dernière étape consiste à afficher les données d'image enregistrées dans le tampon d'image sur un moniteur VGA, et un contrôleur VGA est requis. Vous trouvez le schéma RTL du programme VHDL présenté sur la figure III.33 avec une horloge VGA de 25 MHz.

Figure III.33: RTL du programme VHDL « VGA».

III.7.1.8 Code principal

Ce code VHDL présenté par leur RTL de la figure III.34 définit une entité camera qui capture des images à partir d'une caméra OV7670 et les affiche sur un écran VGA. Il s'agit d'un système complexe intégrant plusieurs composants pour assurer la capture, le stockage et l'affichage des images. Voici une explication détaillée des composants et des interactions clés, en mettant l'accent sur le frame_buffer, et ses relations avec ov7670_capture et address_generator.

Ce système intégré permet de capturer des images en temps réel à partir d'une caméra OV7670 et de les afficher de manière fluide sur un écran VGA, en utilisant des techniques avancées de gestion de mémoire et de synchronisation des signaux.

❖ **Fonctionnement global**

✓ La caméra OV7670 capture des images, et ov7670_capture les envoie au frame_buffer.

✓ Le frame_buffer stocke les données d'image.

✓ address_generator génère les adresses de lecture pour récupérer les données du frame_buffer.

✓ RGB convertit les données en signaux RGB pour l'affichage VGA.

✓ VGA gère la synchronisation des signaux VGA pour afficher les images sur l'écran.

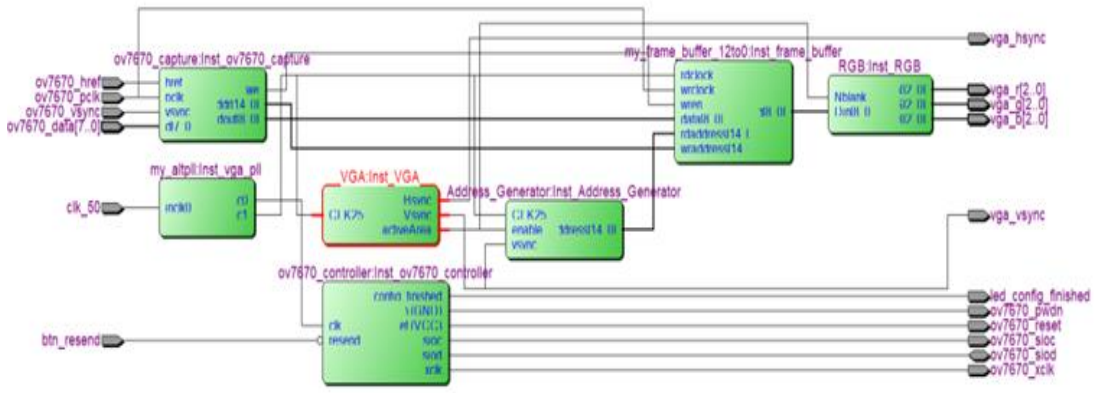


Figure III.34 : Schéma fonctionnel RTL de code principal « camera.vhd ».

III.7.1.9 Affectation des pins

Tableau III.5 : Numéros des pins entrées/sorties utilisés pour les différents composants du système [III.3].

Nom	Numéro des pins	Nom	Numéro des pins	Nom	Numéro des pins	Nom	Numéro des pins
btn_resend	PIN_91	ov7670_data[3]	PIN_103	ov7670_siod	PIN_85	vga_g[1]	PIN_138
clk_50	PIN_23	ov7670_data[2]	PIN_104	ov7670_vsync	PIN_88	vga_g[0]	PIN_141
led_config_finished	PIN_86	ov7670_data[1]	PIN_105	ov7670_xclk	PIN_87	vga_hsync	PIN_11
ov7670_data[7]	PIN_98	ov7670_data[0]	PIN_106	vga_b[2]	PIN_1	vga_r[2]	PIN_129
ov7670_data[6]	PIN_99	ov7670_href	PIN_89	vga_b[1]	PIN_2	vga_r[1]	PIN_132
ov7670_data[5]	PIN_100	ov7670_pclk	PIN_90	vga_b[0]	PIN_3	vga_r[0]	PIN_133
ov7670_data[4]	PIN_101	ov7670_sioc	PIN_84	vga_g[2]	PIN_137	vga_vsync	PIN_7

III.7.1.10 Résultats d'affichage des différentes distances

III.7.1.10.1 Conception matérielle

Le matériel de ce projet se compose d'une caméra OV7670, d'une carte de développement OMDAZZ de FPGA Cyclone IV, d'un module VGA, d'une carte de connexion et d'un moniteur VGA générique.

Les broches seront configurées pour fonctionner à des niveaux logiques de 2,5 V afin que l'interface électrique soit compatible.

III.7.1.10.2 Mémoire vidéo VGA

Le principal problème rencontré lors de la mise en œuvre de la caméra sur l'OMDAZZ est la taille de la RAM dont dispose l'OMDAZZ. Le FPGA cyclone IV quand on a utilisé n'a pas assez de mémoire pour stocker l'image VGA complète.

La figure III.35 représente l'image des étudiants captée par la caméra OV7670 affichée sur le moniteur VGA.

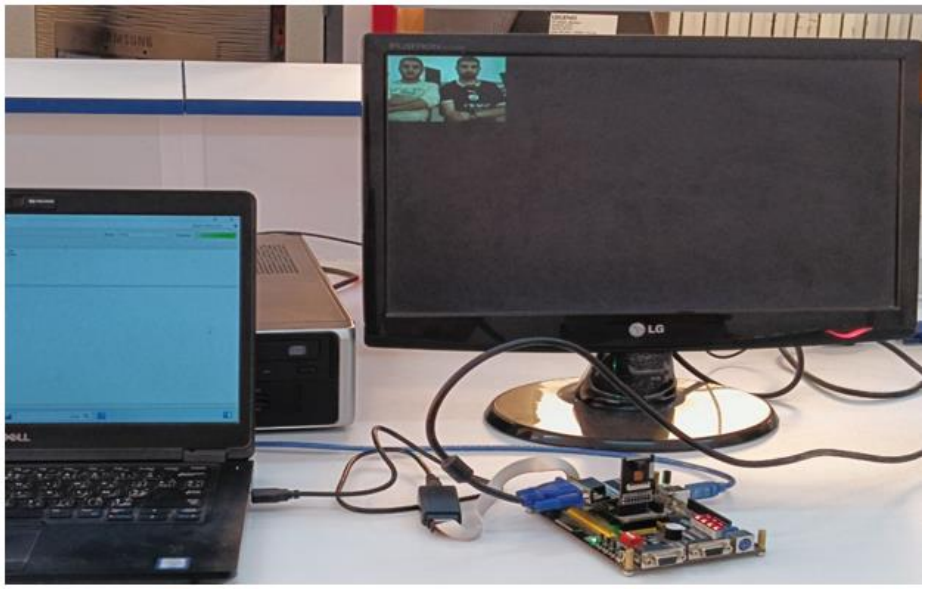


Figure III.35 : L'image des étudiants affichée sur le moniteur VGA de taille 160x120 pixels.

L'image ci-dessus montre le flux d'image en direct de la caméra CMOS OV7670 diffusé sur l'écran VGA.

Figure III.36: Flux d'image en direct de la caméra CMOS OV7670 diffusé sur l'écran VGA.

Cela entraînera le transfert d'un flux vidéo continu vers l'écran VGA à 30 images par seconde. Cela conclut à peu près le projet.

III.7.2 Intégration de la Caméra OV7670 et du Capteur Ultrasonique

Ce projet combine une caméra OV7670 et un capteur ultrasonique pour créer un système intégré de vision et de détection. Les images capturées par la caméra sont affichées sur un écran VGA, tandis que le capteur ultrasonique détecte la présence d'objets à proximité. Lorsque le capteur détecte un objet à une certaine distance, un buzzer est activé pour fournir une alerte sonore. Cette intégration permet une surveillance visuelle en temps réel tout en offrant une alerte sonore lorsque des objets s'approchent. Le système est conçu pour fonctionner de manière harmonieuse, avec chaque composant jouant un rôle crucial dans l'acquisition, le traitement et l'affichage des données ainsi que dans la détection et l'alerte.

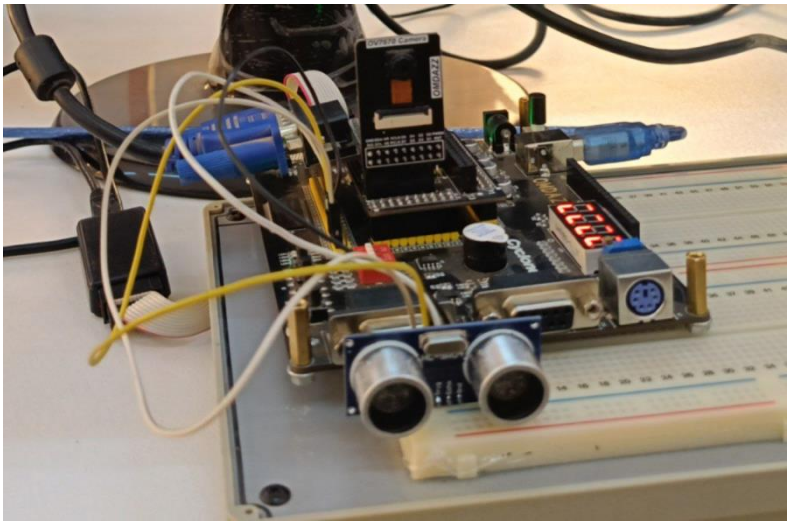


Figure III.37 : Schéma principal pour connecter la caméra OV7670, VGA et le capteur ultrasonique avec le FPGA.

La figure III.38 représente le schéma fonctionnel de code VHDL de la camera intégrée avec le capteur ultrasonique pour obtenir une image sur le moniteur VGA de taille 160x120 pixels.

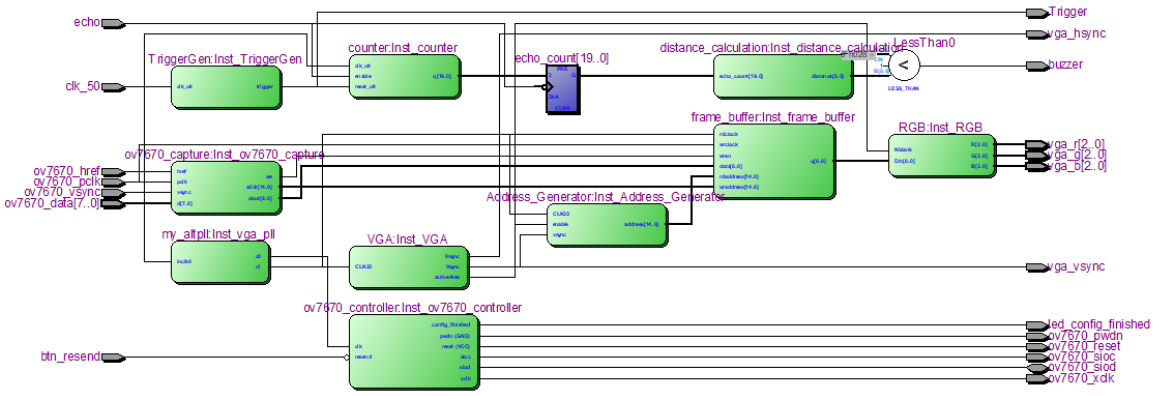


Figure III.38 : Schéma fonctionnel RTL du code principal de la camera intégrée avec le capteur ultrasonique

Remarque :

Le programme display_decoder sa fonction était d'afficher à quelle distance se trouve un objet, mais nous ne l'avons pas ajouté car le pin est spécifique à son fonctionnement. Nous l'avons utilisé pour la caméra, comme indiqué dans les tableaux précédents « Numéros des pins entrées/sorties utilisés ».

III.7.2.1 Affectation des pins

Les numéros des pins utilisés sont mentionnés dans le tableau suivant.

Tableau III.6 : Numéros des pins entrées/sorties utilisés dans le cas d'intégration de la caméra avec l'ultrasonique [III.3].

Name	Numéro des pins	Name	Numéro Des pins	Name	Numéro des pins	Name	Numéro des pins
btn_resend	PIN_91	ov7670_data[3]	PIN_103	ov7670_siod	PIN_85	vga_g[1]	PIN_138
clk_50	PIN_23	ov7670_data[2]	PIN_104	ov7670_vsync	PIN_88	vga_g[0]	PIN_141
led_config_finished	PIN_86	ov7670_data[1]	PIN_105	ov7670_xclk	PIN_87	vga_hsync	PIN_11
ov7670_data[7]	PIN_98	ov7670_data[0]	PIN_106	vga_b[2]	PIN_1	vga_r[2]	PIN_129
ov7670_data[6]	PIN_99	ov7670_href	PIN_89	vga_b[1]	PIN_2	vga_r[1]	PIN_132
ov7670_data[5]	PIN_100	ov7670_pclk	PIN_90	vga_b[0]	PIN_3	vga_r[0]	PIN_133
ov7670_data[4]	PIN_101	ov7670_sioc	PIN_84	vga_g[2]	PIN_137	vga_vsync	PIN_7
echo	PIN_76	buzzer	PIN_110	Trigger	PIN_80		

III.8 Conclusion

Dans de ce chapitre, nous avons abordé diverses méthodes de programmation et d'utilisation d'un FPGA en VHDL pour commander un écran VGA. Nous avons commencé par la conception d'une vidéo basée sur des caractères, où nous avons affiché un message de bienvenue "WELCOME JURY" sur un moniteur VGA avec une résolution de 640 par 480 pixels.

Ensuite, nous avons programmé notre FPGA pour mesurer et afficher la distance détectée par un capteur ultrasonique. Cette fonction a été implémentée en utilisant toujours le langage VHDL, démontrant ainsi la polyvalence et la puissance des FPGAs pour gérer différentes tâches.

Dans la deuxième partie de notre travail, nous avons interfacé une caméra CMOS OV7670 avec la carte de développement OMDAZZ. Nous avons transmis en direct le flux d'images capturées par la caméra sur un moniteur VGA avec une résolution de 160 par 120 pixels.

REFERENCES BIBLIOGRAPHIQUES -CHAPITRE III-

[III.1] https://fr.wikipedia.org/wiki/Video_Graphics_Array
[III.2] HADJ SAID DJAMAL, «Implémentation d'une application de tracking sur FPGA »mémoire de fin d'études en vue de l'obtention du diplôme de master 2 en électronique option réseaux et télécommunication université, MOULOUD MAMMERI, TIZIOUIZOU 2010.
[III.3]https://onedrive.live.com/?authkey=%21ABZ%5Fj27B4HZ3Adg&id=B2CDC3A30980D5BD%2182722&cid=B2CDC3A30980D5BD
[III.4] https://f-leb.developpez.com/tutoriels/fpga/controleur-vga/
[III.5] https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf
[III.6] https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

CONCLUSION GENERALE

CONCLUSION GENERALE

L'objectif de ce travail consistait à la programmation d'un FPGA de la famille Cyclone IV d'ALTERA en VHDL afin de lui permettre de commander un écran vidéo VGA pour obtenir un affichage en 640 par 480 pixels pour les caractères et une résolution de 160 par 120 pixels pour l'affichages des images. La carte de développement que nous avons utilisé contient un circuit FPGA de type EP4CE10E22C8 dont les sorties sont reliées à un connecteur VGA.

Nous avons utilisé pour l'implémentation le langage de description matériel VHDL qu'on a développé au deuxième chapitre en présentant ses avantages et ses fonctionnalités, nous avons démontré que c'est un langage complet destiné à représenter le comportement ainsi que l'architecture des systèmes numériques, comme nous avons présenté en deuxième partie de ce chapitre une généralité sur l'image, le signal vidéo VGA et comment générer l'affichage vidéo VGA à l'aide d'un FPGA ensuite une présentation de la vision artificielle.

Nous avons étudié en premier chapitre les circuits logiques programmables FPGA, nous avons montré les avantages que présentent de point de vue flexibilité, configuration et reprogrammation à volonté qui permet la modification et l'amélioration des conceptions réalisées.

Dans la première partie de troisième chapitre nous avons abordé une description détaillée sur la méthode de conception et d'implémentation des circuits numériques dans un FPGA on utilisant le logiciel Quartus II d'ALTERA, ensuite, nous avons réalisé les différentes descriptions en VHDL de nos fonctions de génération d'affichage vidéo VGA.

Dans un premier temps nous avons présenté les diverses méthodes de la programmation et d'utilisation d'un FPGA en VHDL pour commander un écran VGA. Nous avons commencé par la conception d'une vidéo basée sur des caractères, où nous avons affiché un message "WELCOME JURY" sur un moniteur VGA avec une résolution de 640 par 480 pixels.

Ensuite, nous avons programmé notre FPGA pour mesurer et afficher la distance détectée par un capteur ultrasonique. Cette fonction a été implémentée en utilisant toujours le langage VHDL, démontrant ainsi la polyvalence et la puissance des FPGAs pour gérer différentes tâches.

Dans la dernière partie pratique de notre travail, nous avons interfacé une caméra CMOS OV7670 avec la carte de développement OMDAZZ. Nous avons transmis en direct le flux d'images capturées par la caméra sur un moniteur VGA avec une résolution de 160 par 120 pixels.

Les résultats obtenus nous ont permis de démontrer le bon fonctionnement de notre circuit logique programmable FPGA, et de saisir l’importance de leur utilisation dans des applications variées.

Comme perspective pour les travaux futurs, nous proposons d'explorer davantage l'intégration des systèmes de vision artificielle sur FPGA. En ajoutant des algorithmes de vision par ordinateur, notre système pourrait non seulement capturer et afficher des images, mais aussi analyser et interpréter les scènes pour identifier des objets ou des situations spécifiques, augmentant ainsi les capacités de surveillance et d'alerte.