الجـــــــــــــــــــــمـهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــعـليـم 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

Conception et réalisation d'un véhicule omnidirectionnel avec système embarqué FPGA pour des applications d'intelligence artificielle

Présenté par : MAHMOUDI Sara Chafika

Soutenue publiquement, Juin 2025, devant le jury composé de :

Mr. CHAMI Nadir	MCB	Univ. Saida	Président
Melle. SEGHIER Salima	MCB	Univ. Saida	Encadrant
Melle. MOSTEFAI Lotfi	MCA	Univ. Saida	Examinateur

Année universitaire 2024/2025

Remerciements

Je remercie d'abord «Allah » qui m'a donné la force

et le courage pour terminer mes études et élaborer

ce modeste travail.

En second lieu, ma profonde gratitude à mon encadreur ***Dr SEGHIER Salima****, pour son accompagnement, ses conseils pertinents, et sa disponibilité tout au long de ce travail. Sa rigueur scientifique et ses remarques constructives ont grandement contribué à l'aboutissement de ce mémoire.*

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

Un grand merci à Mr A.ABDOUN l'ingénieur de laboratoire de FPGA.

Un grand remerciement à mes parents qui ont sacrifié
Pour moi, car grâce a leurs efforts nous somme là.

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

Dédicaces

Je dédie tout d'abord ma réussite à moi-même, cette âme ambitieuse qui a œuvré durant toutes ces années.

Je dédie ce mémoire À :

Mon cher père

Et ma chère mère

Pour leur amour, leurs sacrifices, leur soutien inestimable et leur confiance indéfectible. Ce travail est le fruit de vos encouragements constants.

À mes chères sœurs :

Iness, Nessrine, Marwa.

*À **ma famille***

Pour sa présence, son soutien moral et ses prières tout au long de mon parcours.

*À **mes enseignants et surtout mon encadreur,** pour la qualité de leur enseignement et leur accompagnement tout au long de ma formation.*

*À **mes chers amis et collègues,** avec qui j'ai partagé des moments précieux de travail, de réflexion et d'amitié.*

SARA

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. Notre projet, s'inscrit dans le cadre de la conception de systèmes embarqués intelligents, en exploitant la puissance des FPGA pour la commande en temps réel d'un véhicule mobile omnidirectionnel. L'objectif principal est de réaliser un robot capable de se déplacer dans toutes les directions, d'interagir avec son environnement, et de fonctionner à la fois en mode manuel (piloté par Bluetooth via une application mobile) et en mode autonome (avec prise de décision basée sur un capteur à ultrasons). Dans une première phase, le véhicule est commandé manuellement à l'aide d'une application mobile développée avec MIT App Inventor, qui envoie des instructions via un module Bluetooth HC-05. Le FPGA, programmé en VHDL, interprète ces commandes pour contrôler les moteurs et assurer des déplacements fluides dans toutes les directions. Un écran LCD est intégré au système pour fournir une interface visuelle, permettant par exemple l'affichage d'un message d'accueil (« WELCOME ») ou des distances mesurées par un capteur ultrasonique HC-SR04. Dans une deuxième phase, le véhicule fonctionne en mode autonome. Il détecte les obstacles dans son environnement à l'aide du capteur ultrasonique. Si un objet est détecté à moins de 30 cm, le buzzer s'active automatiquement, et le véhicule change de trajectoire en effectuant une rotation vers la gauche, illustrant une logique décisionnelle simple, caractéristique d'une IA embarquée de premier niveau. Une extension importante du projet concerne la sécurisation de l'accès au véhicule par identification sans contact, grâce à l'intégration d'un module RFID PN532. Lorsqu'un badge RFID autorisé est présenté, le FPGA valide l'identifiant reçu via une communication UART, et actionne un servomoteur pour ouvrir automatiquement la porte du véhicule, indiquant un refus d'accès. Ce système illustre une application pratique de la reconnaissance RFID dans un contexte embarqué sécurisé, avec un contrôle précis du mouvement via PWM. Ce projet met ainsi en œuvre de manière concrète l'intégration de différents modules : capteurs, moteurs, interface utilisateur, logique VHDL, communication sans fil, authentification RFID, et prise de décision autonome. Il démontre la pertinence des FPGA dans la robotique mobile intelligente et ouvre des perspectives vers des systèmes plus avancés, combinant mobilité, sécurité et intelligence embarquée.

MOTS CLES:

FPGA, VHDL, Cyclone IV, Systèmes embarqués, Robot mobile omnidirectionnel, Mode autonome, Bluetooth, RFID, UART, MIT App Inventor, logiciel Qartus II.

Abstract:

FPGAs (Field Programmable Gate Arrays) are parallel electronic circuits that today make it possible to develop applications with increasingly high execution speeds and demanding hardware resources. Our project is part of the design of intelligent embedded systems, exploiting the power of FPGAs for real-time control of an omnidirectional mobile vehicle. The main objective is to create a robot capable of moving in all directions, interacting with its environment, and operating both in manual mode (controlled by Bluetooth via a mobile application) and in autonomous mode (with decision-making based on an ultrasonic sensor). In a first phase, the vehicle is manually controlled using a mobile application developed with MIT App Inventor, which sends instructions via an HC-05 Bluetooth module. The FPGA, programmed in VHDL, interprets these commands to control the motors and ensure smooth movements in all directions. An LCD screen is integrated into the system to provide a visual interface, allowing for example the display of a welcome message ("WELCOME") or distances measured by an HC-SR04 ultrasonic sensor. In a second phase, the vehicle operates in autonomous mode. It detects obstacles in its environment using the ultrasonic sensor. If an object is detected within 30 cm, the buzzer is automatically activated, and the vehicle changes trajectory by rotating to the left, illustrating a simple decision logic, characteristic of a first-level embedded AI. An important extension of the project concerns securing vehicle access by contactless identification, thanks to the integration of a PN532 RFID module. When an authorized RFID badge is presented, the FPGA validates the received identifier via UART communication, and activates a servomotor to automatically open the vehicle door, indicating a refusal of access. This system illustrates a practical application of RFID recognition in a secure embedded context, with precise movement control via

PWM. This project thus concretely implements the integration of different modules: sensors, motors, user interface, VHDL logic, wireless communication, RFID authentication, and autonomous decision-making. It demonstrates the relevance of FPGAs in intelligent mobile robotics and opens up perspectives towards more advanced systems, combining mobility, security and embedded intelligence.

Keywords:
FPGA, VHDL, Cyclone IV, Embedded Systems, Omnidirectional Mobile Robot, Autonomous Mode, Bluetooth, RFID, UART, MIT App Inventor, 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. مشروعنا جزء من تصميم أنظمة مُدمجة ذكية، تُستغل قوة مصفوفات البوابات القابلة للبرمجة ميدانيًا للتحكم الفوري في مركبة متنقلة متعددة الاتجاهات. الهدف الرئيسي هو إنشاء روبوت قادر على الحركة في جميع الاتجاهات، والتفاعل مع بيئته، والعمل في الوضع اليدوي (يتم التحكم فيه عبر البلوتوث عبر تطبيق جوال) والوضع الذاتي (مع اتخاذ قرارات بناءً على مستشعر فوق صوتي). في المرحلة الأولى، يتم التحكم في المركبة يدويًا باستخدام تطبيق جوال مُطور بالتعاون مع MIT App Inventor، والذي يُرسل التعليمات عبر وحدة بلوتوث HC-05. تُفسر مصفوفة البوابات القابلة للبرمجة ميدانيًا (FPGAs)، المُبرمجة بلغة VHDL، هذه الأوامر للتحكم في المحركات وضمان حركة سلسة في جميع الاتجاهات. شاشة LCD مُدمجة في النظام لتوفير واجهة مرئية، تسمح، على سبيل المثال، بعرض رسالة ترحيب ("WELCOME") أو المسافات التي يُقاسها مستشعر فوق صوتي HC-SR04. في المرحلة الثانية، تعمل المركبة في وضع التشغيل الذاتي. تكتشف العوائق في بيئتها باستخدام مستشعر الموجات فوق الصوتية. في حال اكتشاف جسم ضمن نطاق 30 سم، يُفعّل الجرس تلقائيًا، وتغيّر المركبة مسارها بالالتفاف إلى اليسار، مما يُظهر منطق قرار بسيطًا، وهو سمة من سمات الذكاء الاصطناعي المدمج من المستوى الأول. ومن الإضافات المهمة للمشروع تأمين دخول المركبة عبر تحديد الهوية بدون تلامس، وذلك بفضل دمج وحدة PN532 RFID. عند تقديم شارة RFID معتمدة، يتحقق FPGA من صحة المعرف المُستلم عبر اتصال UART، ويُفعّل محرك سيرفو لفتح باب المركبة تلقائيًا، مُشيرًا إلى رفض الدخول. يُوضح هذا النظام تطبيقًا عمليًا للتعرف على RFID في سياق مدمج آمن، مع تحكم دقيق في الحركة عبر PWM. وبالتالي، يُطبّق هذا المشروع بشكل ملموس دمج وحدات مختلفة: المستشعرات، والمحركات، وواجهة المستخدم، ومنطق VHDL، والاتصالات اللاسلكية، ومصادقة RFID، واتخاذ القرارات بشكل ذاتي. ويوضح أهمية FPGAs في مجال الروبوتات المتنقلة الذكية ويفتح آفاقًا نحو أنظمة أكثر تقدمًا، تجمع بين التنقل والأمان والذكاء المضمن.

الكلمات الدالة:
FPGA، VHDL، Cyclone IV، الأنظمة المضمنة، الروبوت المتنقل متعدد الاتجاهات، الوضع المستقل، البلوتوث، RFID، UART، MIT App Inventor، برنامج Qartus II.

SOMMAIRE

Chapitre II : Présentation de langage VHDL & RFID

Chapitre III Résultats de conception et réalisation d'un véhicule utilisant un FPGA

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.

CPU : Central **P**rocessing **U**nit

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

D

DC : D*irect* **C**urrent.

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

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

F

FLASH: Flash Memory

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

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

G

GPU: Graphics **P**rocessing **U**nit

H

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

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

I

IA: Intelligence **A**rtificial

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

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

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

I2C : Inter-**I**ntegrated **C**ircuit

J

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

L

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

N

NFC: Near **F**ield **C**ommunication

P

PAL: Programmable Array Logic

PLD: Programmable Logic Device

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

PWM: Pulse **W**idth **M**odulation

R

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

RFID: Radio **F**requency **Id**entification

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

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

S

SDRAM: Synchronous **D**ynamic **R**andom **A**ccess **M**emory

SPLD: Simple **P**rogrammable **L**ogic **D**evice

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

SPI: Serial **P**eripheral **I**nterface

T

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

U

UART : Universal **A**synchronous **R**eceiver/**T**ransmitter

UID : Unique **Id**entifier

V

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

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

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

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Depuis la commercialisation du premier circuit programmable FPGA (Field Programmable Gate Arrays) 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].

Parallèlement, l'évolution rapide des technologies embarquées et l'essor de l'intelligence artificielle (IA) ont renforcé le besoin en systèmes intelligents, autonomes et réactifs, notamment dans des domaines comme la robotique mobile, l'industrie ou la logistique automatisée. Dans ce contexte, les FPGA apparaissent comme une solution idéale pour concevoir des architectures matérielles parallèles et adaptables, capables de satisfaire des exigences strictes en matière de performance, de consommation et de traitement en temps réel.

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 fournis 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.

Le présent mémoire s'inscrit dans ce cadre et porte sur la conception et la réalisation d'un véhicule omnidirectionnel piloté par un FPGA de la famille Cyclone IV de type EP4CE10E22C8. Ce véhicule intègre plusieurs fonctions avancées combinant mobilité, perception de l'environnement et interaction intelligente. Il a pour but de démontrer comment le langage VHDL peut être exploité pour implémenter un système embarqué complet, englobant la lecture de

capteurs, le contrôle de moteurs, la communication sans fil, ainsi qu'un traitement logique simple et réactif.

Le véhicule développé fonctionne selon deux modes :

- ✓ Mode manuel, où l'utilisateur commande le déplacement via une application mobile Bluetooth, à l'aide d'un module HC-05.
- ✓ Mode autonome, où le véhicule réagit à son environnement à l'aide d'un capteur ultrasonique HC-SR04. Il mesure les distances, affiche les valeurs sur un écran LCD, active un buzzer en cas d'obstacle à moins de 30 cm, et prend des décisions comme tourner à gauche pour éviter la collision.

Parmi les interfaces de communication utilisées dans ce projet, le module RFID PN532 occupe une place importante. Il permet d'implémenter une méthode simple et efficace d'identification des utilisateurs via la technologie Radio Frequency Identification (RFID)**.**

Le système RFID repose sur l'échange de données entre un lecteur et un tag (ou carte) à l'aide d'ondes radio. Lorsqu'un tag RFID entre dans la zone de détection du lecteur, ce dernier peut lire son identifiant unique (UID). Le module PN532 utilisé ici est compatible avec la norme NFC (Near Field Communication) et communique avec le FPGA via une interface série UART, bien adaptée aux architectures embarquées.

Dans le cadre de notre projet, ce système est utilisé pour sécuriser l'accès à la porte du véhicule**.** Si le tag présenté est reconnu comme autorisé, un servomoteur est activé pour permettre l'ouverture automatique de la porte. Cette solution présente l'avantage d'être sans contact, rapide et fiable, tout en offrant une couche de sécurité adaptée aux systèmes embarqués.

Cette fonctionnalité est particulièrement pertinente si le véhicule est envisagé comme un distributeur mobile (par exemple, pour la livraison de médicaments, d'outils ou de colis). Seuls les utilisateurs munis d'un badge ou d'une carte RFID autorisée peuvent alors accéder à son contenu, renforçant ainsi le contrôle d'accès dans des environnements partagés ou sensibles.

L’intégration du module RFID illustre non seulement l’exploitation d’un périphérique externe via une liaison série UART, mais aussi la mise en œuvre d’un comportement conditionnel basé sur une authentification, qui constitue une brique élémentaire d’une intelligence embarquée sur FPGA.

Le choix d’un véhicule omnidirectionnel, capable de se déplacer dans toutes les directions sans rotation préalable, améliore considérablement la manœuvrabilité, notamment dans des espaces restreints, ce qui est essentiel dans de nombreuses applications de robotique moderne.

Ce projet a permis de concrétiser les enseignements théoriques liés aux FPGA et au VHDL, tout en abordant des notions simples d’intelligence embarquée. Il ouvre également la voie à des perspectives d’amélioration, telles que l’intégration de modules de vision artificielle, de cartographie dynamique, ou d’algorithmes d’optimisation de trajectoire.

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 plus spécialement les FPGA de la famille cyclone IV, en détaillant leurs caractéristiques et leurs avantages dans les systèmes embarqués.

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. Ce chapitre aborde également des notions liées à la présentation des notions de base du système RFID utilisé dans le projet.

Le troisième chapitre est le cœur pratique du projet. Il présente la conception matérielle et logicielle. Il décrit le processus complet de conception et d’implémentation sur FPGA à l’aide de l’environnement Quartus II. Il présente la carte de développement FPGA utilisée, le matériel connecté, ainsi que les blocs VHDL développés (schémas fonctionnels RTL).

Dans un premier temps, nous allons afficher le message "WELCOME " sur un écran LCD pour l'accueil. Ensuite, nous avons utilisé un capteur ultrasonique pour mesurer la distance et afficher la valeur mesurée sur l'écran LCD. Lorsque

quelqu'un entre dans une zone de 30 cm, un buzzer s'active pour alerter de sa présence.

La conception et la réalisation de notre véhicule omnidirectionnel se divise en deux parties :

- Commande manuelle : création d'une application mobile (via MIT App Inventor) pour envoyer des commandes Bluetooth au FPGA, contrôle des moteurs pour un déplacement omnidirectionnel.
- Mode autonome : développement d'un système de prise de décision simple basé sur les mesures d'un capteur ultrasonique HC-SR04. Si la distance est inférieure à 30 cm, le véhicule active un buzzer et effectue une rotation à gauche.

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. En particulier, le FPGA de la famille Cyclone IV fait l'objet d'une présentation détaillée de ses composants et caractéristiques.

I.2 Circuits Logiques Programmables du type FPGA

I.2.1 Définition du FPGA

Les FPGA, sigle anglais qui signifie « **F**ield **P**rogrammable **G**ates **A**rrays » 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].

Figure I.1 : Image d'un circuit FPGA [I.7].

Les FPGAs, sont des composants électroniques programmables de la famille des PLDs (**P**rogrammable **L**ogic **D**evice). 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 souples se 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.8].

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 (**L**ook **U**p **T**able) [I.9]. Le résultat peut être éventuellement stocké dans un registre, les blocs RAM permettent d'implanter des mémoires adressables et des FIFO (**F**irst **I**n, **F**irst **O**ut), 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 (**H**ardware **D**esign **L**anguage) 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.10].

I.2.2 Architecture [I.11]

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.

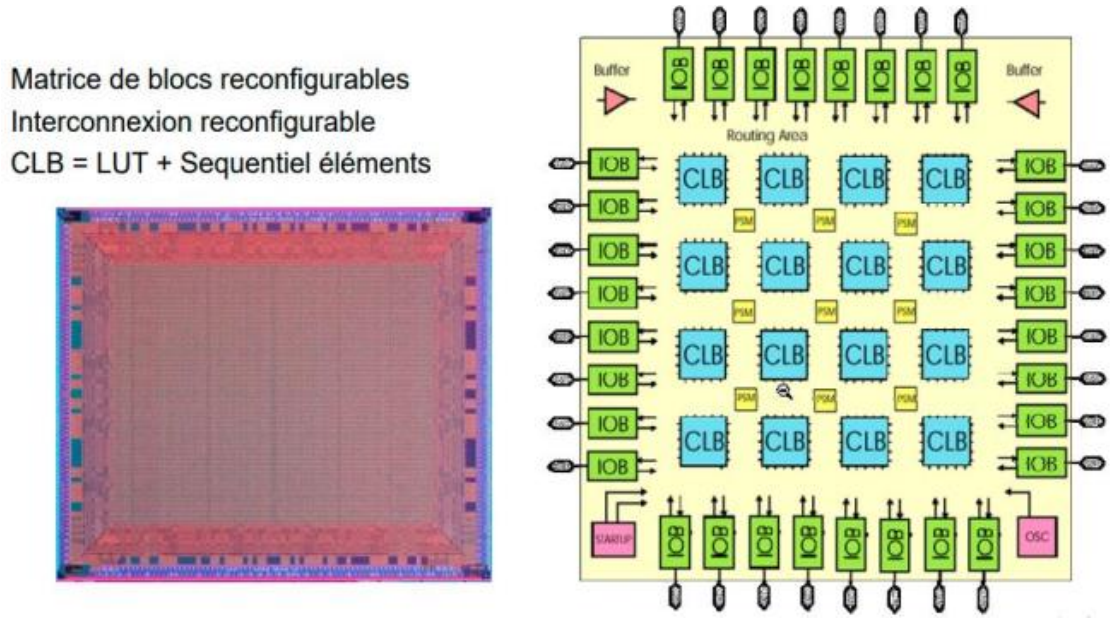


Figure I.2 : Architecture interne des FPGAs

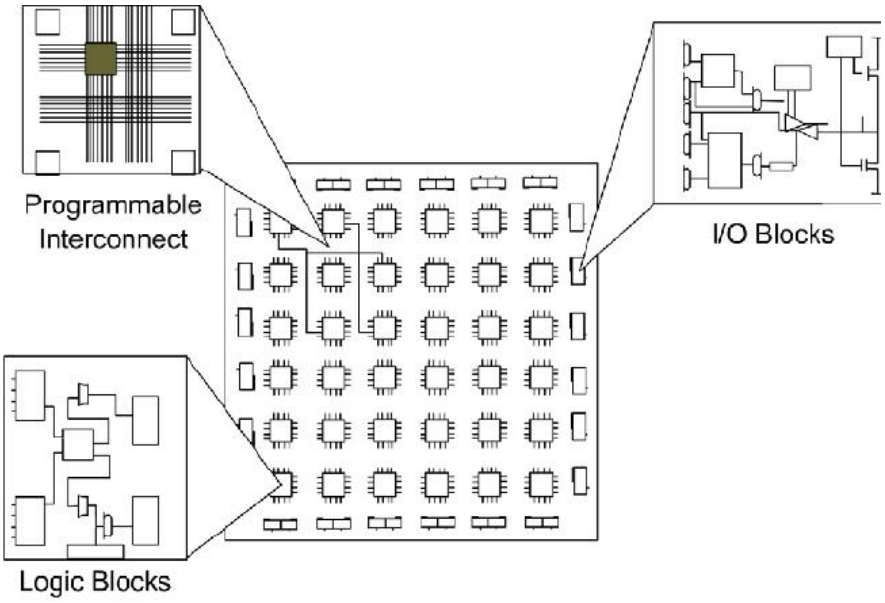


Figure I.3 : Différents secteurs d'un FPGA

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 I.4.

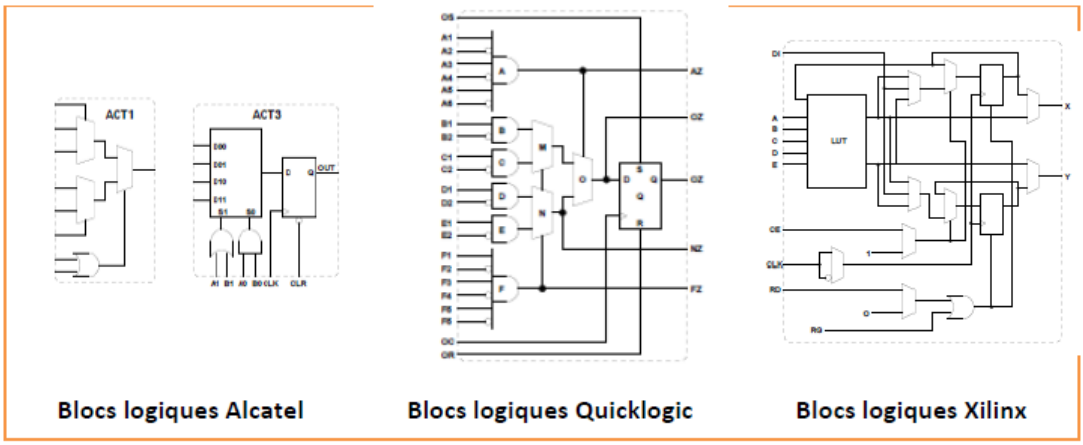


Figure I.4 : Exemple de blocs logiques de différents fabricants.

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 I.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.

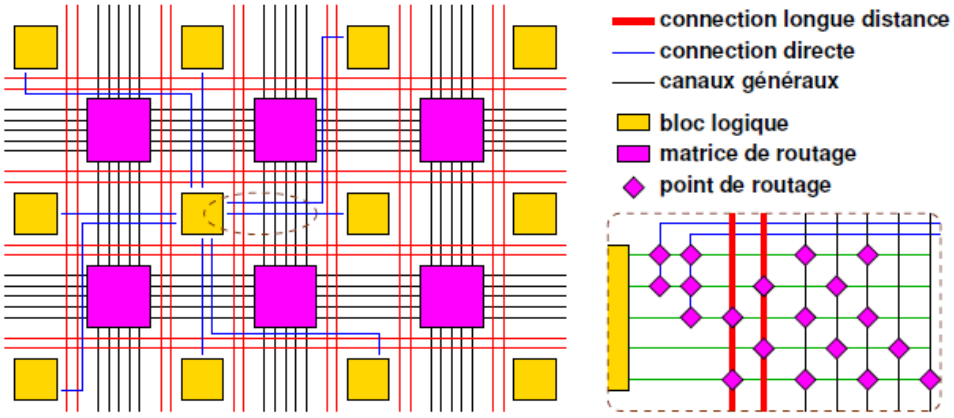


Figure I.5 : Structure générale du routage

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.2.3 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].

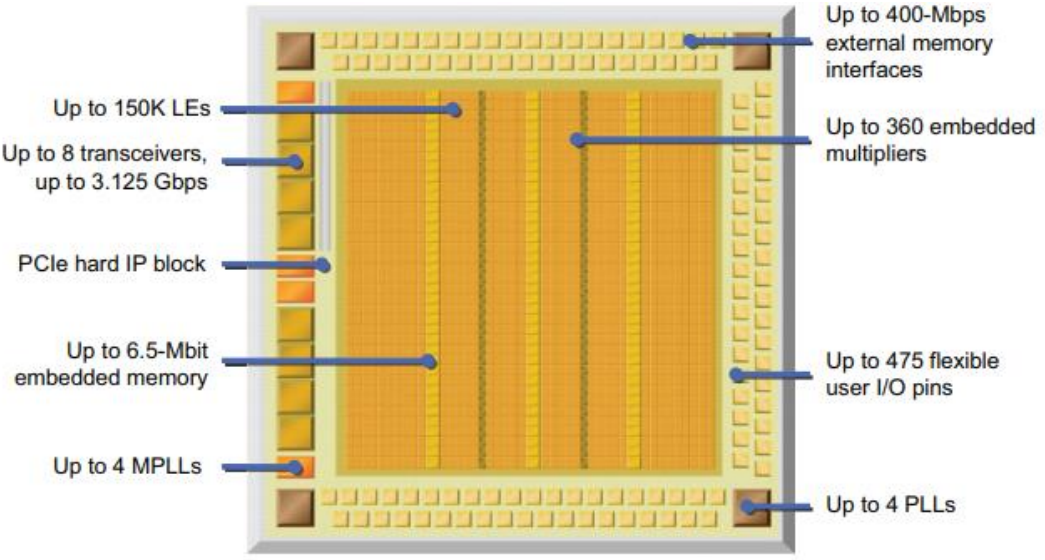


Figure I.7 : Caractéristiques architecturales clés du FPGA Cyclone IV [I.12].

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 :

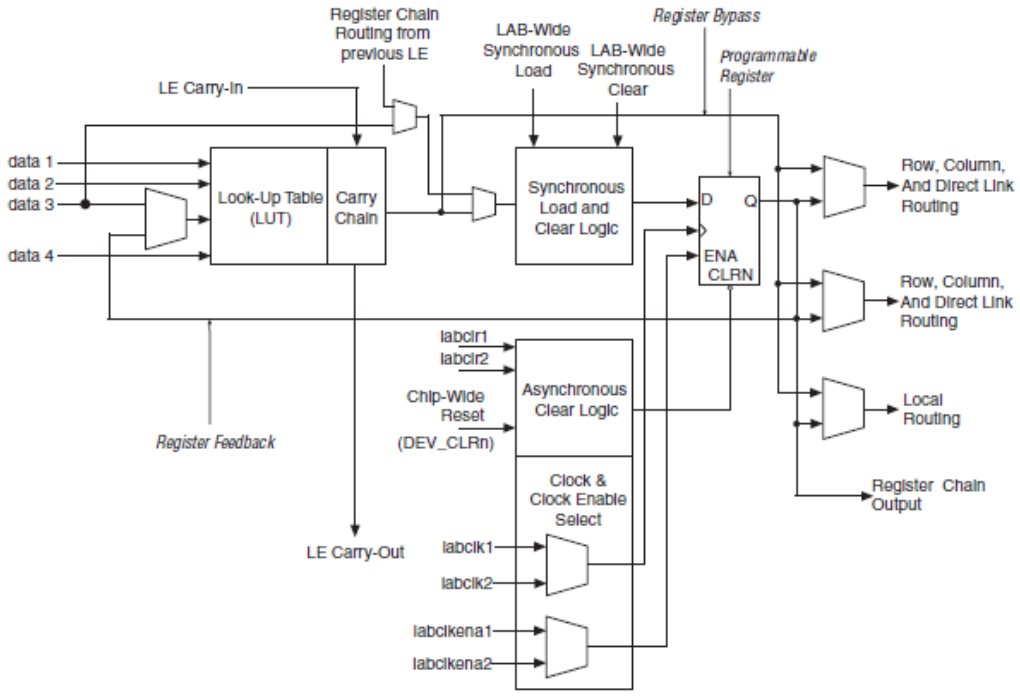


Figure I.8 : Structure d'un Logic Element LE [I.13].

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.

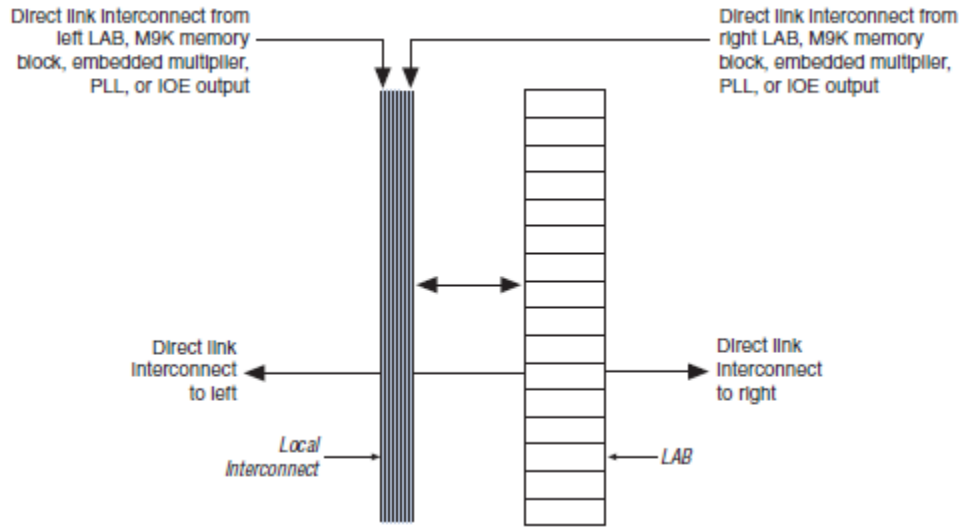


Figure I.9 : Connexions chaînées entre LEs dans un LAB de circuit Cyclone IV [I.13].

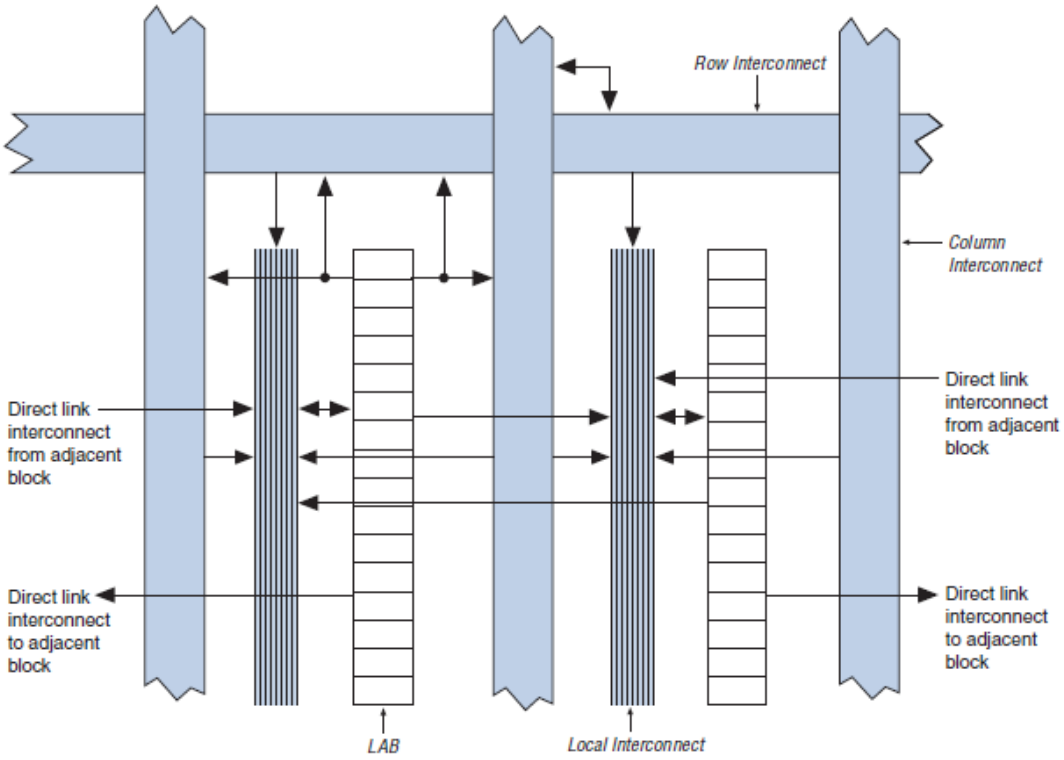


Figure I.10 : Structure LAB de circuit Cyclone IV [I.13].

❖ **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 IV 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 IV 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.2.3.1 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].

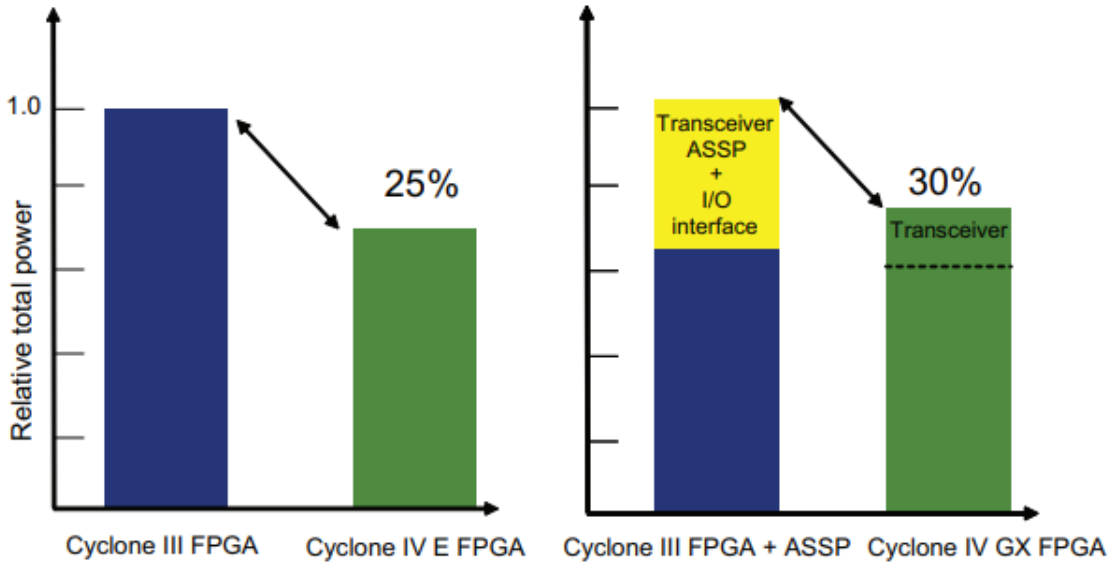


Figure I.11: Puissance Cyclone IV E (à gauche) et Cyclone IV GX (à droite) [I.14].

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 .3.2 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 .3.3 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 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.17].

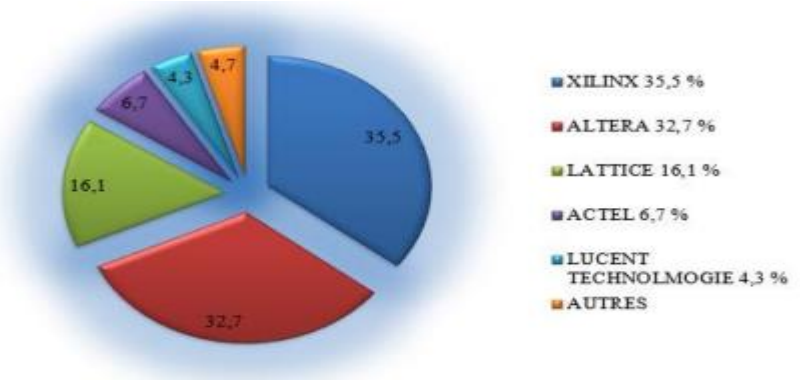


Figure I.13: Statistiques du marché occupé par les vendeurs d'FPGA [I.18].

I.4 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.17], [I.19].

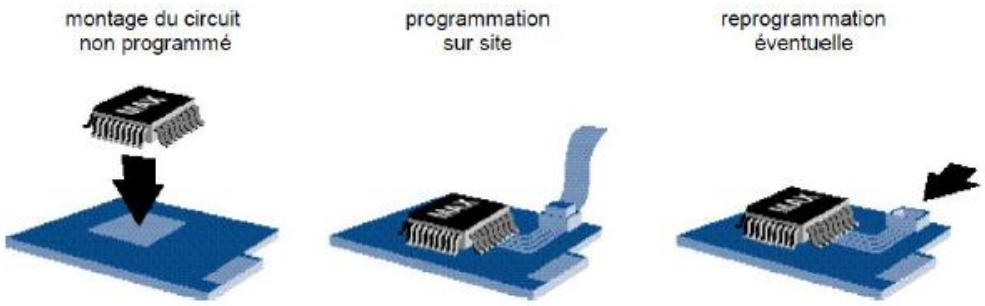


Figure I.14: Reprogrammabilité sur site d'un FPGA.

I.5 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.20].

- Flexibilité et reconfigurabilité

Les FPGA (Field-Programmable Gate Arrays) peuvent être reprogrammés autant de fois que nécessaire après fabrication. Cela permet d'adapter le matériel à de nouvelles exigences sans changer de circuit.

- Parallélisme massif

Contrairement aux processeurs classiques (CPU) qui exécutent les instructions de manière séquentielle, les FPGA peuvent effectuer plusieurs tâches en parallèle, ce

qui accélère considérablement certaines applications comme le traitement du signal, l'IA, ou la vision par ordinateur.

- Temps de développement plus court

Par rapport à une puce ASIC (Application-Specific Integrated Circuit), le FPGA permet de réduire le temps de conception, car il ne nécessite pas de fabrication spécifique : la programmation se fait sur du matériel standard.

- Performance temps réel

Les FPGA offrent des performances déterministes (temps de réponse prévisible), essentielles dans les systèmes embarqués critiques comme l'automobile, l'aéronautique ou les dispositifs médicaux.

- Consommation énergétique optimisée

Bien conçus, certains FPGA peuvent atteindre une consommation énergétique inférieure par rapport à des processeurs généralistes réalisant les mêmes tâches, car les circuits logiques sont optimisés pour la tâche spécifique.

- Coût réduit pour des productions en faible volume

Lorsqu'on ne produit que quelques centaines ou milliers d'unités, utiliser des FPGA est souvent beaucoup plus rentable que concevoir un ASIC.

I.6 Critères de choix du circuit programmable FPGA [I.17], [I.21]

Les FPGA sont développés récemment grâce aux progrès de la technologie VLSI (**V**ery **L**arge **S**cale **I**ntegration), 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.

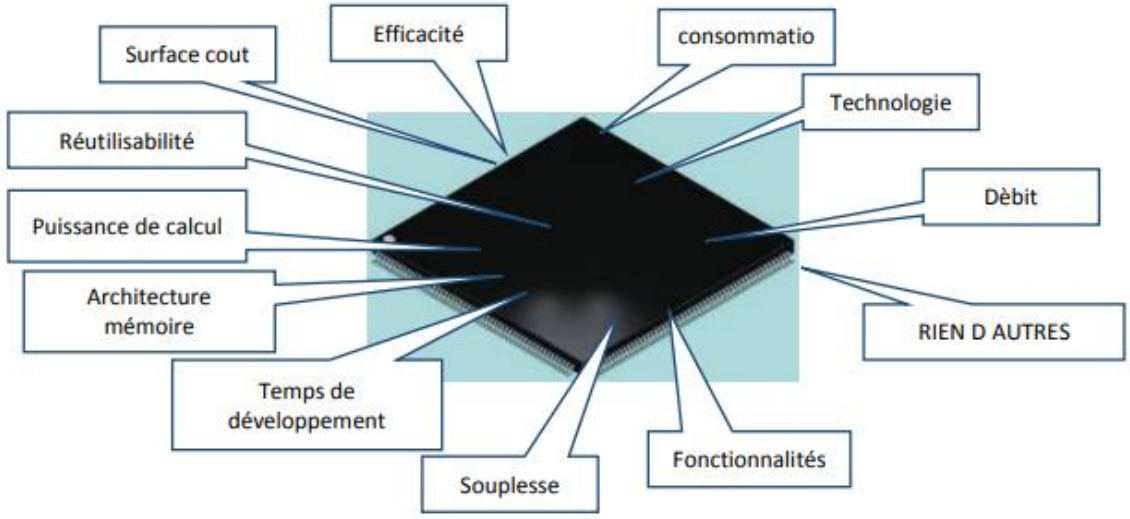


Figure I.15: Critères de choix du circuit logique programmable FPGA.

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.

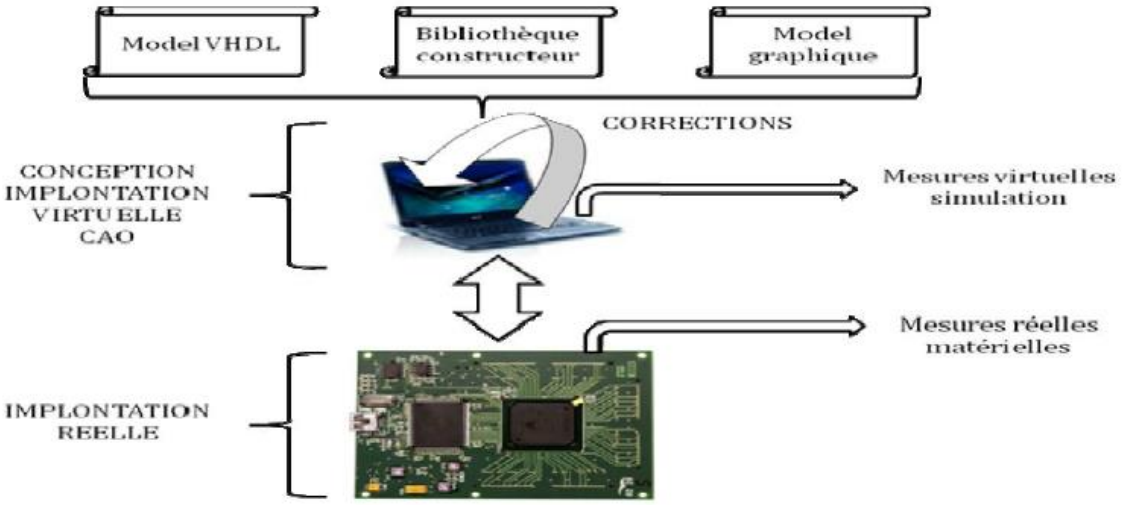


Figure I.16: Mode d'exécution matériel des outils de CAO.

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.

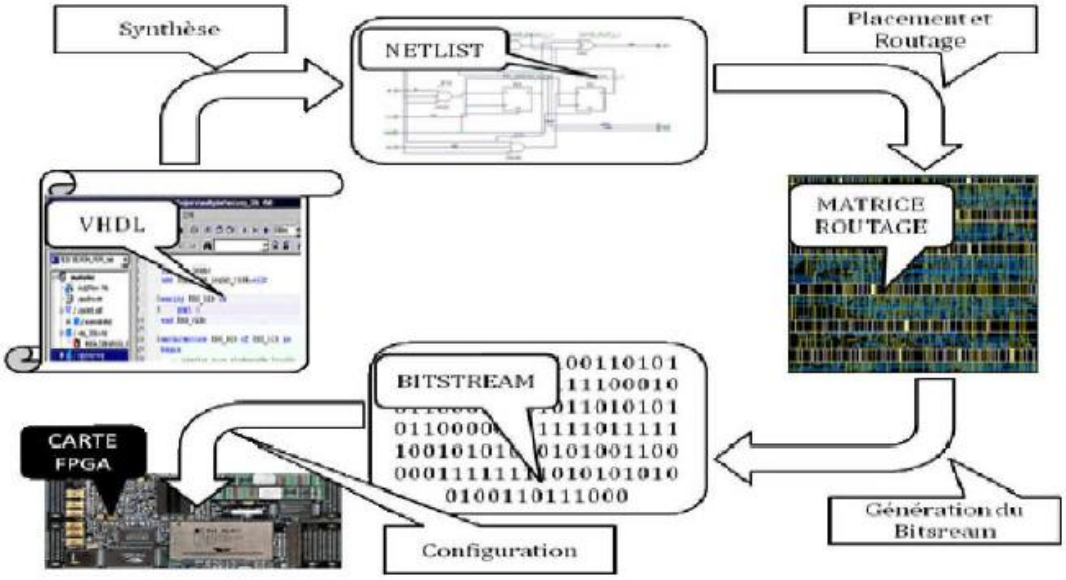


Figure I.17: Cycle de programmation d'un FPGA en utilisant les outils de CAO.

I.8 Principales applications des FPGA

I.8.1 Les applications spécifiques d'un FPGA comprennent [I.23]

✓ Le traitement du signal numérique,

✓ La bioinformatique,

✓ Les contrôleurs de périphériques,

✓ La radio logiciel restreinte (SDR),

✓ La logique aléatoire,

✓ Le prototypage ASIC,

✓ L'imagerie médicale,

✓ L'émulation de matériel informatique,

✓ L'intégration de plusieurs SPLD,

✓ La reconnaissance vocale,

✓ La cryptographie,

✓ Le filtrage et le codage de communication et bien d'autres.

I.8.2 L’application d’un FPGA dans le domaine d’intelligence artificielle

L'utilisation des FPGA (Field Programmable Gate Arrays) dans le domaine de l'intelligence artificielle (IA) est en pleine expansion, offrant des solutions flexibles et performantes pour diverses applications [I.24].

I.8.2.1 Accélération de l'IA avec les FPGA

Les FPGA sont utilisés pour accélérer les charges de travail en IA, notamment grâce à leur capacité de traitement parallèle et leur reconfigurabilité, ce qui les rend adaptés aux applications en périphérie.

Figure I.18 : Illustration représentative de l'accélération de l'intelligence artificielle (IA) à l'aide des FPGA [I.25].

I.8.2.2 Projets de machine learning sur FPGA

Des discussions communautaires offrent des suggestions de projets pour débuter dans le domaine de la machine learning sur FPGA, fournissant des idées et des ressources pour les passionnés.

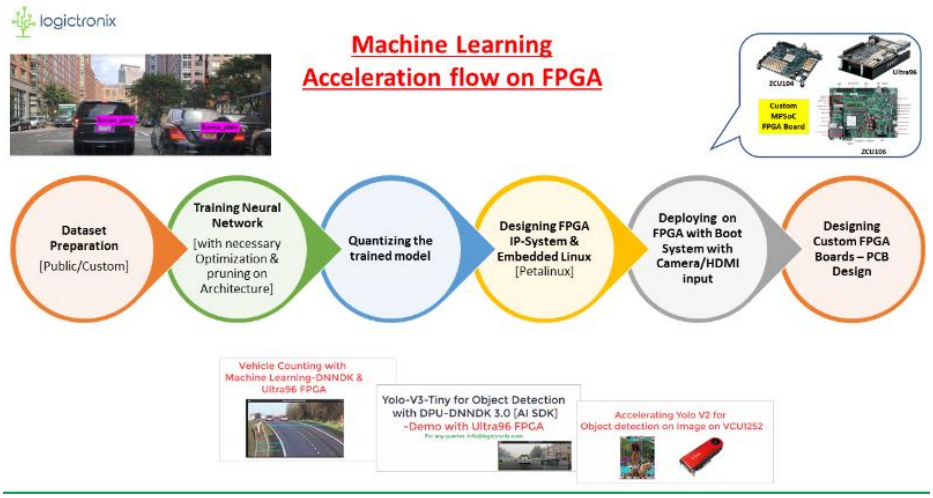


Figure I.19 : Illustrations représentant des projets de machine learning sur FPGA [I.26]

I.8.2.3 Systèmes avancés d'aide à la conduite

Les FPGA sont utilisés pour le traitement en temps réel des données provenant de capteurs tels que les caméras, les radars et les lidars. Ils permettent la détection d'objets, la reconnaissance de voies et la prévention des collisions grâce à leur capacité de traitement parallèle et leur faible latence.

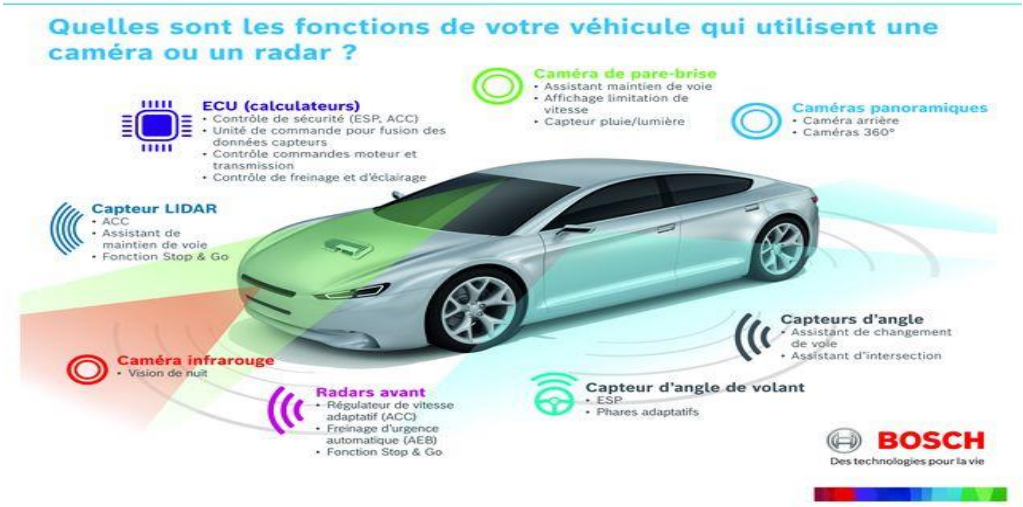


Figure I.20 : Systèmes avancés d'aide à la conduite [I.27].

I.8.2.4 Fusion de capteurs et prise de décision

Dans les véhicules autonomes, les FPGA facilitent la fusion des données de différents capteurs pour une perception précise de l'environnement. Ils exécutent des algorithmes d'IA personnalisés pour la planification de trajectoire et la prise de décision en temps réel, réduisant ainsi la dépendance au Cloud et améliorant la réactivité du système.

I.8.2.5 Contrôle des moteurs et gestion de l'énergie

Dans les véhicules électriques et hybrides, les FPGA assurent un contrôle précis des moteurs, la conversion de puissance et la gestion des batteries, contribuant ainsi à améliorer l'efficacité énergétique et l'autonomie du véhicule.

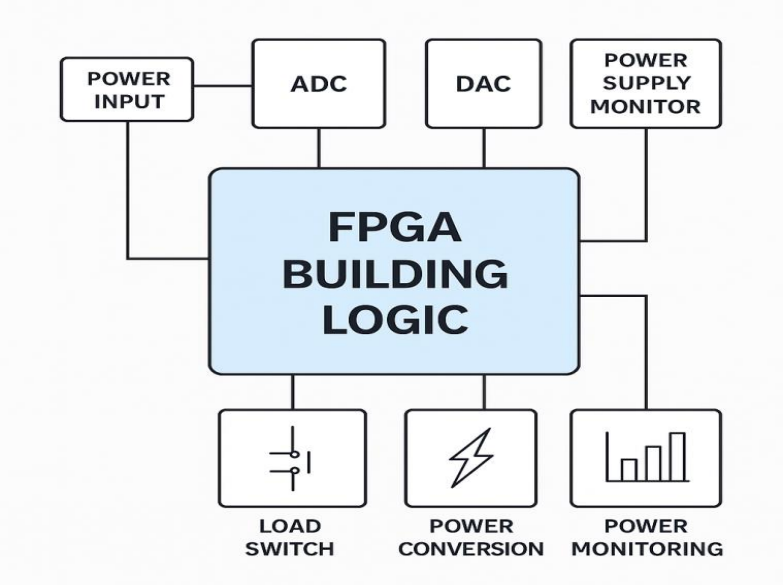


Figure I.21 : Schéma de circuit FPGA pour la gestion de l'énergie .

I.9 Conclusion

Dans ce premier chapitre, nous avons exploré en profondeur le fonctionnement des FPGA (Field Programmable Gate Arrays), en mettant en lumière leur architecture interne, leur structure modulaire ainsi que leur capacité de reconfiguration matérielle. Grâce à une matrice de blocs logiques interconnectables, les FPGA permettent une conception matérielle souple, rapide et optimisée pour des applications spécifiques.

L'intérêt croissant pour ces dispositifs ne cesse de s'amplifier dans de nombreux domaines, en particulier celui de l'intelligence artificielle. Contrairement aux architectures classiques de processeurs ou même de GPU, les FPGA offrent des avantages notables comme une faible latence, une parallélisation massive, une consommation énergétique réduite. Leur intégration dans les systèmes embarqués intelligents, et notamment dans le secteur des véhicules intelligents, confirme leur rôle stratégique dans les applications où la réactivité et l'efficacité sont primordiales. Qu'il s'agisse de traitement d'images en temps réel, de reconnaissance d'objets, ou de prise de décision autonome, les FPGA démontrent leur capacité à répondre aux exigences de l'IA moderne tout en conservant une flexibilité matérielle essentielle.

Ainsi, ce chapitre pose les bases nécessaires pour aborder, dans les sections suivantes, la mise en œuvre pratique de ces concepts dans le cadre de notre projet basé sur un système embarqué à base de FPGA appliqué à un véhicule intelligent.

Pour programmer un FPGA nous avons besoin d'un langage de description matériel, dans le chapitre suivant nous allons présenter le langage VHDL, et les fonctionnalités de base de celui-ci.

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] https://www.shutterstock.com/fr/image-illustration/3d-illustration-fpga-chip-letters-on-2453711307

[I.8] 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.9]Michel PAINDAVOINE, «Traitement des images en temps réel - Applications industrielles », Techniques de l'Ingénieur, Mesures et Contrôle, R 6-720, 1-10.

[I.10] 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.11] 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.12] http://www.electronique-mag.com/article2230.html

[I.13] https://www.intel.fr/content/www/fr/fr/products/details/fpga/cyclone/iv.html

[I.14] https://www.ee.ryerson.ca/~courses/coe608/Data-Sheets/Cyclone-IV.pdf

[I.15] file:///C:/Users/a/Downloads/wp-01113-lowest-system-cost.pdf

[I.16] https://www.ti.com/lit/ug/tidu737/tidu737.pdf

[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.

[I.18] 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.19] Synopsys, «Saber HDL : language-Independant Mixed –Signal Multi-Technology Simulator », Synposys Inc, Etats-Unis d'Amérique 2003.

[I.20] Maxfield, C. (2004). *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*. Elsevier.

[I.21] GUETTAT, ABDELGHANI, « Conception et Implémentation d'un Corrélateur Numérique sur FPGA », Thèse de doctorat. USTO, 2012.

[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] Sze,V., Chen,Y. H.,Yang, T.J., & Emer,J. S. (2017). *Efficient Processing of Deep Neural Networks: A Tutorial and Survey.* Proceedings of the IEEE, 105(12), 2295-2329.

[I.24]https://onedrive.live.com/?authkey=%21ABZ%5Fj27B4HZ3Adg&id=B2CDC3A30980D5BD%2182722&cid=B2CDC3A30980D5BD

[I.25]https://www.intel.fr/content/www/fr/fr/software/programmable/p4-suite fpga/overview.html

[I.26]https://www.google.com/imgres?imgurl=https%3A%2F%2Flookaside.fbsbx.com%2Flookaside%2Fcrawler%2Fmedia%2F%3Fmedia_id%3D853693281768738&tbnid=Znayawd2ymYYDM&vet=1&imgrefurl=https%3A%2F%2Fwww.facebook.com%2Fgroups%2FFPGAASICVLSI%2Fposts%2F1122145418134864%2F&docid=Vpu4Po6vphDHGM&w=1280&h=720&hl=fr-CH&source=sh%2Fx%2Fim%2Fm1%2F4&kgs=3907a83d7ea3757d

[I.27] https://images.app.goo.gl/hn82qRru5SM17zXG6

Chapitre II

Présentation de langage VHDL & RFID

II.1 Introduction

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].

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

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 première partie de ce chapitre.

Dans la deuxième partie de ce chapitre, nous souhaitons mettre en évidence le principe physique du système RFID. Nous rappellerons les différents éléments d'un système RFID, nous décrirons ensuite le principe de fonctionnement et les grandes familles de cette technologie RFID. Nous attirons l'attention sur quelques applications de la technologie RFID.

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 [II.6], [II.7]

Le VHDL est un langage de description matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique indépendamment d'un fournisseur d'outils. Ainsi, techniquement, il est incontournable car c'est un langage puissant, moderne et qui permet une excellente lisibilité, une haute modularité et une meilleure productivité des descriptions. Il permet de mettre en œuvre les nouvelles méthodes de conception.

En outre, le développement du l'ensemble synthétisable du langage VHDL est de mieux en mieux défini. Cependant, la majorité des outils de synthèse sont compatibles avec cette norme, ce qui reflète un véritable standard pour la synthèse automatique avec le langage VHDL.

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:

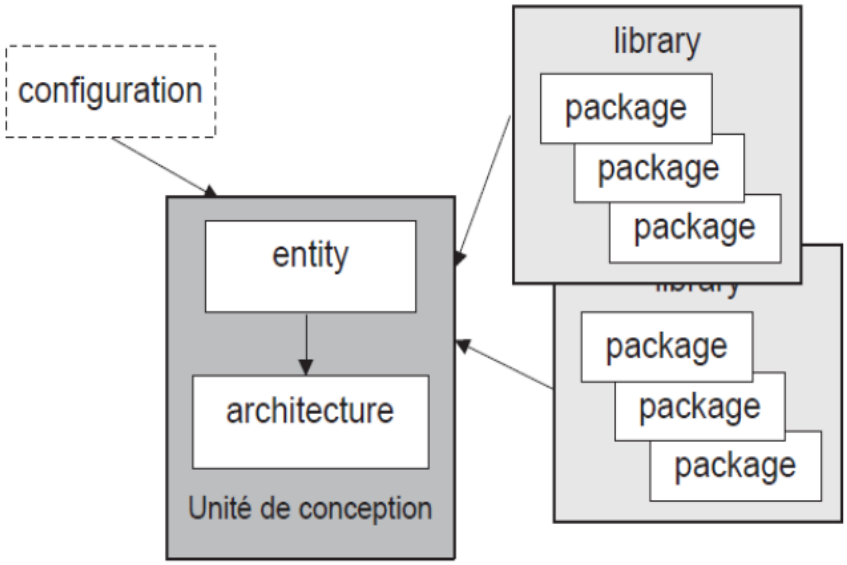


Figure II.1 : Structure de base d'un module VHDL [II.7].

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.8].

II.2.3.1.2 Déclaration des librairies [II.9], [II.10]

Toute description VHDL utilisée pour la synthèse à besoin de bibliothèque. Tout d'abord la librairie principale qui est en générale IEEE (institut of Electrical and Electronics Engineers). Elle contient les définitions des types de signaux électroniques, des fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques…

Ensuite il vient le mot clé ‹ use › qui fait indiqué le package de la librairie, après on écrit le nom du package. Enfin le .all qui indique l'utilisation de tout ce qui se trouve dans ce package. Elle est déclarée comme l'exemple suivant :

- Library ieee.std_logique_1164.all ;
- Use ieee.std.numeric_std.all;
- Use ieee.std_logique_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.

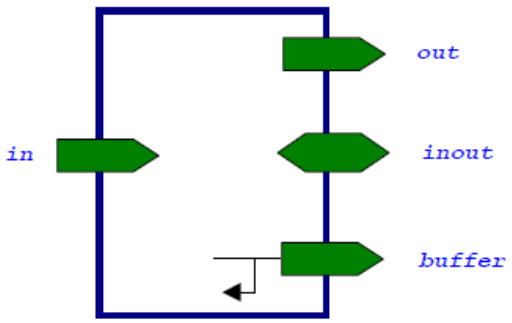


Figure II.2: Représentation de sens de signal.

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.

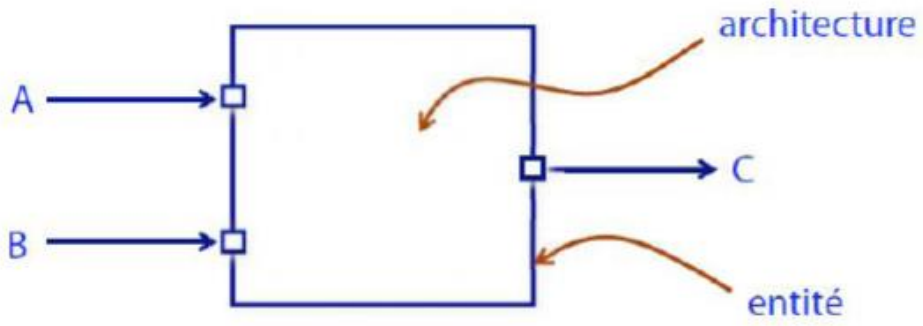


Figure II.3 : Représentation de l’entité et l’architecture.

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.11]

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.11]

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.2.3.2 Relation entre une structure VHDL et un circuit numérique

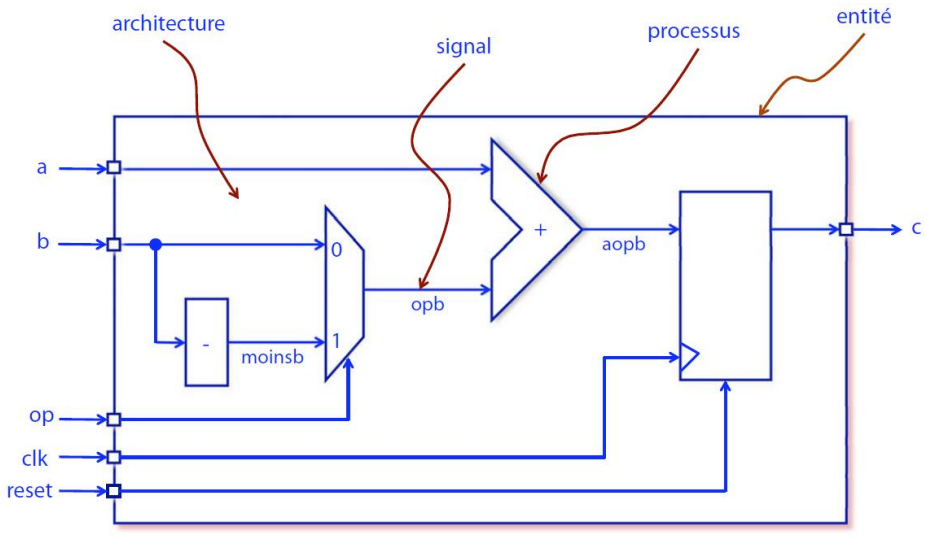


Figure II.4 : Relation entre les composants d'un VHDL et un circuit numérique.

- 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.12].

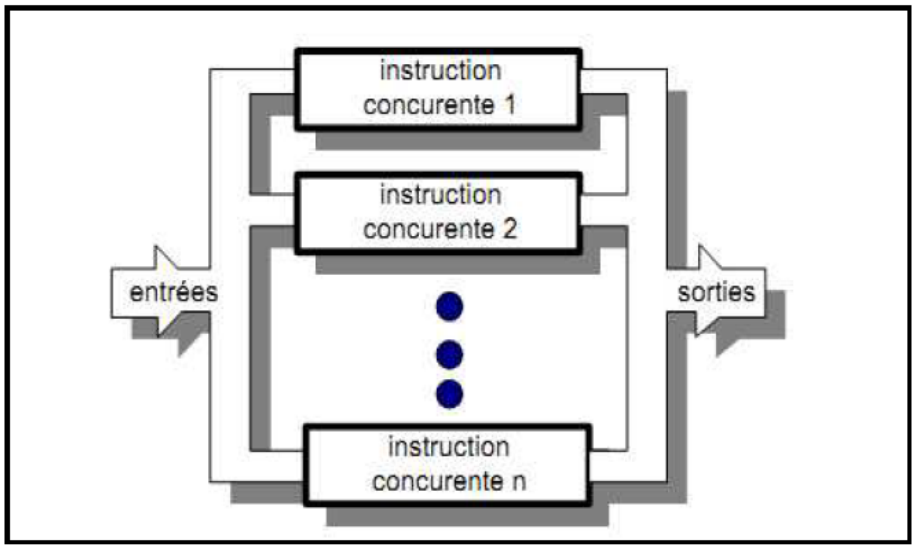


Figure II.5: Instructions en mode concurrent.

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.13].

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.14].

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.

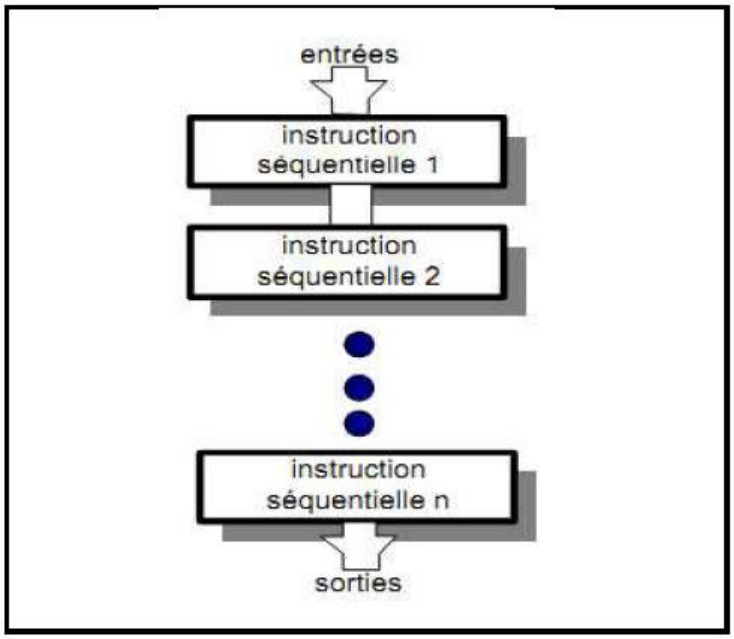


Figure II.6. Instructions en mode séquentiel.

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.11]

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.15].

« 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.16]

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.14].

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.16].

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.17]

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.17]

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.16].

II.3 Différences entre VHDL et un langage de programmation [II.11]

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.4 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.5 Développement d'un projet en VHDL [II.15], [II.16]

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.

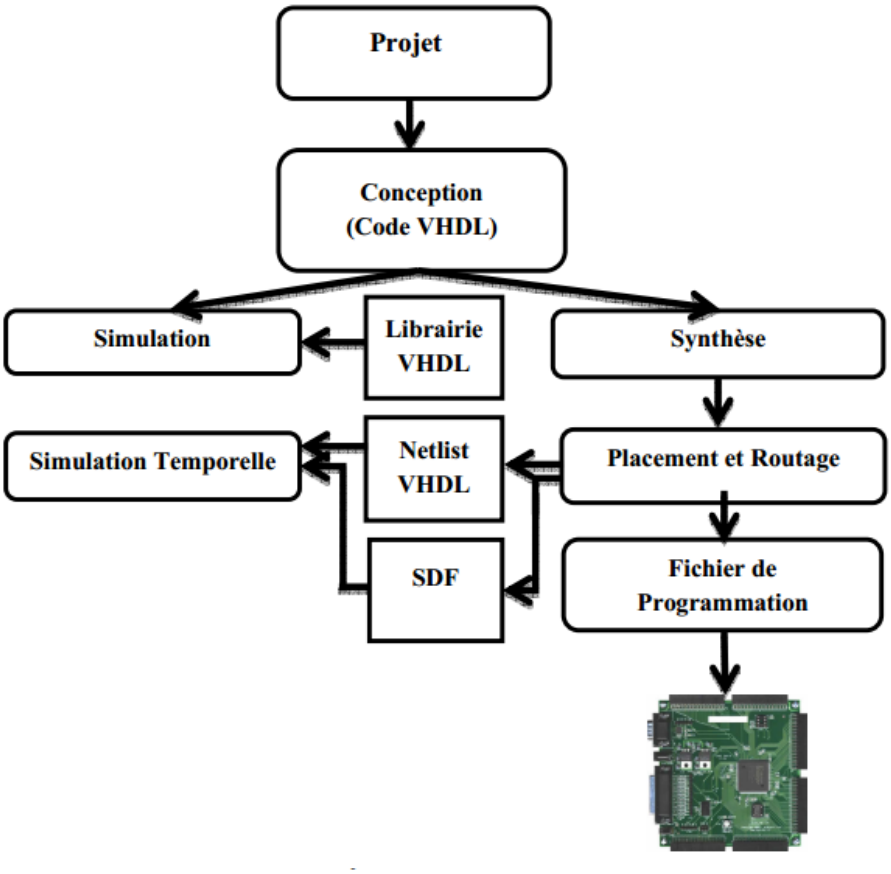


Figure II.7: Différentes étapes de l'implémentation sur FPGA.

II.6 Technologie RFID

L'identification par radiofréquence, mieux connu comme la RFID, est une technologie intelligente qui est très performante, flexible et convient bien pour des opérations automatiques. La RFID est une méthode d'identification automatique qui utilise les ondes radio pour lire les données contenues dans des dispositifs appelés étiquettes ou Tags RFID. Elle combine des avantages non disponibles avec d'autres technologies d'identification comme les codes à barres. La RFID peut être fourni en lecture seule ou en lecture/écriture, sans contact, peut fonctionner sous une variété de conditions environnementales, permet de stocker une grande quantité d'information et fournit un haut niveau de sécurité. La technologie RFID est utilisée

pour surveiller, identifier et suivre des objets, des animaux et des personnes à distance en utilisant les ondes radio. Les Tags RFID sont plus chères que les codes-barres, mais le rapport bénéfice-coût est généralement favorable.

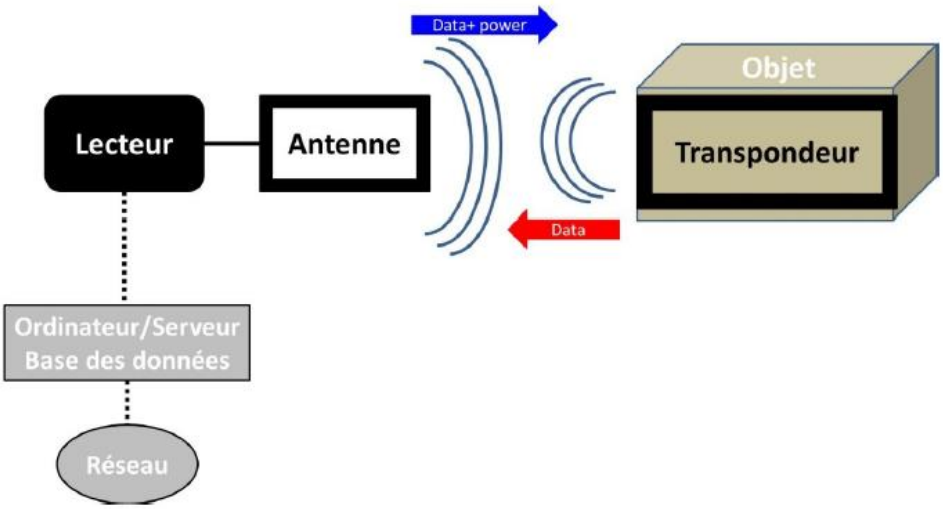


Figure II.8: Eléments d'un système RFID.

II.6.1 Définition générale d'un système RFID

Un système RFID est un système de communication sans fil dédié principalement au domaine de l'indentification. Il permet par exemple de réaliser le suivi des animaux, des articles, des produits et des personnes. Une station de base va permettre d'identifier des tags ou transpondeurs qui sont généralement positionnés sur les cibles à identifier [II.18-II.22], [II.23]. Dans sa configuration de base, un système RFID est composé d'un lecteur, d'un transpondeur et d'un terminal qui permet l'exploitation des données collectées (figure II.8).

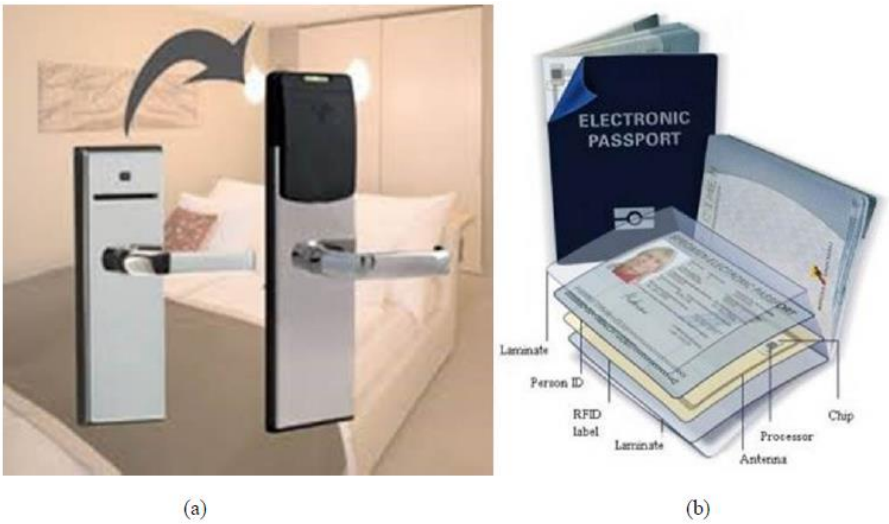


Figure II.9: (a) lecteur RFID pour le contrôle d'accès, (b) passeport biométrique utilisant la RFID

Le lecteur envoie des requêtes aux tags RFID pour récupérer des données stockées dans leur mémoire. Le lecteur peut également procéder à une écriture d'information dans le tag. Le lecteur et le tag sont équipés d'antennes qui doivent s'adapter à l'environnement de la communication RFID. On note à ce propos que cet environnement change au cours du temps, il peut être l'espace libre ou présenter des obstacles de différents types comme des, liquides ou encore des matériaux métalliques. Le tag, généralement télé-alimenté par le signal du lecteur, génère en premier lieu un code permettant d'identifier l'objet sur lequel il est déposé (figure II.8). L'information contenue dans le tag RFID peut être accessible au grand public comme le prix ou encore les caractéristiques des produits. Elle peut être aussi d'ordre privée et par suite restreinte aux fabricants ou aux services qui exploitent la technologie RFID. On peut citer l'exemple des passeports biométriques, le contrôle d'accès et la traçabilité des produits le long des chaines de production (figure II.9). Dans ce cas de figure, un niveau de sécurité supérieur est indispensable du coté réseau pour la gestion des données. En effet, au niveau de la communication RFID, les requêtes échangées entre le lecteur et les tags doivent dans ce cas être sécurisées. La RFID doit cohabiter d'un point de vue spectral avec d'autres technologies sans fil. Pour la RFID, nous distinguons les bandes suivantes (figure II.10) :

- La bande LF à 125kHz et 143 kHz
- La bande HF à 13.56 MHz,
- La bande UHF 860-960 MHz,
- La bande SHF à 2.45 et 5.8 GHz

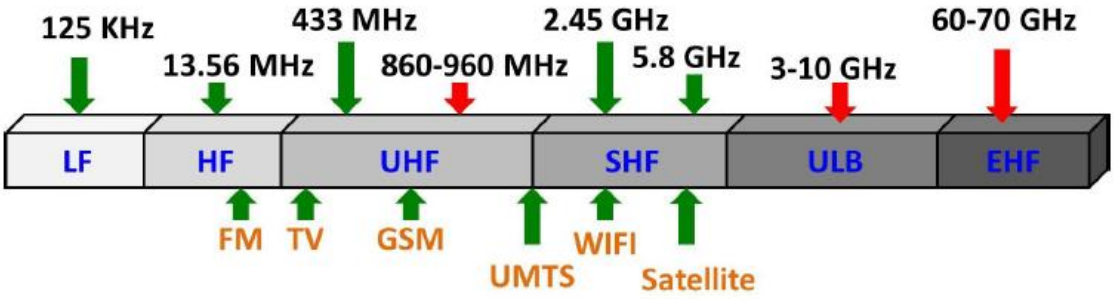


Figure II.10 : Bandes de fréquence RFID.

II.6.2 Principe de fonctionnement d'un système RFID

Un système d'Identification par Radio Fréquence se compose de deux éléments principaux: un Tag et un lecteur. Le Tag contient toutes les données relatives à l'objet qui l'identifie de façon unique. Les données, stockées dans une puce électronique « chip », peuvent être lues grâce à une antenne qui reçoit et transmet des signaux radio vers et depuis le lecteur ou interrogateur.

Le lecteur, fixe ou tenu à la main, est le dispositif qui est en charge de la lecture des Tags RFID situées dans son champ de lecteur et capable de convertir les ondes radio de Tag en un signal numérique qui peut être transféré à un PC. La figure II.11 décrit le fonctionnement général d'un système d'identification par radiofréquence [II.24].

Figure II.11 : Fonctionnement général d'un système RFID.

II.6.3 Domaines d'application de la technologie RFID

Les applications de la technologie RFID sont variées et englobent différents domaines.

Voici certains des domaines d'application majeurs de la technologie RFID :

- **Contrôle d'accès et sécurité**

Dans les systèmes de contrôle d'accès, la technologie RFID joue un rôle essentiel en assurant la sécurité des entrées dans les bâtiments ou en identifiant les véhicules entrant dans les parkings sécurisés grâce à des badges magnétiques et des cartes de sécurité [II.25].

- **Santé publique et industrie médicale**

Il est possible de trouver une application similaire dans le domaine de la santé, en utilisant le même principe, mais pour les médicaments. Cependant, il est possible d'aller plus loin en intégrant une étiquette RFID dans le bracelet des patients, afin que les médecins puissent la scanner et connaître immédiatement le patient et sa pathologie [II.26].

- **Suivre des documents dans les bibliothèques**

Les étiquettes RFID offrent la possibilité d'identifier chaque document de manière distincte, ce qui simplifie le suivi et la gestion des transactions d'emprunt/retour [II.27],[II28].

II.7 Conclusion

Dans ce chapitre, nous avons présenté de manière approfondie le langage de description matérielle VHDL ainsi que la technologie RFID, deux éléments fondamentaux dans la conception de systèmes embarqués modernes. Le VHDL s'est révélé être un outil puissant permettant de modéliser, simuler et implémenter des circuits logiques complexes sur FPGA, offrant ainsi une flexibilité et une précision indispensables dans le développement matériel. En parallèle, la technologie RFID, en tant que solution de communication sans fil basée sur l'identification par radiofréquence, joue un rôle crucial dans l'automatisation, la sécurité et la traçabilité dans divers domaines, notamment dans les systèmes d'accès intelligents. L'intégration de ces deux technologies dans un même projet permet de tirer parti des performances matérielles du VHDL et de la connectivité sans fil du RFID, ouvrant la voie à des applications embarquées intelligentes, efficaces et évolutives.

Dans le chapitre suivant nous allons programmer un FPGA en VHDL afin de lui permettre de commander un véhicule omnidirectionnel qui contient la base des applications de l'intelligence artificielle.

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

[II.9] 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.10] http://hdl.telecom-paristech.fr/vhdl_structurel.html. COURS EN LIGNE VHDL (site consulté le 16/02/2019).

[II.11] 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.12] Philippe LARCHER, « VHDL Introduction à la synthèse logique », 2000 .

[II.13] WEBER, Jacques et MEAUDRE, Maurice, « *Circuits numériques et synthèse logique: un outil: VHDL* », Jacques Weber, 1995.

[II.14] 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.15] 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.16] 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.17] A. Abdelaziz, "Compression d'images par fractale", Thèse de magister en informatique, Université de Batna, 2002.
[II.18] K. Finkenzeller, *RFID Handbook: Radio-Frequency Identification Fundamentals andApplications.*John Wiley & Sons, Ltd, *1999*.
[II.19] S. M. Roy et N. C. Karmakar, « Introduction to RFID Systems », *Handb. SmartAntennas RFID Syst.*, p. 13, 2011.
[II.20] I. Mayordomo, R. Berenguer, I. Fernandez, I. Gutierrez, W. Strauss, et J. Bernhard,« Simulation and measurement of a long-range passive RFID system focused on readerarchitecture and backscattering communication », in *Microwave Conference, 2008.EuMC 2008. 38th European*, 2008, p. 1058-1061.
[II.21] I. Mayordomo, A. Ubarretxena, D. Valderas, R. Berenguer, et Í. Gutiérrez, « Design and analysis of a complete RFID system in the UHF band focused on the backscatteringcommunication and reader architecture », in *RFID Systems and Technologies (RFID SysTech), 2007 3rd European Workshop on*, 2007, p. 1-6.
[II.22] D. J. Hind, « Radio frequency identification and tracking systems in hazardous areas », 1994.
[II.23] Mossaab Daiki, "Contribution au développement d'antennes lecteurs champ proche pour les systèmes RFID UHF passifs », Thèse de doctorat en optique et radiofréquence de l'Université de GRENOBLE, 2006.
[II.24] MERAHI MIRA & GUERROUDJ BOUCHRA, « Conception d'antennes imprimées pour Identification Radio Fréquence RFID UHF », Mémoire De fin d'étude pour l'obtention du diplôme de Master en Télécommunications, UNIVERSITE Dr. TAHAR MOULAY – SAIDA, 2020.

[II.25] https://www.simons-voss.com/fr/ce-quil-faut-savoir/technologie-rfid.html
[II.26] https://smartmakers.io/fr/rfid-anwendungen-in-verschiedenen-branchen/
[II.27] BASSOU. A, ZIANE. A, MERIHIL. K. LARBI Omar, «Gestion d'une Bibliothèque Intelligente basée sur le Système RFID», Université Tahri Mohammed de Béchar.
[II.28] BENAISSA Rayham & FOUTEM Rabab, « Etude d'un système de présence basé sur le module RFID », MEMOIRE Présenté pour l'obtention du diplôme de MASTER En : Electronique, Université Aboubakr Belkaïd – Tlemcen – , 2024.

Chapitre III

Résultats de conception et réalisation d'un véhicule utilisant un FPGA

III.1 Introduction

Après avoir posé les bases théoriques des FPGA et du langage VHDL dans les chapitres précédents, ce chapitre est consacré à la réalisation pratique du véhicule omnidirectionnel et à sa manipulation intelligente via différentes interfaces matérielles et logicielles. L'objectif est de concrétiser l'architecture d'un système embarqué complet, capable de réagir à des commandes extérieures, de percevoir son environnement, et d'agir de façon autonome.

La première étape a consisté à implémenter un système d'affichage sur écran LCD, permettant une interaction visuelle entre le véhicule et l'utilisateur. Par exemple, lors de la mise sous tension du système, le message « **WELCOME** » s'affiche sur l'écran LCD, indiquant que le système est prêt à fonctionner. Ensuite, cet écran est utilisé pour afficher en temps réel les valeurs mesurées par le capteur ultrasonique HC-SR04, comme la distance séparant le véhicule d'un obstacle situé devant lui. Ces informations sont actualisées en continu et permettent à l'utilisateur de suivre l'état du véhicule en fonctionnement.

La deuxième partie du projet concerne la commande manuelle du véhicule via une application mobile Bluetooth, développée avec MIT App Inventor. Grâce au module HC-05, le véhicule reçoit les ordres de déplacement (avant, arrière, gauche, droite, arrêt… etc.) directement depuis le Smartphone. Cette commande permet de tester la mobilité omnidirectionnelle du robot dans toutes les directions, en profitant de la flexibilité offerte par la configuration de ses roues.

La troisième partie aborde la mise en œuvre du mode autonome, une étape essentielle dans le cadre des applications d'intelligence artificielle embarquée. Le véhicule utilise ici son capteur ultrasonique pour détecter les obstacles : s'il identifie un objet à moins de 30 cm, il active automatiquement le buzzer pour signaler le danger, puis effectue une rotation vers la gauche pour éviter la collision, sans intervention humaine. Ce comportement est codé entièrement en VHDL et illustre une forme de prise de décision basée sur l'environnement, marquant les premiers pas vers un système intelligent.

En complément de ces fonctionnalités, une fonction de sécurité par contrôle d'accès RFID a été ajoutée. Le module PN532, compatible avec les technologies

RFID et NFC, est utilisé pour identifier une carte ou un tag autorisé. Lorsqu'une carte valide est détectée, le FPGA active un servomoteur qui ouvre automatiquement la porte du véhicule. Ce mécanisme, entièrement réalisé en VHDL et reposant sur une communication UART entre le FPGA et le module RFID, offre un accès sécurisé au véhicule tout en illustrant l'intégration d'interfaces externes dans un système embarqué.

Ainsi, ce chapitre présent la construction complète d'un véhicule intelligent, interactif et autonome, mettant en œuvre les différentes technologies étudiées dans les chapitres précédents et démontrant le potentiel des FPGA dans des applications réelles et évolutives.

III.2 Partie Hardware : présentation des outils électroniques

Pour notre projet, nous avons travaillé pour la conception et de la réalisation d'un véhicule omnidirectionnel intelligent basé sur un FPGA, plusieurs outils électroniques ont été intégrés afin d'assurer un fonctionnement autonome, sécurisé et interactif. Cette section présente les différents composants matériels utilisés dans notre projet, chacun jouant un rôle spécifique dans le système global. Le module Bluetooth permet la communication sans fil avec une application mobile pour le contrôle manuel, tandis que le capteur à ultrasons assure la détection d'obstacles pour éviter les collisions. L'écran LCD affiche en temps réel des informations essentielles, telles que la distance détectée ou l'état du système. Le lecteur RFID est utilisé pour l'identification sécurisée, par exemple pour l'activation d'un accès. Enfin, un servomoteur est intégré pour effectuer des actions mécaniques, comme l'ouverture ou la fermeture d'une porte. L'ensemble de ces outils a été soigneusement sélectionné et programmé en VHDL pour garantir une interaction fluide entre les différentes fonctionnalités du véhicule.

III.2.1 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 III .1 présente la carte de développement de la série OMDAZZ.

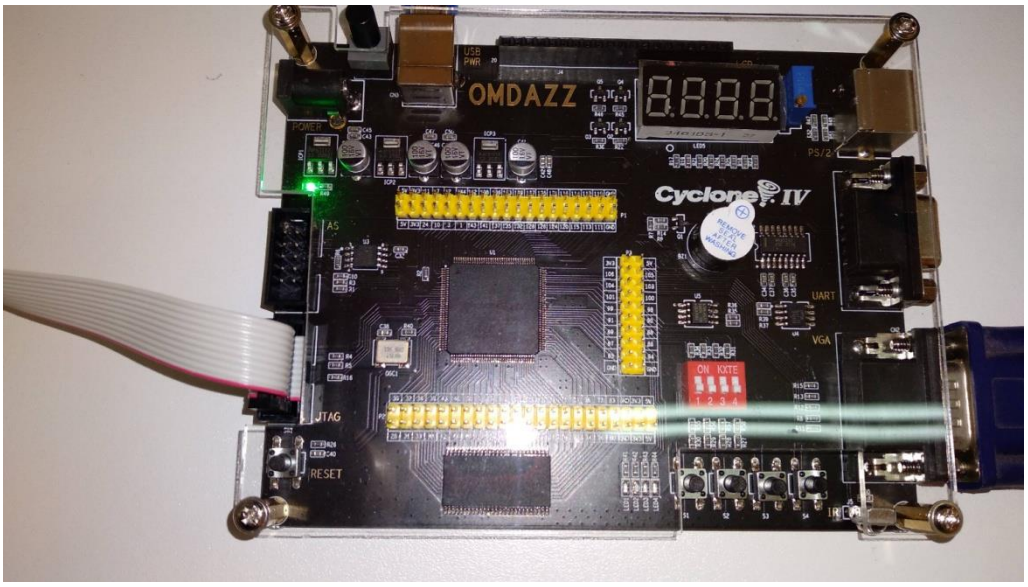


Figure III.1 : Carte de développement FPGA Altera Cyclone IV EP4CE10E22C8.

III.2.1.1 Caractéristiques de notre composant FPGA EP4CE10E22C8 [III.1]

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

Tableau III.1 : Caractéristiques de FPGA EP4CE10E22C8.

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

III.2.1.2 Différentes parties de la carte OMDAZZ [III.1]

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 III.2), construite autour d'un FPGA ALTERA Cyclone IV (10320 LogicElements 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.

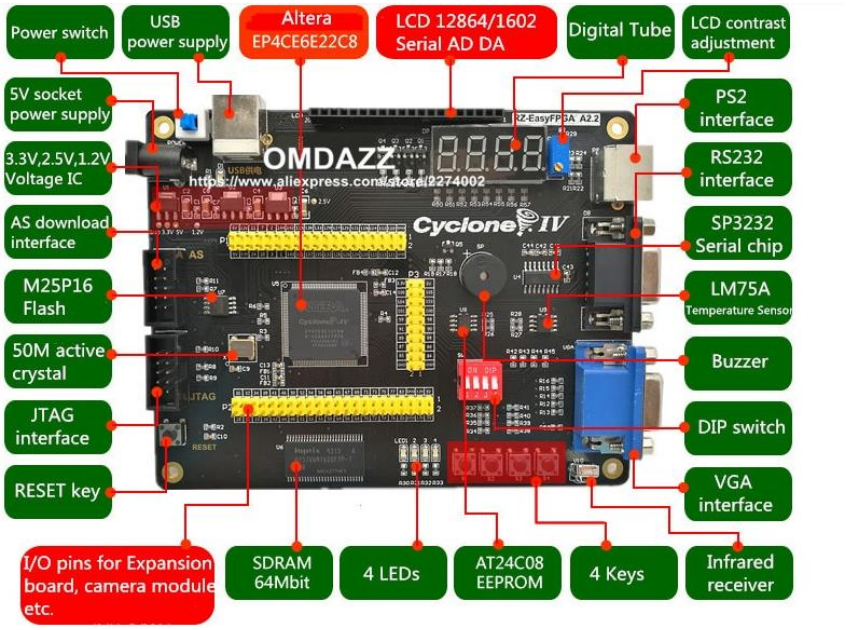


Figure III.2: Constitution de la carte de développement FPGA Altera Cyclone IV.

III.2.2 Module de LCD 16x2

Les afficheurs à cristaux liquides, autrement appelés afficheurs LCD (**L**iquid**C**rystal **D**isplay), sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement, capable d'afficher 16 caractères sur 2 lignes. Ils consomment relativement peu (de 1 à 5 mA), s'utilisent avec beaucoup de facilité. Ils sont très utilisés dans les montages à microcontrôleur. Ils peuvent aussi être utilisés lors de la phase de développement d'un programme, car on peut facilement y afficher les valeurs de différentes variables [III.2].

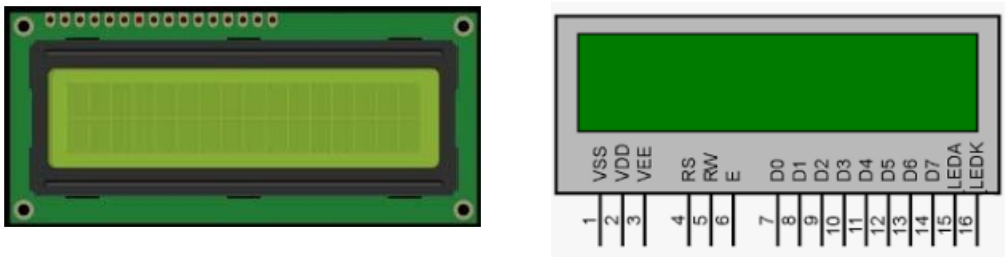


Figure III.3 : Afficheur LCD 16x2.

III.2.2.1 Caractéristiques

- Lignes × Colonnes : 2 lignes × 16 caractères
- Type : Alphanumérique
- Tension d'alimentation : 5V DC (certains modèles acceptent 3.3V)
- Interface : Parallèle (4 bits ou 8 bits)
- Rétro éclairage : Oui (LED, souvent blanc ou bleu)
- Contraste : Réglable via potentiomètre (V0)

III.2.2.2 Brochage (interface 16 broches)

- VSS : Masse (GND)
- VDD : Alimentation +5V
- V0 : Contrôle du contraste (via potentiomètre)
- RS : Registre Select (commande ou données)
- RW : Read/Write (lecture/écriture)
- E : Enable (front montant pour valider)

- D0–D7 : Bus de données (4 ou 8 bits)
- LED+ : Alimentation rétro éclairage
- LED- : Masse rétro éclairage

III.2.2.3 Fonctionnement

L'afficheur LCD est piloté par une séquence de commandes et de données envoyées via les broches de contrôle. Le fonctionnement se divise en deux parties :

- Commandes : permettent de configurer l'afficheur (effacer l'écran, retour à la ligne, position du curseur, etc.)
- Données : les caractères ASCII à afficher.

III.2.2.4Câblage d'afficheur LCD 16x2

Pour connecter l'afficheur LCD, vous reliez les 16 broches de l'afficheur aux 16 broches du FPGA.

La connexion entre le FPGA et l'afficheur LCD 16x2 est illustrée sur le schéma suivant :

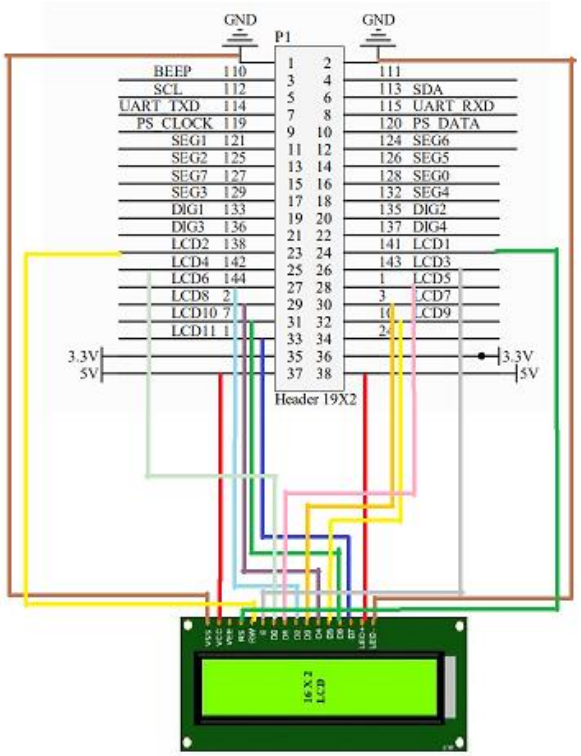


Figure III.4:Câblage d'afficheur LCD avec le FPGA.

III.2.3 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.3].

Figure III.5: Capteur ultrason HC-SR04.

III.2.3.1 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.2.3.2 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.2.3.3 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.

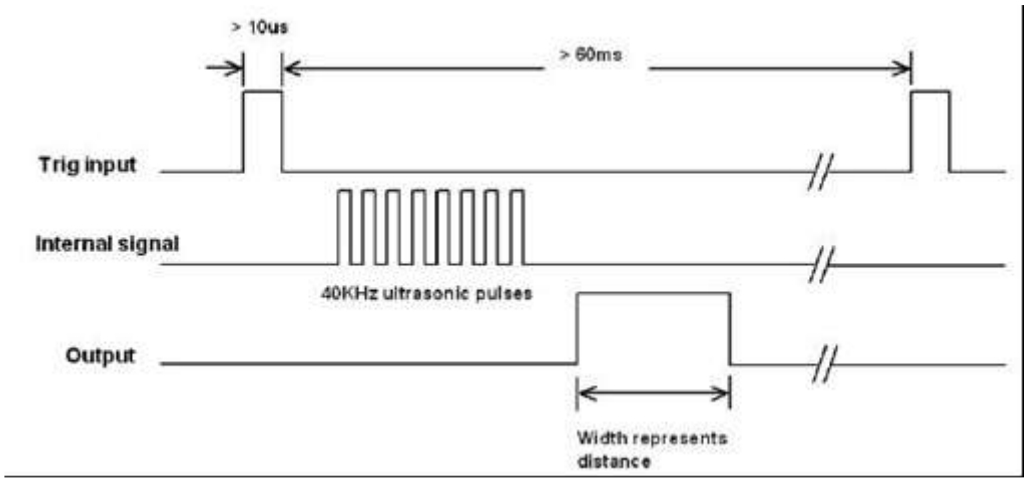


Figure III.6: Diagramme de fonctionnement du capteur ultrason [III.3].

III.2.3.4 Spécifications et limites

Tableau III.2: Spécifications et limites du capteur ultrason [III.3].

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.2.3.5Câblagede détecteur a ultrason avec FPGA

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

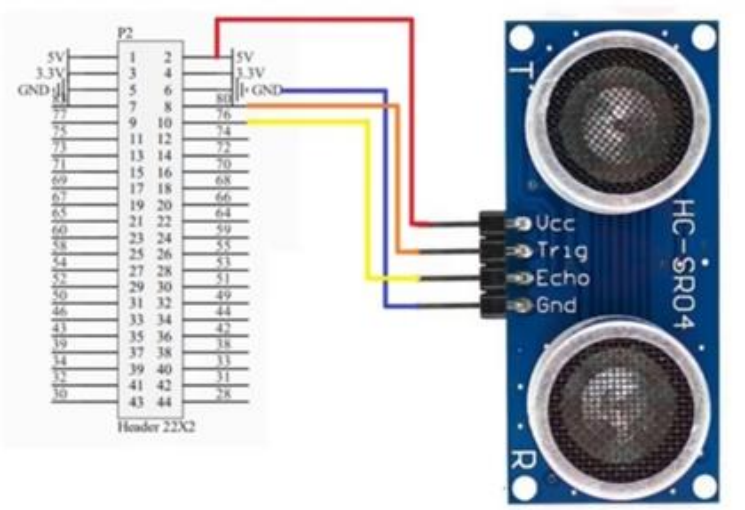


Figure III.7 :Câblage de HC-SR 04 avec FPGA.

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.2.4 Moteurs DC

Un moteur électrique à courant continu (MCC) est un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique. Il est très utilisé en industrie est nécessite une régulation précise de la vitesse de rotation.

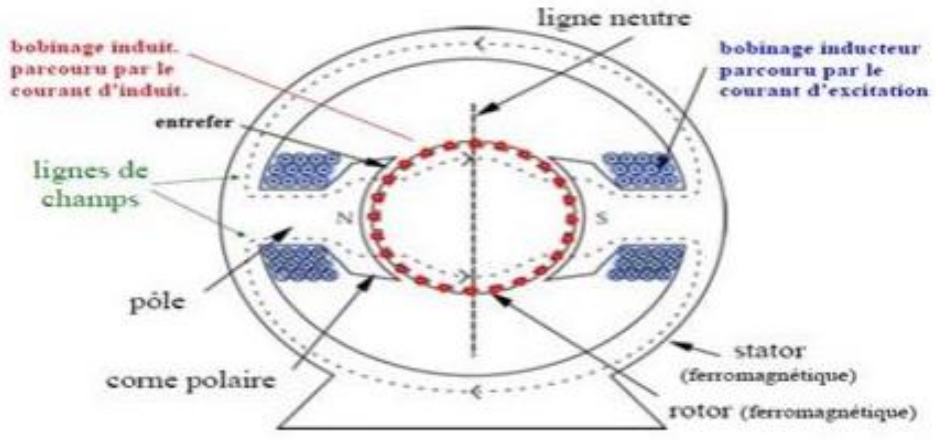


Figure III.8 : Structure générale d'un moteur DC.

III.2.5 Driver L298N

Ce circuit nous permet de contrôler la vitesse et la direction de deux moteurs à courant continu, ou contrôler un moteur pas à pas bipolaire facilement. Le module de pont en H L298N, peut être utilisé avec des moteurs ayant une tension comprise entre 5 et 35 V en courant continu.

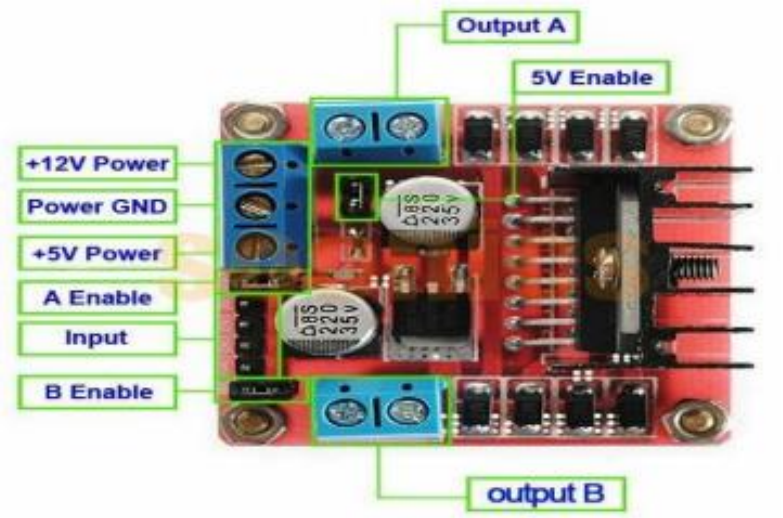


Figure: III.9: Brochage du Module L298N double ponts en H.

III.2.5.1 Caractéristiques

Mode de travail: lecteur de pont H (double canal)

Puce de commande principale: L298N

Tension logique: 5V

Tension d'entraînement: 5V-35V

Courant logique: 0mA-36mA

Courant d'entraînement: 2A (pont simple MAX)

Température de stockage: -20 ° à + 135 °

Max. Puissance: 25W

III.2.5.2 Câblage de FPGA, moteurs DC et L298N

La figure suivante représente la connexion entre notre FPGA, les moteurs DC et les deux drivers L298N:

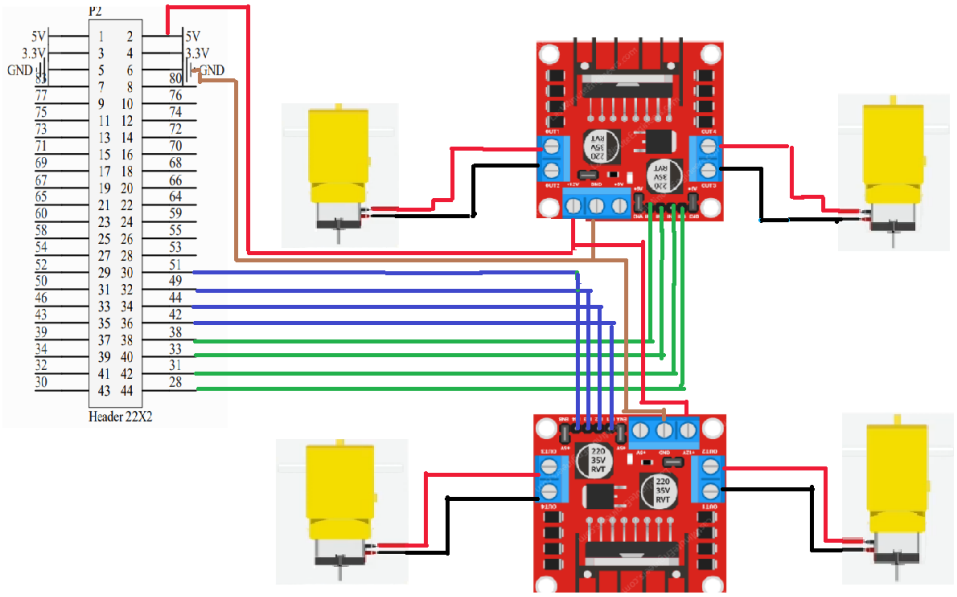


Figure III.10 : Câblage entre FPGA, les moteurs DC et driver L298N.

III.2.6 Module Bluetooth HC 05

HC-05 : 6 sorties. Ce module peut être « maître » (il peut proposer à un autre élément Bluetooth de s'appairer avec lui) ou « esclave » (il ne peut que recevoir des demandes d'appairage)[III.4].

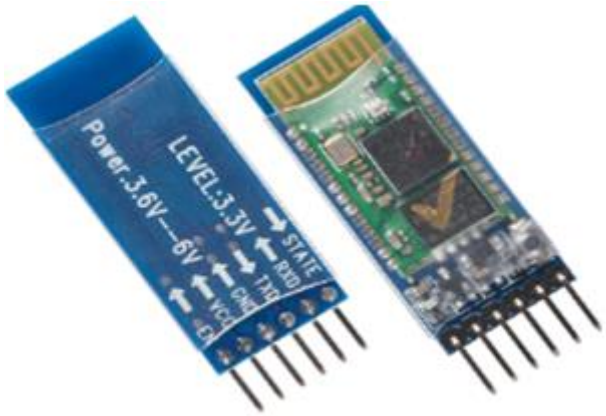


Figure III.11 : Module Bluetooth HC05

III.2.6.1 Câblage de FPGA avec module Bluetooth HC 05

La connexion entre FPGA et le module Bluetooth HC-05 est donnée sur la figure suivante:

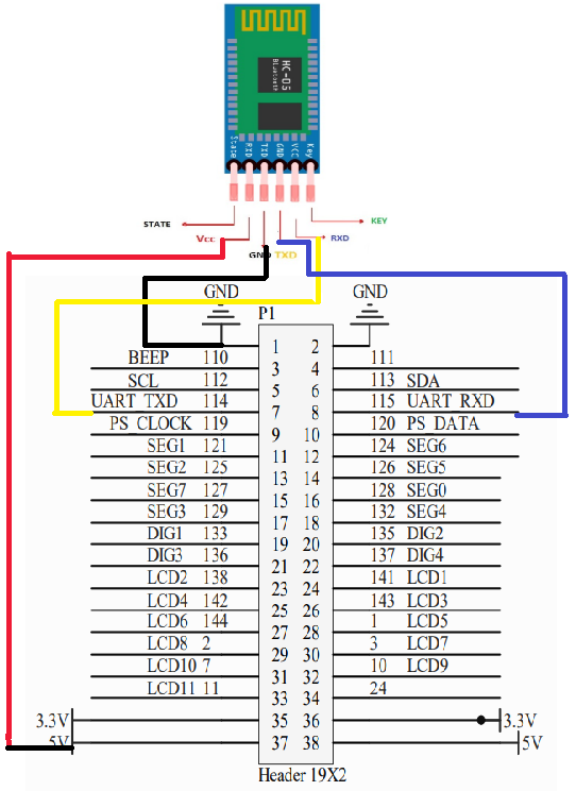


Figure III.12: Connexion entre FPGA et le module Bluetooth HC-05.

III.2.7 Module servomoteur

Un servomoteur est un système qui a pour but de produire un mouvement précis en réponse à une commande externe, C'est un actionneur (système produisant

une action) qui mélange l'électronique, la mécanique et l'automatique. Un servomoteur est composé :

- D'un moteur à courant continu
- D'un axe de rotation
- Un capteur de position de l'angle d'orientation de l'axe (très souvent un potentiomètre)
- Une carte électronique pour le contrôle de la position de l'axe et le pilotage du moteur à courant continu

Un servomoteur est capable d'atteindre des positions prédéterminées dans les instructions qui lui on était donné, puis de les maintenir.

Le servomoteur à l'avantage d'être asservi en position angulaire, cela signifie que l'axe de sortie du servomoteur respectera la consigne d'instruction que vous lui avez envoyée en son entrée.

Figure III.13 : Servomoteur SG90.

III.2.7.1 Caractéristiques de SG90[III.5]

Les caractéristiques du SG90 sont les suivantes :

- Dimensions : 22 x 11.5 x 27 mm.
- Poids : 9 gr.
- Tension d'alimentation : 4.8v à 6v.
- Vitesse : 0.12 s / 60° sous 4.8v.
- Couple : 1.2 Kg / cm sous 4.8v.
- Amplitude : de 0 à 180°.

Le servo est équipé d'une prise de type Graupnerà 3 fils.

La correspondance des fils est la suivante :

Marron : masse

Rouge : + 5v

Orange : commande

La figure suivante met en valeur à la fois la partie câblage (couleurs et fonctions des fils) et les paramètres du signal PWM (durée d'impulsion, période, tension).

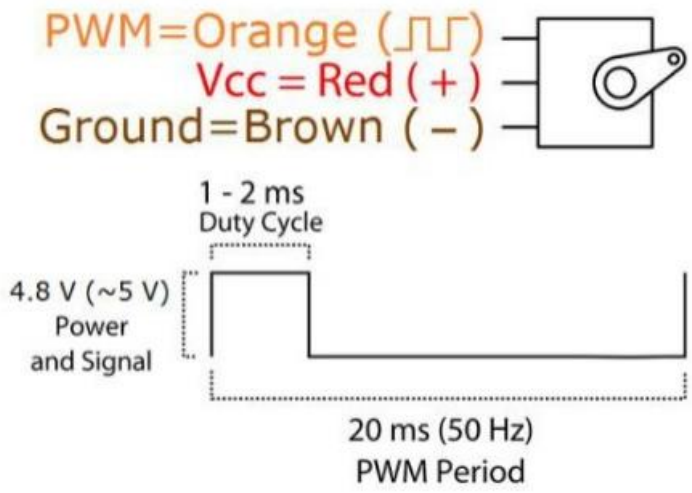


Figure III.14 : Schéma de connexion et caractéristiques du signal PWM pour le Pilotage d'un Servomoteur[III.5].

III.2.8 Module RFID PN532

Le PN532 est un contrôleur de communication sans fil développé par NXP Semiconductors, principalement utilisé pour les applications RFID (Radio Frequency Identification) et NFC (Near Field Communication). Ce composant permet la lecture et l'écriture de tags RFID, ainsi que la communication entre deux dispositifs compatibles NFC.

III.2.8.1 Caractéristiques principales[III.6].

- Protocole supporté : ISO/IEC 14443 Type A & B (tags RFID 13.56 MHz)
- Fréquence de fonctionnement : 13.56 MHz
- Distance de lecture : jusqu'à 5 cm selon l'antenne et le tag utilisé
- Alimentation : 3.3V ou 5V (suivant le module)

III.2.8.2 Modes de fonctionnement

- Lecteur de cartes (Reader/Writer)
- Carte émulée (Card Emulation)
- Peer-to-Peer (mode NFC actif ou passif)

III.2.8.3 Interfaces de communication

- I2C
- SPI
- UART (Série TTL)

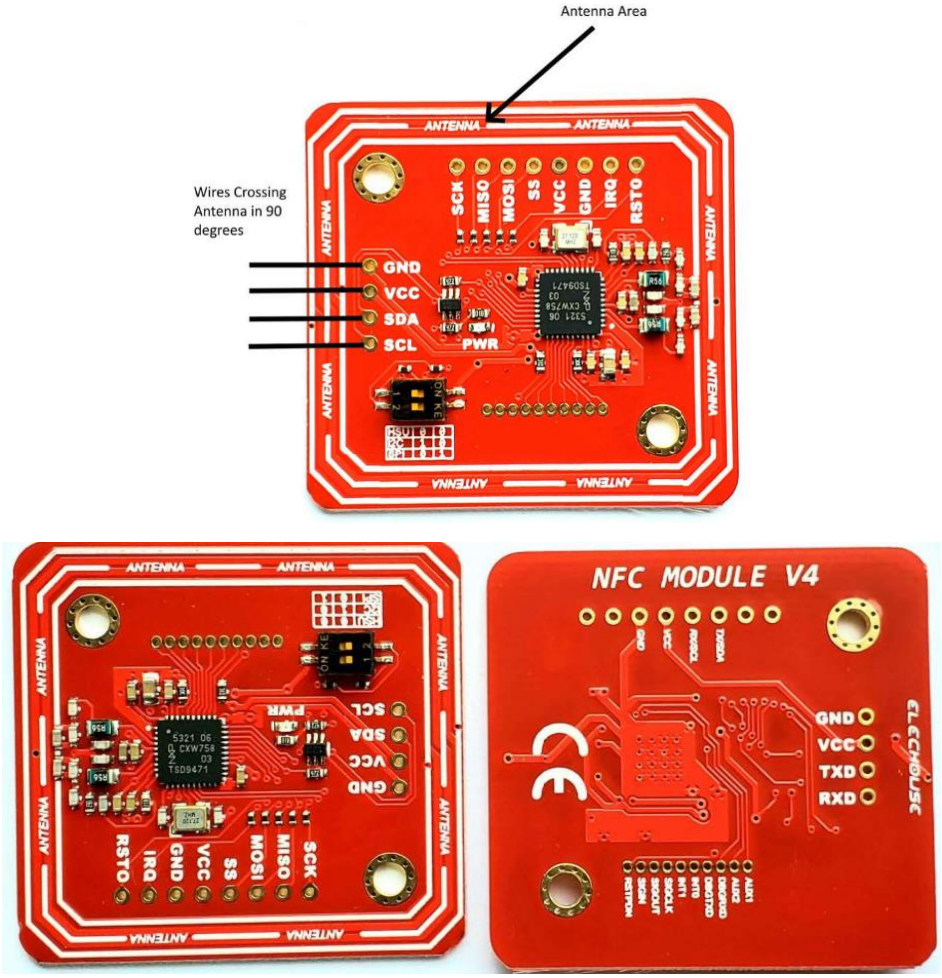


Figure III.15 : RFID PN532.

Les interfaces I2C et UART (HSU) partagent les mêmes broches. La définition des broches I2C est imprimée à l'avant et celle de l'UART (HSU) à l'arrière. Le mode UART (HSU) est configuré par défaut. Vous pouvez toutefois modifier l'interface en réglant l'interrupteur à bascule.

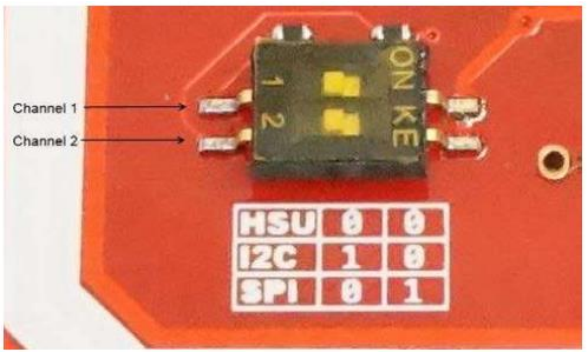


Figure III.16: Cas des commutateurs pour les modes UART (HSU), I2C et SPI.

Le réglage du commutateur pour les différents modes est indiqué comme suit :

Tableau III.3 : Etats des commutateurs pour chaque mode [III.6].

Interface de travail	commutateur 1	commutateur 2
HSU (UART)	Éteint	Éteint
I2C	Allumé	Éteint
SPI	Éteint	Allumé

III.2.8.4 UART (Universal AsynchronousReceiver and Transmitter)[III.7]

UART est un système de communication asynchrone, ce qui signifie que les données ne dépendent pas du signal d'horloge. UART est une communication série utilisant un seul fil pour la transmission et la réception des données. UART possède des paramètres de configuration standard : débit en bauds (2 400, 4 800, 9 600…), bit de départ, bits de données, bit d'arrêt, bit de parité et contrôle de flux. Avant d'établir la communication, nous devons définir ces paramètres, comme pour le récepteur et l'émetteur. La figure III.17 illustre le format des données UART. UART contient des modules récepteur et émetteur. Dans cette configuration, les paramètres de configuration sont : débit en bauds (9 600), bits de données (8 bits), bit de départ (1 bit), bit d'arrêt (1 bit), sans contrôle de flux ni bit de parité.

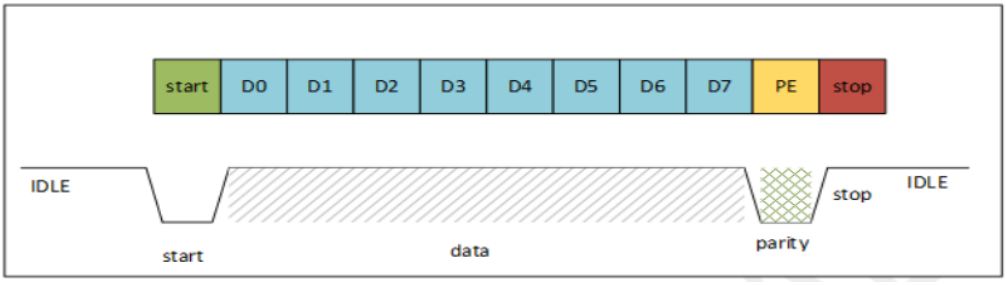


Figure III.17: Format d'une trame de communication série UART[III.7].

III.2.8.5 Câblage de FPGA, le servomoteur et le RFID PN532

La connexion entre FPGA, le servomoteur et le RFID PN532 est présentée sur la figure III.18:

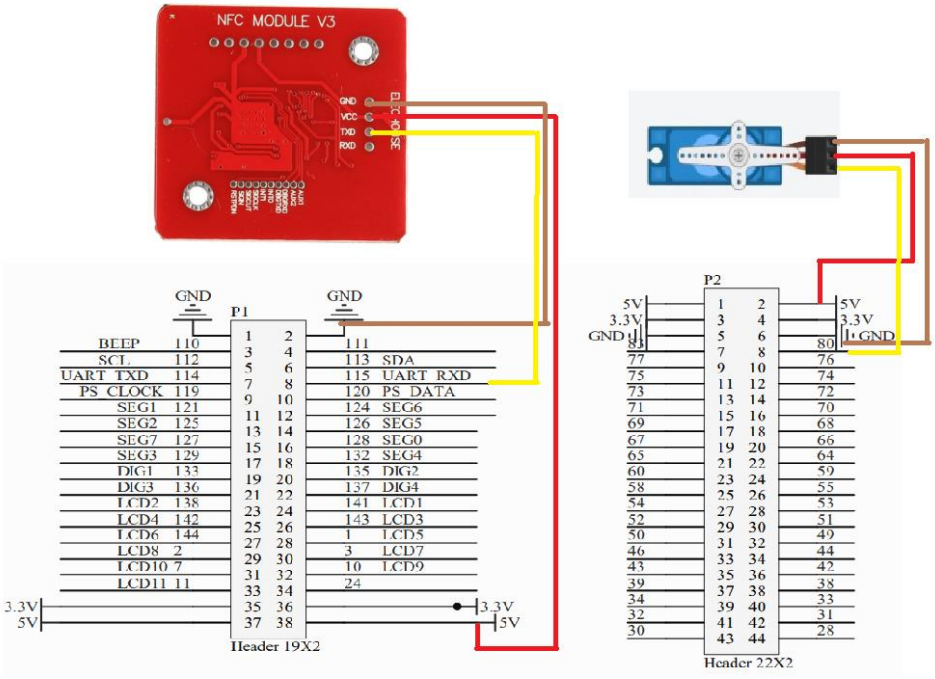


Figure III.18: Connexion entre FPGA, le servomoteur et le RFID PN532.

III.2.9 Roues omnidirectionnelles

Les roues omnidirectionnelles ou Mecanum sont idéales pour les mouvements latéraux rapides. Les roues Mecanum permettent à un robot de se déplacer dans n'importe quelle direction (omnidirectionnelle) grâce à des rouleaux inclinés à 45° par rapport à l'axe de rotation de la roue.

La roue omnidirectionnelle est constituée de rouleaux assemblés autour de chaque roue principale [III.8].

Figure III.19: Roue omnidirectionnelle.

III.3 Partie software

III.3.1 Plateforme de développement Quartus II [III.9]

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.

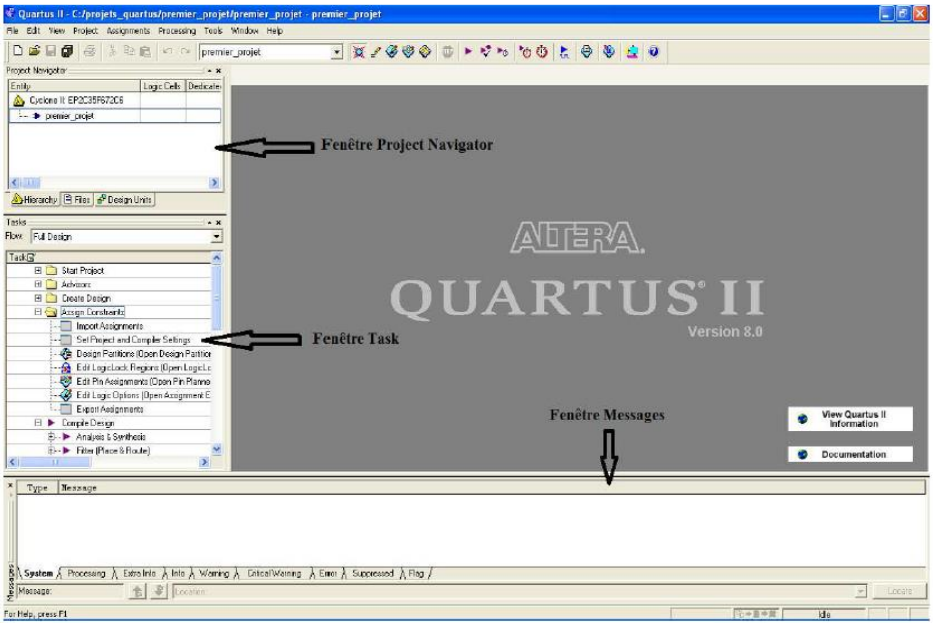


Figure III.20: Environnement de développement Quartus II.

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[III.10].

- **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.3.2 Déroulement de la conception

L'implémentation d'un circuit FPGA sous Quartus suit les étapes de la figure III.21 :

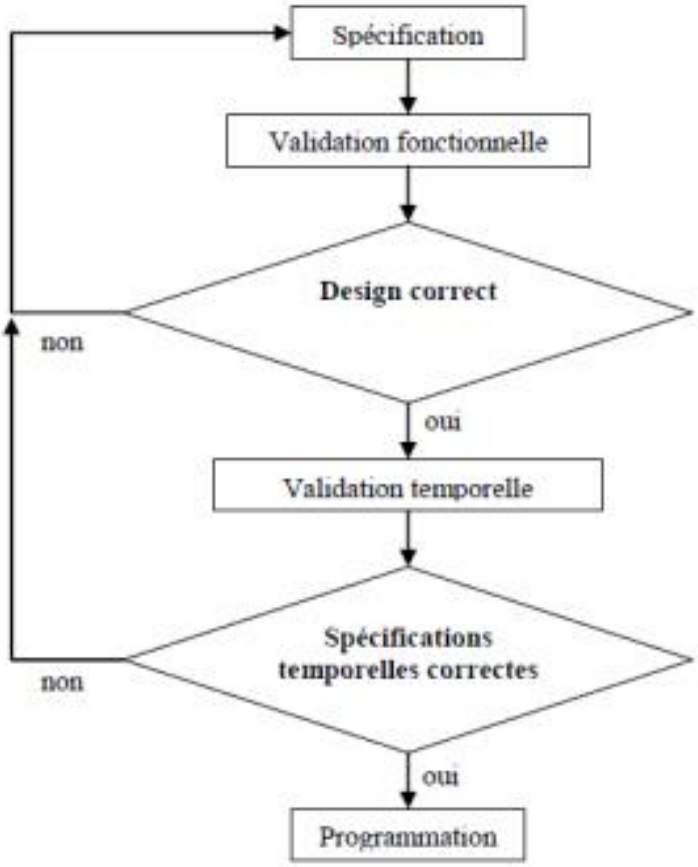


Figure III.21: Déroulement de la conception.

- 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.4 Utilisation d'un FPGA/VHDL pour afficheur LCD 16x2 : affichage d'un message « WELCOME »

III.4.1Création d'un projet

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 :

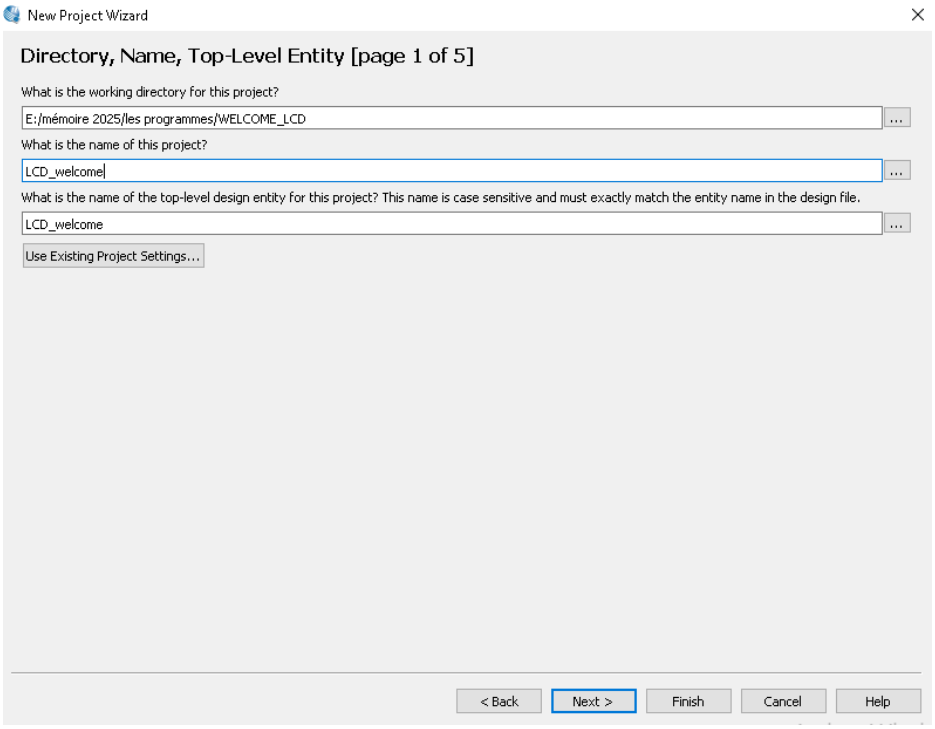


Figure III.22 : Création d'un projet sous quartus II.

Dans la seconde, il est possible d'ajouter des fichiers déjà existants au projet. Ne rien 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.23) à savoir le circuit EP4CE10E22C8 qui est intégré dans la carte de développement OMDAZZ.

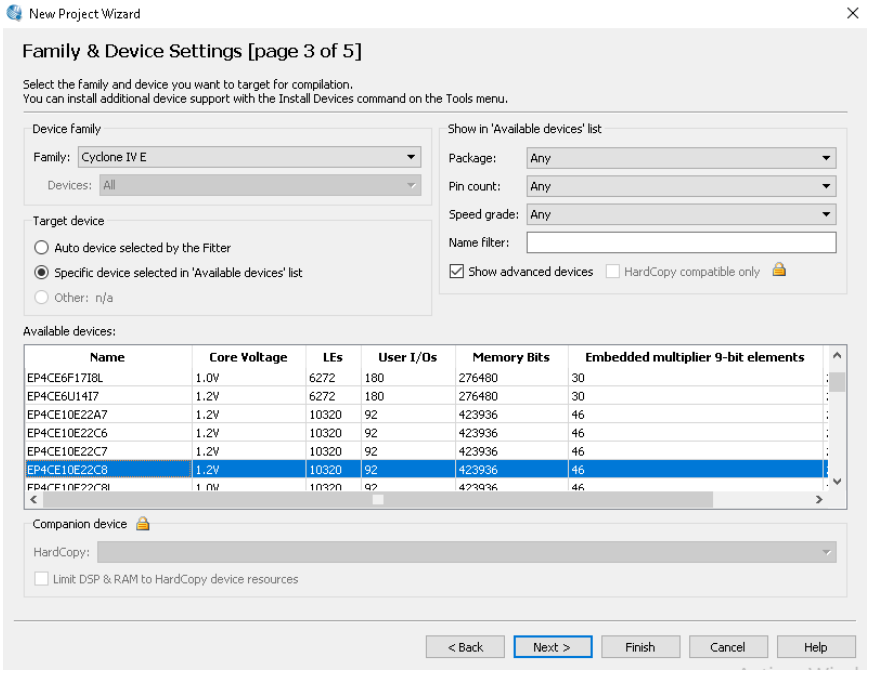


Figure III.23: Fenêtre de choix du circuit.

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

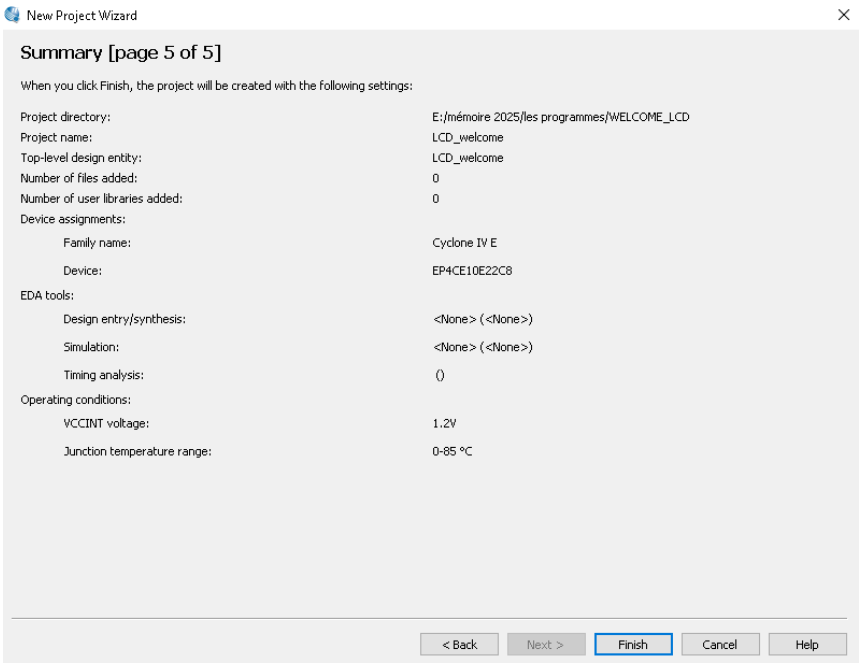


Figure III.24: Fenêtre de création du projet finie.

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 :

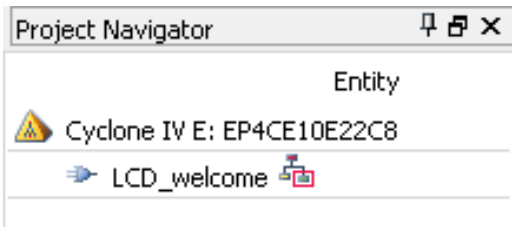


Figure III.25: Fenêtre de type de projet.

III.4.2 Ajout des sources VHDL

Ce projet VHDL a pour objectif de générer un message personnalisé **"WELCOME"** sur un écran LCD contrôlé par un FPGA, en utilisant des caractères spéciaux définis manuellement en matrice 5x8 bits. Le système repose sur une architecture modulaire, composée de plusieurs entités interconnectées, permettant la personnalisation, la gestion et l'affichage du message.

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

III.4.2.1 Module CARACTERES_ESPECIALES_REVC

Ce module définit manuellement les motifs binaires des lettres W, E, L, C, O, M, E sous forme de vecteurs STD_LOGIC_VECTOR (39 DOWNTO 0). Chaque vecteur représente un caractère en 8 lignes de 5 bits, formant une matrice 5x8 utilisée pour la mémoire de l'écran LCD.

Figure III.26: RTL du programme VHDL« CARACTERES_ESPECIALES_REVC ».

III.4.2.2 Module PROCESADOR_LCD_REVC

Ce composant (inclus via instanciation) est responsable de gérer le protocole de communication avec l'écran LCD (gestion de RS, RW, ENA, timing, incrémentation des adresses, etc.). Il permet d'envoyer les instructions ou les caractères à afficher de façon séquentielle.

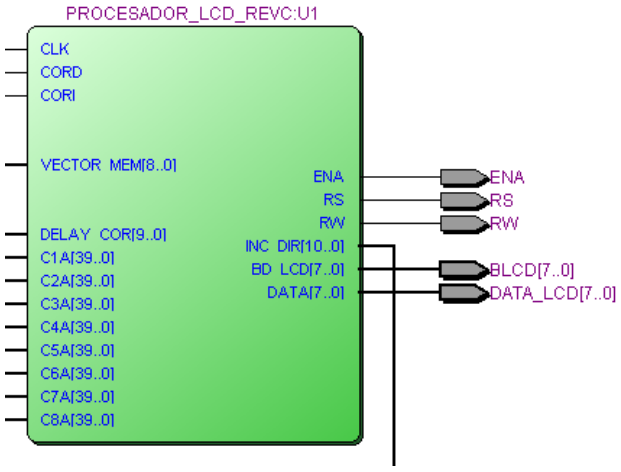


Figure III.27: RTL du programme VHDL « PROCESADOR_LCD_REVC ».

III.4.2.3 Package COMANDOS_LCD_REVC

Ce fichier regroupe les commandes et les constantes nécessaires au pilotage de l'écran LCD. Il contient notamment des fonctions utilitaires comme et les constantes associées aux caractères personnalisés.

III.4.2.4 Code principale (LCD_welcome)

Ce module coordonne l'affichage du mot "WELCOME" sur un écran LCD à l'aide des caractères spéciaux définis dans CARACTERES_ESPECIALES_REVC. Il instancie deux composants : un générateur de caractères spéciaux et un processeur LCD (PROCESADOR_LCD_REVC) qui gère la séquence d'initialisation, d'écriture et de positionnement du curseur. Le mot "WELCOME" est écrit en utilisant les caractères spéciaux personnalisés via la mémoire INST, puis transmis à l'écran. Le module utilise des signaux internes pour synchroniser l'affichage à chaque front d'horloge.

III.4.3 Sauvegarde d'un projet

Après avoir tapé les codes, la sauvegarde se fait comme suite : ***file→ Save As*** …. Puis entrer nom de chaque programme avec l'extension**. vhd**).

III.4.4 Compilation

Après la création du projet avec l'entité «**LCD_welcome**» et création du fichier «**LCD_welcome.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.28.

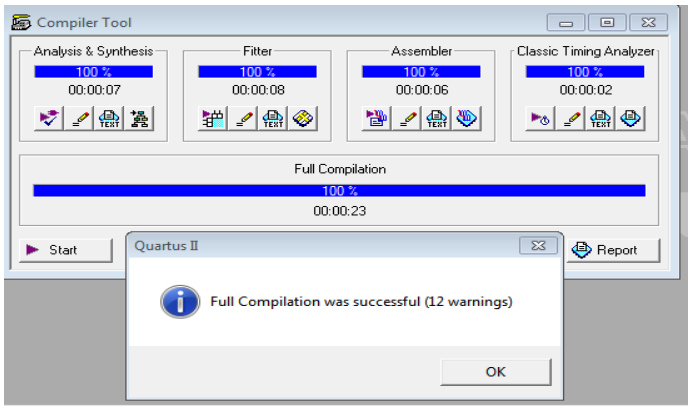


Figure III.28 : Fenêtre de compilation projet «**LCD_welcome**».

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.4.5 Schéma fonctionnel (RTL : **R**egister **Tr**ansfer**L**evel)

Le schéma fonctionnel définit les entrées et les sorties du module 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 → netlistviewer → RTL viewer*** comme la montre la figure ci-après.

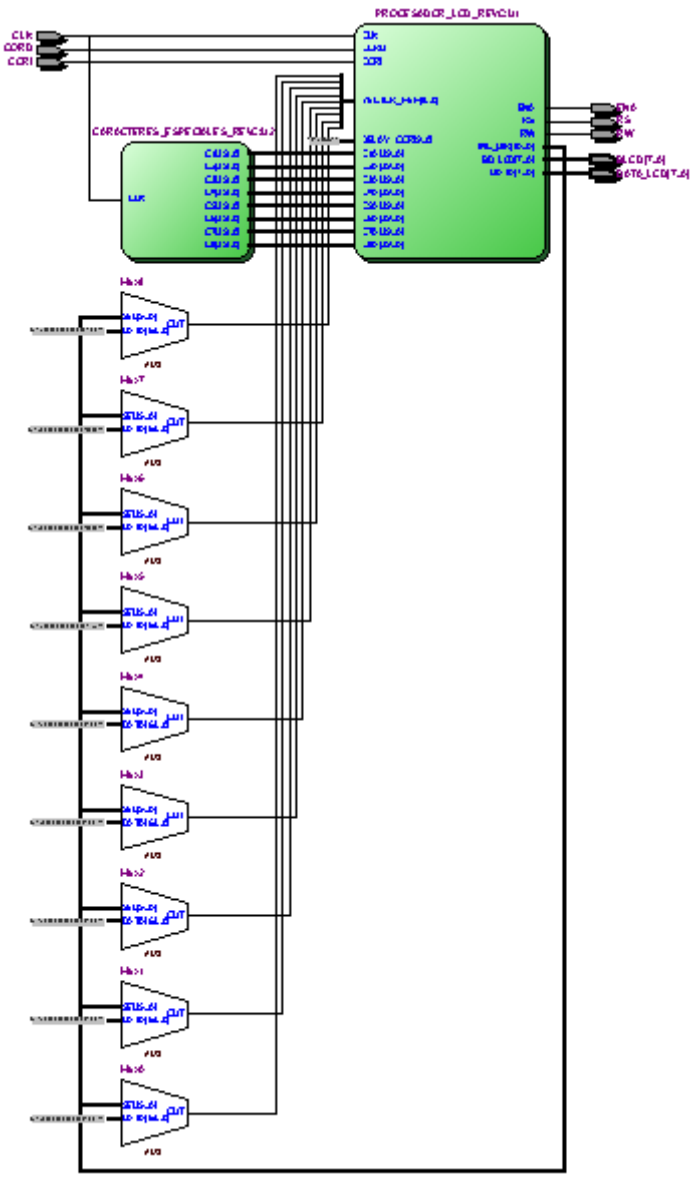


Figure III.29 : Schéma fonctionnel RTL de LCD_welcome.

III.4.6 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.4.6.1 Affectation des pins

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

Dans la fenêtre 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.1] 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.4.

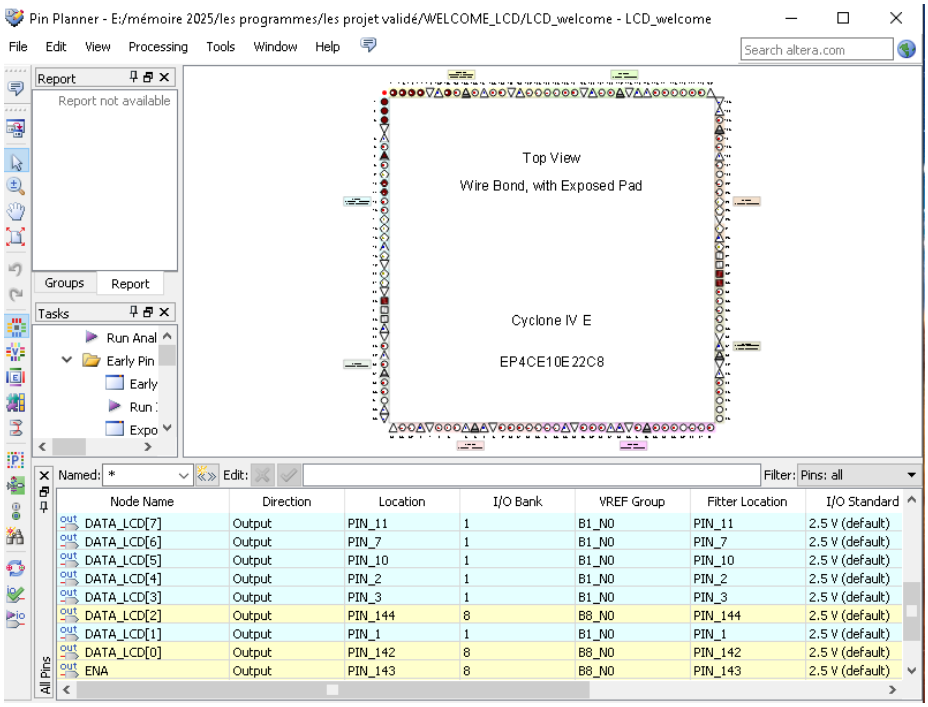


Figure III.30 : Fenêtre de type d'assignement des pins de l'affichage sur LCD.

Les numéros des pins entrées/ sorties utilisés sont résumés sur le tableau suivant :

Tableau III.4 : Numéros des pins entrées/sorties utilisés de l'affichage sur LCD [III.1].

Entrés	Pin	Sorties	Pin	Sorties	Pin
Clk	PIN_23	DATA_LCD7	PIN_11	DATA_LCD1	PIN_1
CORD	PIN_88	DATA_LCD6	PIN_7	DATA_LCD0	PIN_142
CORI	PIN_89	DATA_LCD5	PIN_10	ENA	PIN_143
		DATA_LCD4	PIN_2	RW	PIN_138
		DATA_LCD3	PIN_3	RS	PIN_141
		DATA_LCD2	PIN_144		

III.4.6.2 Programmation du circuit

Il est alors possible de programmer le FPGA de la maquette. Le câble de programmation associé doit être relié au port USBde l'ordinateurpar son entrée BLASTERet 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**.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 notreFPGA 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.***

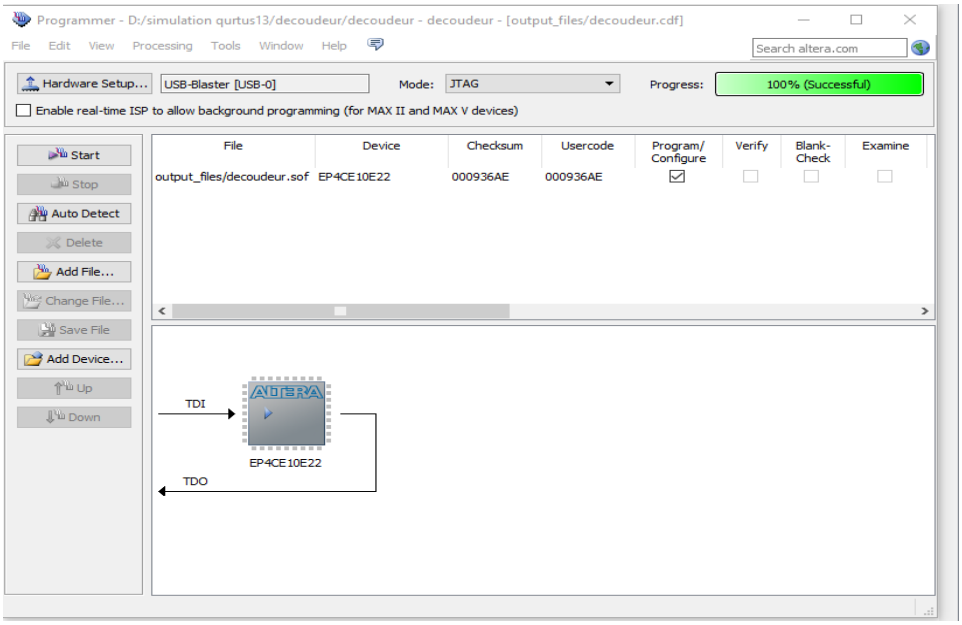


Figure III.31: Boite de dialogue de programmation.

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 programmé voir la figure III.32:

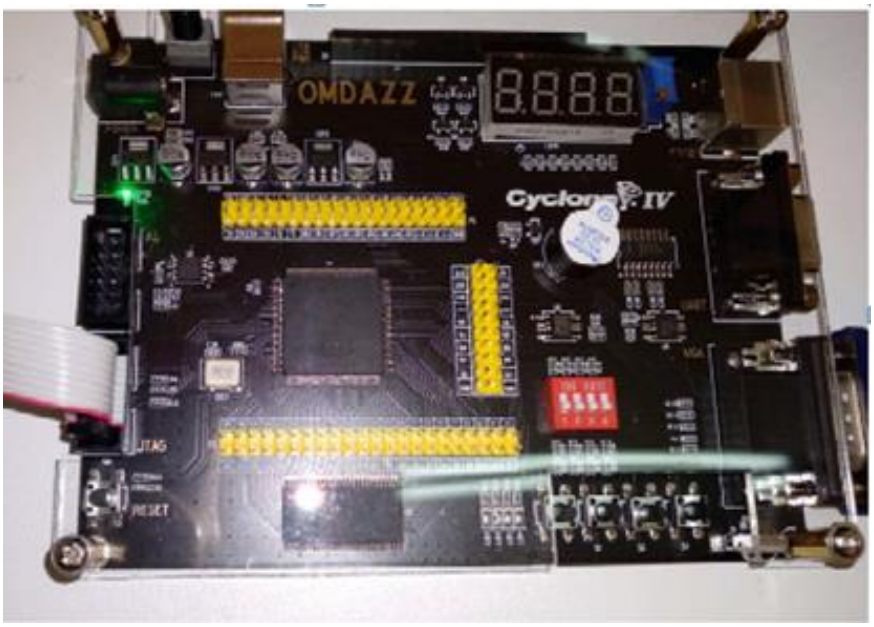


Figure III.32: Plateforme Cyclone IV programmée.

La figure III.33 montre le résultat obtenu après la programmation de notre FPGA avec le programme VHDL qui s'affiche le message **WELCOME** sur l'afficheur LCD.

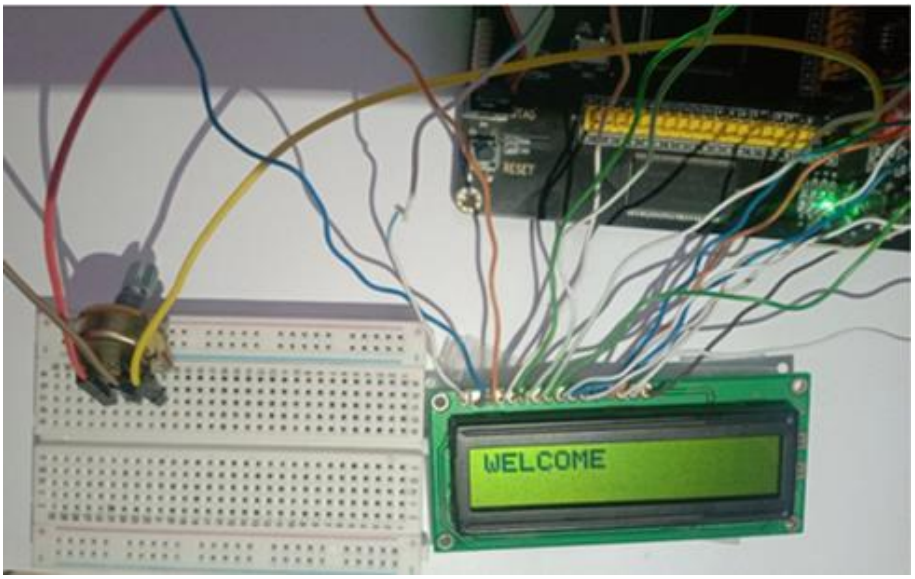


Figure III.33: Affichage du message **« WELCOME »** sur l'afficheur LCD 16x2.

III.5 Utilisation d'un FPGA/VHDL pour la détection des obstacles et la mesure de la distance en utilisant un capteur ultrasonique et afficheur LCD 16x2

Ce projet du capteur ultrasonique et l'affichage LCD dans notre travail sont principalement responsables de la gestion de la mesure de la distance et de son affichage en temps réel sur un écran LCD. Le capteur ultrasonique mesure la distance, qui est ensuite traitée et affichée sur l'écran LCD. Si un obstacle est trop proche, un buzzer peut être activé pour signaler un danger. Ce système est crucial pour la gestion des interactions avec l'environnement, notamment pour la navigation autonome du véhicule omnidirectionnel, où la détection d'obstacles et l'affichage de la distance sont essentiels. L'intégration de ces modules permet un contrôle plus précis et intuitif du système.

III.5.1 Codes VHDL

III.5.1.1 Capteur Ultrasonique (INTESC_LIB_ULTRASONICO_RevC)

Ce programme gère le capteur ultrasonique pour mesurer la distance entre le capteur et un objet. Il utilise le principe du temps de vol de l'écho, où un signal est émis par le capteur (via le TRIGGER) et renvoyé sous forme d'écho (via ECO). Une fois l'écho reçu, la distance est calculée et exprimée en centimètres. Le signal de distance est transmis au reste du système via le port **DISTANCIA_CM**. Le programme déclenche également une **DATO_LISTO** pour signaler que la mesure est prête. Ce module permet ainsi de récupérer des données précises pour la navigation ou la détection d'obstacles du véhicule.

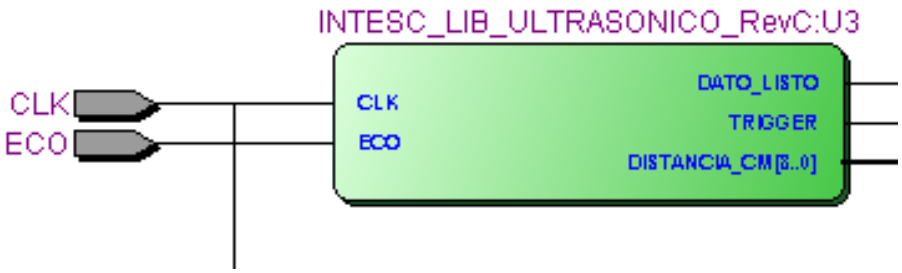


Figure III.34: RTL du programme VHDL « INTESC_LIB_ULTRASONICO_RevC ».

III.5.1.2 Module PROCESADOR_LCD_REVC

Ce module assure la gestion de l'écran LCD. Il reçoit les instructions de type commande ou données (par **VECTOR_MEM**) et les transmet à la LCD via les signaux de contrôle (**RS, RW, ENA**) et les lignes de données (**BD_LCD**). Il prend aussi en charge le défilement ou la mise à jour des lignes de texte à l'aide d'un compteur d'instruction (**INC_DIR**) et d'un délai configurable (**DELAY_COR**). Il permet également l'affichage de caractères spéciaux définis dans les modules externes.

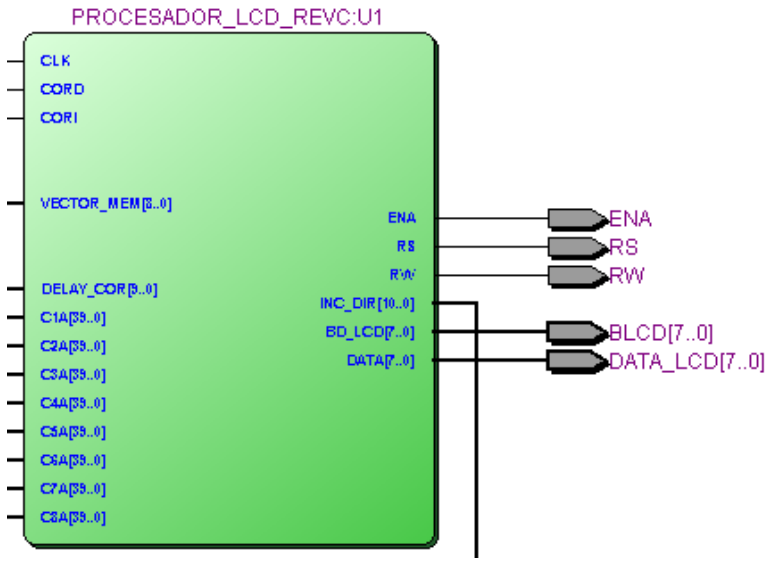


Figure III.35: RTL du programme VHDL « PROCESADOR_LCD_REVC ».

III.5.1.3 Module CARACTERES_ESPECIALES_REVC

Ce module génère des caractères spéciaux personnalisés pour l'affichage sur la LCD (par exemple, des symboles graphiques, icônes ou lettres accentuées). Il produit huit caractères (C1 à C8) sous forme de vecteurs binaires de 5 lignes par 8 colonnes chacun. Ces caractères sont ensuite utilisés par le processeur LCD pour enrichir l'affichage, offrant une meilleure interface utilisateur ou des symboles visuels spécifiques à l'application.

Figure III.36: RTL du programme VHDL « CARACTERES_ESPECIALES_REVC ».

III.5.1.4 Module DIVISION_ULTRASONICO_RevA

Ce module réalise la division du temps mesuré par le capteur ultrasonique pour convertir le nombre de cycles d'horloge (pendant lesquels le signal d'écho est actif) en une valeur de distance exprimée en centimètres. Il effectue une division matérielle entre le temps mesuré et une constante correspondant au facteur d'échelle requis pour obtenir une mesure en cm (basée sur la vitesse du son et la fréquence du signal d'horloge). Il s'agit d'un module clé pour la conversion physique des données du capteur HC-SR04 en valeurs numériques utilisables par le système.

III.5.1.5 Module COMANDOS_LCD_REVC

Le module COMANDOS_LCD_REVC est responsable de la gestion des commandes et des données envoyées à un écran LCD via des signaux de contrôle standard tels que RS, RW, et ENA. Il interprète les commandes et les données à afficher, contrôlant ainsi le fonctionnement de l'écran. Ce module facilite l'affichage de texte et de caractères spéciaux sur l'écran LCD, en utilisant des signaux synchronisés pour le transfert des données. Il est conçu pour être intégré dans un système FPGA et utilisé avec d'autres composants tels que des processeurs de texte et des modules de génération de caractères spéciaux

III.5.1.6 Module principal LIB_LCD_INTESC_REVC

Ce module central intègre trois sous-modules (capteur ultrasonique, processeur LCD et caractères spéciaux) pour afficher en temps réel la distance mesurée sur un écran LCD et activer un buzzer si un obstacle est trop proche. Il

convertit la distance binaire en centimètres, la décompose en centaines, dizaines et unités, puis affiche ces valeurs sur la deuxième ligne de l'écran LCD. Un signal sonore (BUZZER) est activé si la distance mesurée est inférieure à 10 cm. Ce module coordonne les composants matériels pour fournir une interface interactive et visuelle de mesure de distance.

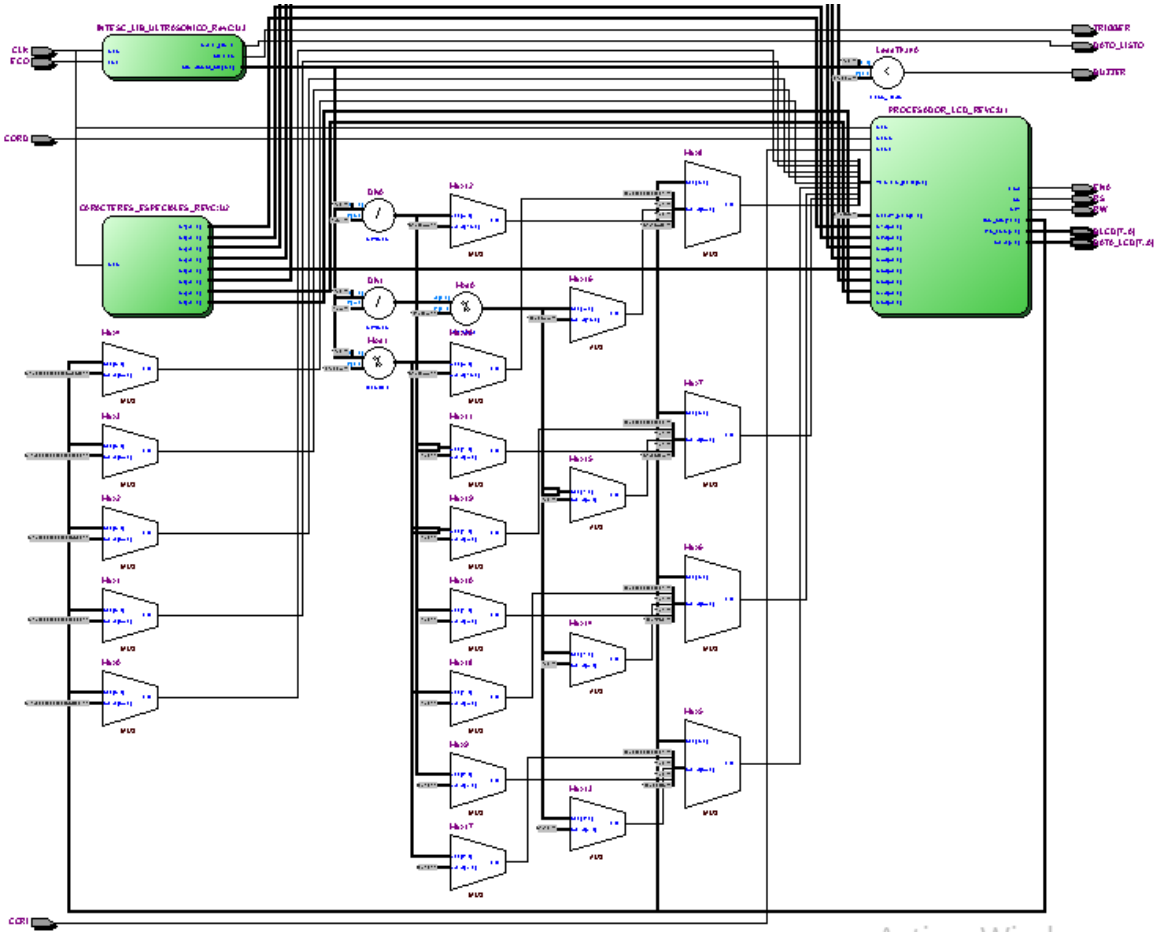


Figure III.37: Schéma fonctionnel RTL de code principal « LIB_LCD_INTESC_REVC».

III.5.2 Affectation des pins

Le tableau III.5 résume les different numéros des pins utilizes dans le brochage de FPGA avec le LCD et le capteur ultrasonique HC-SR04.

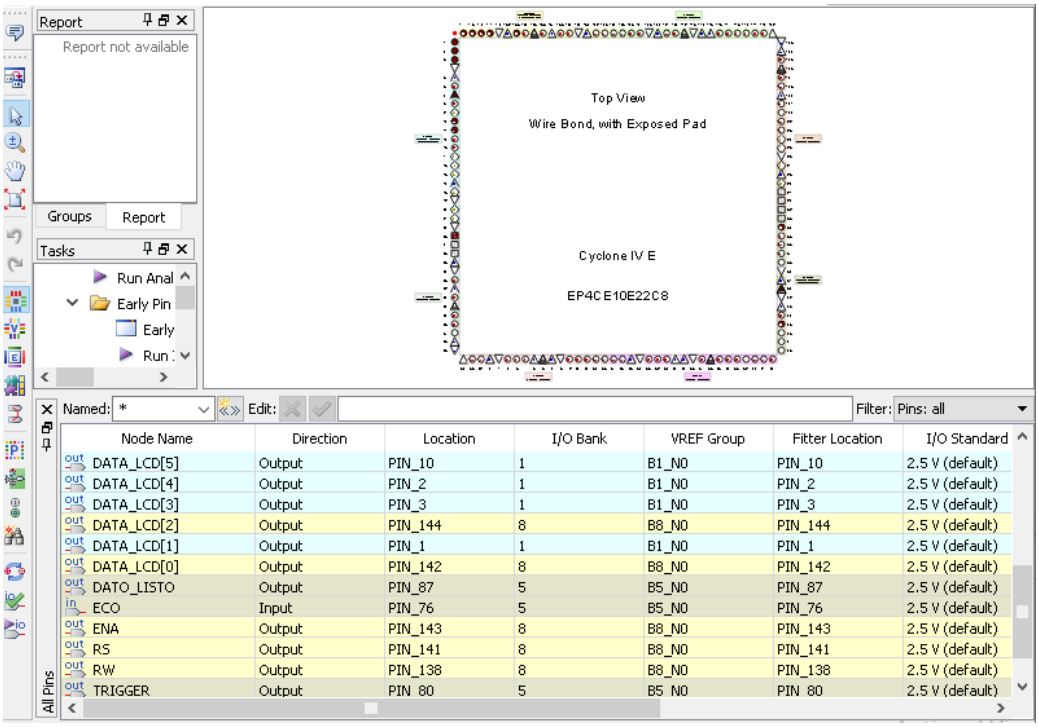


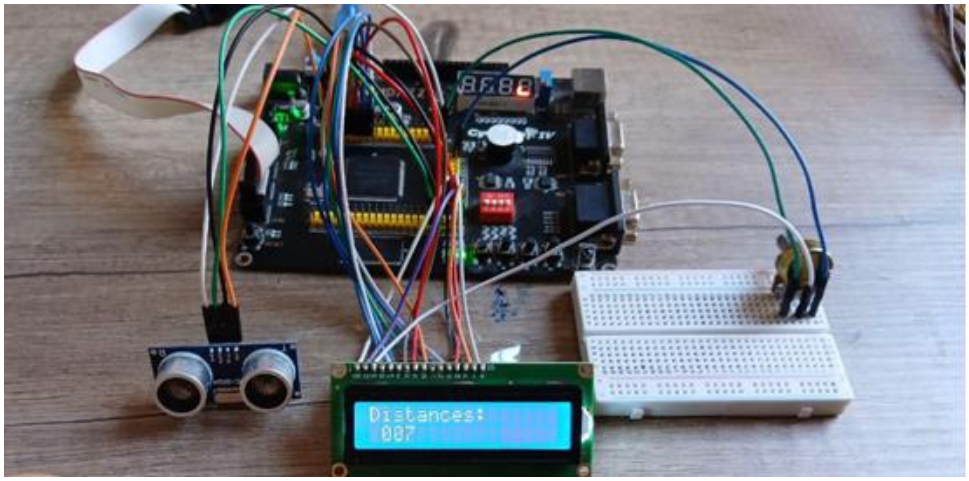
Figure III.38: Fenêtre de type d'assignement des pins de mesure de la distance.

Tableau III.5 : Numéros des pins entrées/sorties utilisés pour la mesure de la distance [III.1]

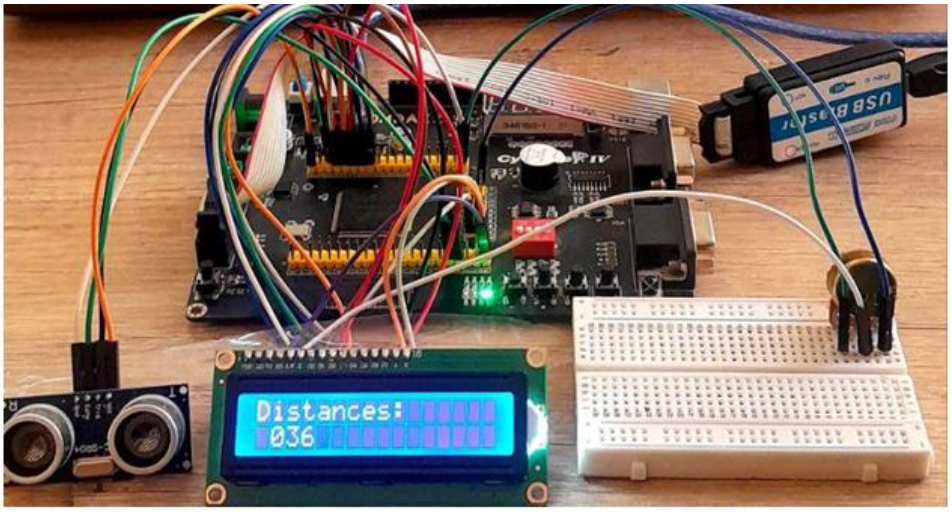
Entrés	Pin	Sorties	Pin	Sorties	Pin
Clk	PIN_23	DATA_LCD7	PIN_11	DATA_LCD1	PIN_1
CORD	PIN_88	DATA_LCD6	PIN_7	DATA_LCD0	PIN_142
CORI	PIN_89	DATA_LCD5	PIN_10	ENA	PIN_143
ECO	PIN_76	DATA_LCD4	PIN_2	RW	PIN_138
		DATA_LCD3	PIN_3	RS	PIN_141
		DATA_LCD2	PIN_144	TRIGGER	PIN_80

III.5.3 Résultats d'affichage des différentes distances

Les deux figures suivantes représentent respectivement l'affichage d'une distance de 7 cm et 36 cm.



(a)



(b)

Figure III.39 : Exemples de la distance mesurée: (a) 07 cm, (b) 36 cm.

III.6 Utilisation d'un FPGA/VHDL pour réaliser un véhicule omnidirectionnel

Dans un contexte où les systèmes embarqués prennent une place de plus en plus importante dans le développement des technologies intelligentes, les véhicules robotisés représentent un domaine d'application particulièrement dynamique. L'intégration de la logique programmable via des FPGA offre une alternative

puissante et flexible aux microcontrôleurs classiques, permettant de concevoir des systèmes parallèles, rapides et hautement personnalisables. Ce projet s'inscrit dans cette démarche innovante, en exploitant les avantages du langage VHDL pour programmer un FPGA, et concevoir un véhicule omnidirectionnel intelligent.

III.6.1 Sources VHDL

III.6.1.1 Module UART_rs232_tx

Le module **VHDL UART_rs232_tx** implémente un transmetteur série UART permettant d'envoyer une donnée de 8 bits (TxData) via la ligne Tx, selon le protocole RS232. Lorsqu'un signal de transmission (TxEn) est détecté, la machine à états passe de l'état d'attente (IDLE) à l'état d'écriture (WRITE). Le module envoie d'abord un bit de Start (niveau bas), puis les bits de données un par un à intervalles réguliers définis par le signal Tick, et termine par un bit de stop (niveau haut). La transmission est cadencée à raison d'un bit toutes les 16 impulsions de Tick, et le signal TxDone indique la fin de l'envoi. L'ensemble du processus repose sur un registre à décalage pour l'envoi séquentiel des bits et une logique de contrôle simple pour respecter le protocole UART.

III.6.1.1.1 Fonctionnement du module VHDL UART_rs232_tx

Ce module joue le rôle de transmetteur série UART pour envoyer une donnée de 8 bits via une ligne série (Tx), en respectant le protocole UART (RS232 au niveau logique, mais avec des niveaux TTL)**.**

a) Description fonctionnelle du module UART_rs232_tx

Le module **UART_rs232_tx** est un **transmetteur série UART** implémenté en VHDL. Il permet d'envoyer une donnée de 8 bits (TxData) sur une ligne série (Tx) selon le **protocole UART** conforme à la norme **RS232** (au niveau logique).

Le module est basé sur une **machine à états** ayant au minimum deux états :

- **IDLE :** état d'attente, aucune transmission en cours.
- **WRITE :** état actif où la donnée est transmise bit par bit.

- Lorsque le signal de validation **TxEn** est activé (**'1'**), le module commence une nouvelle transmission.
- La transmission se fait en trois étapes selon le protocole UART :

1. **Bit de start** : niveau bas ('0'), indiquant le début de la transmission.
2. **8 bits de données** : envoyés du bit de poids faible (LSB) au bit de poids fort (MSB).
3. **Bit de stop** : niveau haut ('1'), marquant la fin de la trame.

- Le signal **Tick**définit l'horloge de transmission UART. Chaque bit est émis toutes les 16 impulsions de**Tick**, ce qui permet de générer un **baudrate**précisà partir d'une horloge système plus rapide (par ex. 50 MHz).
- Cela signifie qu'un compteur de 4 bits (0 à 15) est utilisé pour cadencerchaque bit transmis.
- Le signal **TxDone** est activé à **'1'** à la fin de la transmission complète (bit de stop inclus), indiquant que le module est prêt pour une nouvelle transmission.

Composants internes typiques :

- Un **registre à décalage** est utilisé pour envoyer les bits de la donnée séquentiellement.
- Une **logique de contrôle** (FSM : Finite State Machine) pilote la séquence Start → Data → Stop.
- Un **compteur de Tick** gère la temporisation entre les bits.

En résume les éléments et leurs fonctions comme suit :

TxData Entrée : donnée à transmettre (8 bits)

TxEn Entrée : déclenchement de la transmission

Tx Sortie : ligne série (UART TX)

Tick Entrée : horloge base (bit chaque 16 Ticks)

TxDone Sortie : transmission terminée ('1' = fin)

III.6.1.2 Module UART_rs232_rx

Le module VHDL **UART_rs232_rx** implémente un récepteur série UART qui permet de recevoir des données sur la ligne Rx selon le protocole RS232. Il fonctionne à l'aide d'une machine à états qui alterne entre les états IDLE (attente) et READ (lecture). Lorsqu'un bit de start (niveau bas) est détecté sur Rx et que le signal RxEn est actif, le module passe à l'état de lecture. La réception est cadencée par le signal Tick, chaque bit étant lu au centre de sa durée (après 8 ticks pour le start, puis tous les 16 ticks pour les bits de données). Les bits sont stockés séquentiellement dans le registre Read_data, puis transférés vers la sortie RxData une fois le bit de stop (niveau haut) détecté. Le signal RxDone est activé à la fin de la réception pour indiquer que la donnée reçue est valide. Le module prend en charge différents formats de données (6, 7 ou 8 bits) selon la valeur de NBits.

III.6.1.2.1 Fonctionnement du module VHDL UART_rs232_tx

Le module VHDL **UART_rs232_rx**, est le **récepteur UART,**il est donc complémentairedu module **UART_rs232_tx**décrit précédemment.

En résume les éléments et leurs fonctions comme suit :

IDLE (repos) "état IDLE (attente)"

Start bit (0) "lorsqu'un bit de start (niveau bas) est détecté"

Bits D0 à D7 "chaque bit étant lu [...] tous les 16 ticks"

Bit de stop (1) "le bit de stop (niveau haut) détecté"

RxDone "signal RxDone est activé à la fin de la réception"

Read_data → RxData "bits sont stockés [...] puis transférés vers RxData"

Support NBits "formats de données (6, 7 ou 8 bits)"

III.6.1.3UART_BaudRate_generator

Le module VHDL **UART_BaudRate_generator** est un générateur de tick d'horloge utilisé pour synchroniser la réception des données UART à une vitesse spécifique (baudrate). Il prend en entrée une horloge système Clk, un signal de réinitialisation Rst_n actif bas, et une valeur BaudRate représentant le nombre de cycles à attendre pour générer un tick. Le compteur interne baudRateReg s'incrémente à chaque cycle d'horloge. Lorsqu'il atteint la valeur de BaudRate, il génère une impulsion courte (Tick) et se réinitialise. Ce tick est ensuite utilisé par le module de réception UART pour échantillonner les bits au bon moment, assurant une lecture fiable des données sérient.

III.6.1.4 TOP

Le module TOP regroupe les composants nécessaires à une communication série UART complète entre un FPGA et un périphérique externe (comme un module Bluetooth HC-05). Il intègre trois sous-modules : un récepteur (UART_rs232_rx) pour lire les données série entrantes via la ligne Rx, un émetteur (UART_rs232_tx) pour envoyer des données via la ligne Tx, et un générateur de tick (UART_BaudRate_generator) qui produit des impulsions de synchronisation à un taux défini (BaudRate fixé ici pour une communication à 9600 bauds). Le récepteur détecte les bits entrants synchronisés par les ticks, les assemble en octets et les place dans RxData quand la réception est terminée (RxDone). L'émetteur envoie les données TxData bit par bit, également synchronisées par les ticks. Le tout fonctionne avec une configuration fixe de 8 bits de données (NBits = 8), assurant une communication série fiable entre le FPGA et un périphérique compatible UART.

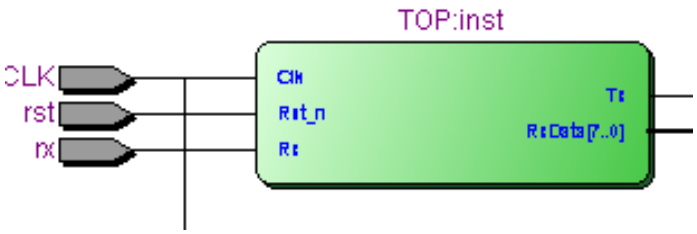


Figure III.40: RTL du programme VHDL « TOP ».

III.6.1.5 MOTOR_DC

Le module VHDL MOTOR_DC permet de contrôler la vitesse et le sens de rotation d'un moteur à courant continu (DC) à l'aide d'un signal PWM. Il prend en entrée une horloge CLK et un vecteur de 3 bits SW pour configurer le comportement du moteur. Les bits SW(0) et SW(1) déterminent la vitesse du moteur en sélectionnant une valeur de rapport cyclique PWM, tandis que SW(2) contrôle le sens de rotation. Le module génère les sorties PWM1 et PWM2, qui commandent un pont en H pour faire tourner le moteur dans un sens ou l'autre. Un compteur COUNT cadencé par CLK permet de créer le signal PWM, et selon la comparaison entre COUNT et la valeur SPEED, les sorties PWM sont activées ou non. Ce système permet un contrôle simple mais efficace de la vitesse et du sens de rotation d'un moteur DC.

Figure III.41: RTL du programme VHDL « MOTOR_DC ».

III.6.1.6 Controlador

Ce programme VHDL décrit un module appelé **controlador,** destiné à piloter un véhicule omnidirectionnel à l'aide d'un FPGA. Il gère plusieurs fonctions principales : le contrôle des quatre moteurs via les signaux sw1 à sw4, la mesure de distance à l'aide d'un capteur à ultrasons (HC-SR04), l'activation d'un buzzer d'alerte si besoin. Les moteurs sont commandés en fonction des données reçues sur l'entrée led, qui proviennent typiquement d'une commande Bluetooth. Le capteur à ultrasons fonctionne avec des signaux Trigger et Echo, et la distance mesurée est convertie en valeurs binaires. Ce programme illustre une intégration matérielle complète de commande et de capteur dans un seul système embarqué.

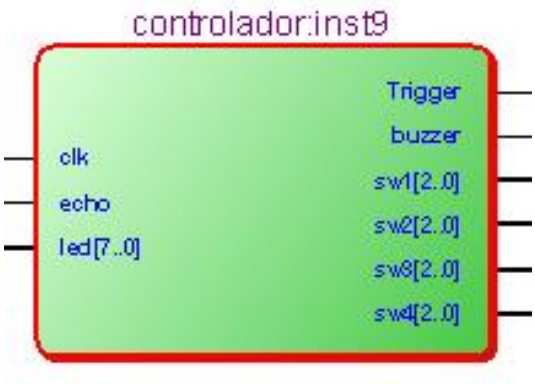


Figure III.42: RTL du programme VHDL « Controlador ».

III.6.2 Schéma bloc de projet

Le schéma bloc est un outil essentiel pour concevoir et organiser et comprendre un projet VHDL. Il sert de plan de construction du système numérique.

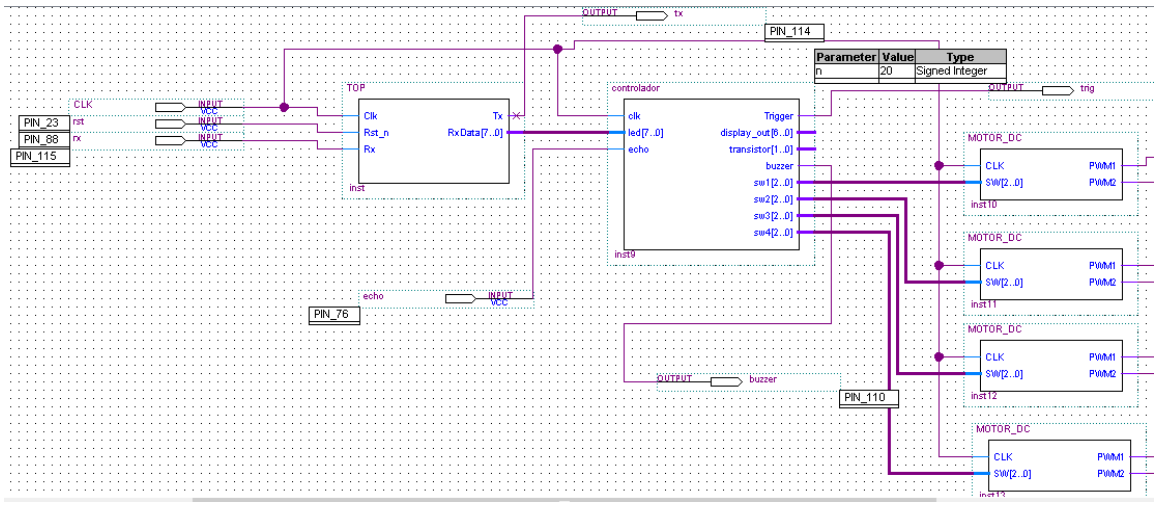


Figure III.43 : Schéma bloc de projetpour réaliser un véhicule omnidirectionnel.

III.6.3 Affectation des pins

Les numéros des broches d'entrées, sortie et d'horloge sont données dans le tableau III.6 et la figure III.44.

Tableau III.6: Numéros des pins entrées/sorties utilisés pour notre véhicule omnidirectionnel[III.1].

Entrés	Pin	Sorties	Pin	Sorties	Pin
Clk	PIN_23	Buzzer	PIN_110	Pwm5	PIN_42
rx	PIN_115	Tx	PIN_114	Pwm6	PIN_44
rst	PIN_88	Pwm1	PIN_51	Pwm7	PIN_33
ECO	PIN_76	Pwm2	PIN_49	Pwm8	PIN_38
		Pwm3	PIN_28	TRIGGER	PIN_80
		Pwm4	PIN_31		

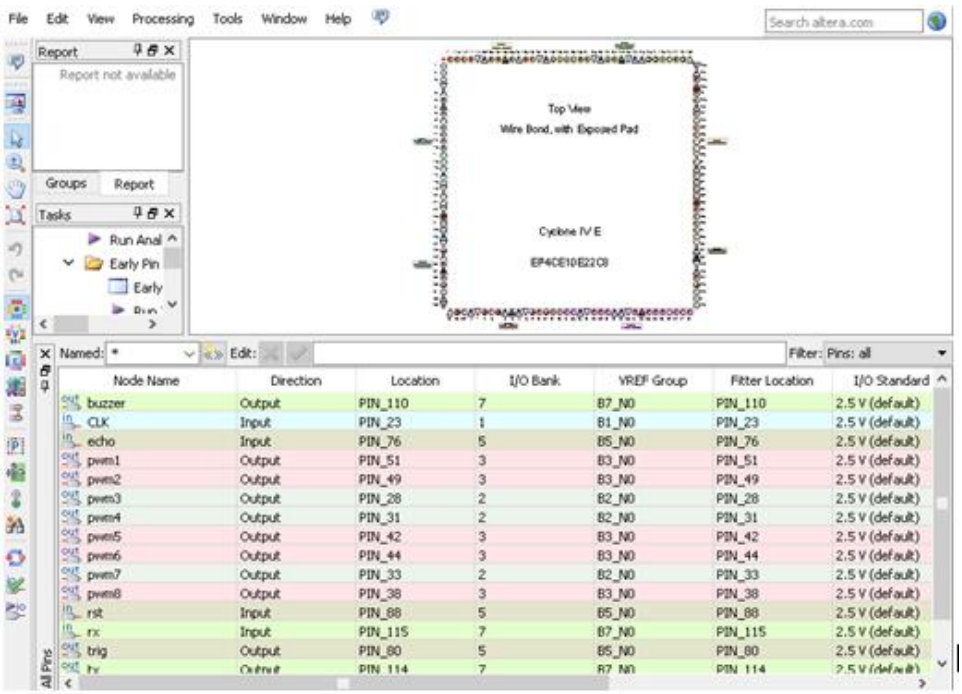


Figure III.44: Fenêtre de type d'assignement des pins du véhicule omnidirectionnel.

.

III.6.4 Résultat de conception et réalisation de notre véhicule omnidirectionnel

La figure suivante représente la photo réelle de résultat de conception et réalisation de notre véhicule omnidirectionnelavec le système embarqué FPGA.

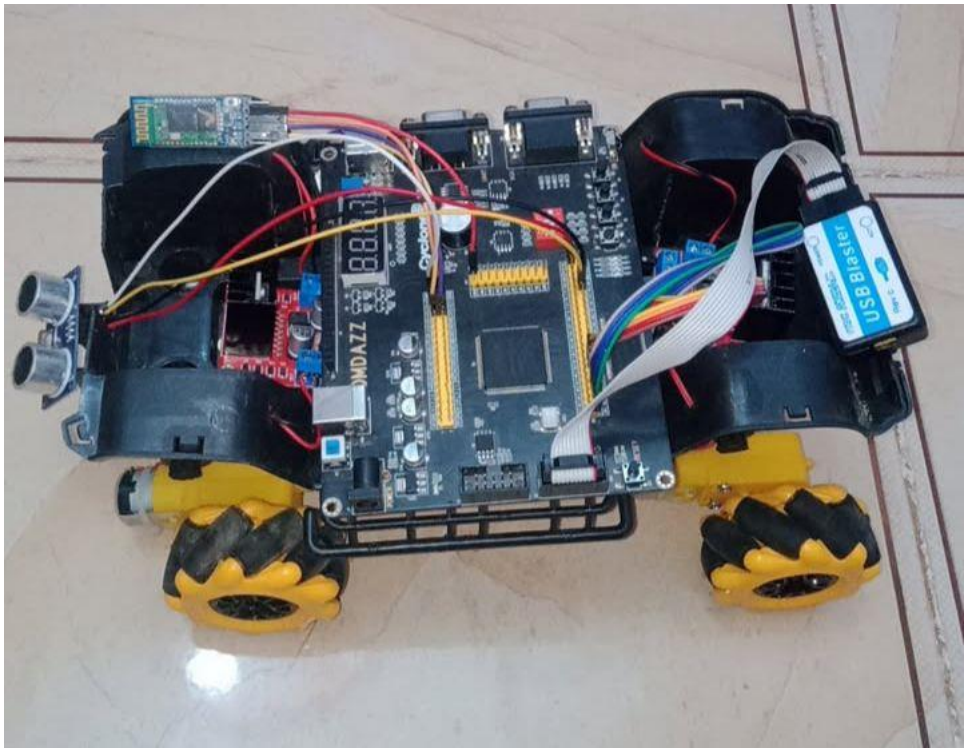


Figure III.45: Photo du véhicule omnidirectionnelle avec le système embarqué FPGA.

III.6.5 Application MIT App Inventor pour le contrôle du Véhicule via Bluetooth

Dans le cadre de la réalisation d'un système embarqué de contrôle de véhicule omnidirectionnel, il est essentiel de disposer d'une interface utilisateur simple et efficace permettant d'interagir avec le système en temps réel. Pour répondre à cette exigence, une application mobile a été développée à l'aide de **MIT App Inventor,** une plateforme de développement visuel particulièrement adaptée aux projets pédagogiques et prototypages rapides.

L'application conçue permet à l'utilisateur de piloter le véhicule à distance via une connexion Bluetooth, en envoyant des commandes de direction (avant, arrière, gauche, droite, arrêt) depuis un smartphone. Grâce à une interface graphique

intuitive, cette solution assure une prise en main rapide, tout en facilitant la communication entre l'utilisateur et le système embarqué basé sur FPGA.

Cette section présente la conception, le fonctionnement et l'intégration de l'application mobile avec le module Bluetooth HC-05, ainsi que le format des commandes échangées entre le téléphone et le véhicule. Elle met en lumière l'intérêt de l'utilisation de MIT App Inventor pour créer des interfaces personnalisées adaptées aux projets embarqués.

III.6.5.1 Etapes de la création de l'application

- Cliquez sur **"Projects" > "Start new project"**
- Donnez un nom de projet : **"Car_Controler"**
- Concevoir l'interface utilisateur (UI)
  - ➢ Dans l'onglet Designer :

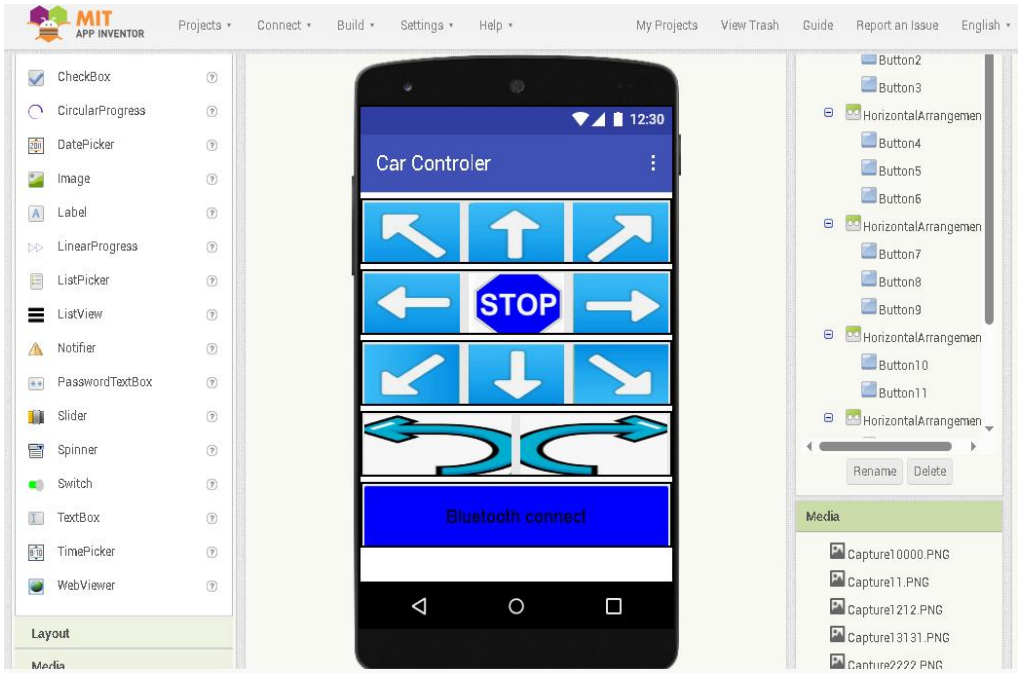


Figure III.46 : Interface de l'application mobile de contrôle du véhicule via Bluetooth sous MIT App Inventor

❖ **Hello world**

Pour le désigne on ajoute un message de « Hello world »

❖ **Image de notre véhicule**

On ajoute une image de notre véhicule sur l'interface de notre application

❖ **Les boutons d'orientation**

L'interface principale de l'application mobile est composée de plusieurs boutons directionnels représentant les différentes commandes de mouvement du véhicule : Avancer, Reculer, Tourner à gauche, Tourner à droite et Stop… etc. Chaque bouton est lié à un événement "Click" dans MIT App Inventor, déclenchant l'envoi d'une commande spécifique via le module Bluetooth HC-05.

❖ **L'activation de Bluetooth**

Avant de pouvoir envoyer des commandes au véhicule, l'application doit établir une connexion Bluetooth entre le Smartphone et le module Bluetooth embarqué (HC-05). Cette étape est essentielle, car elle permet de garantir une communication fiable et bidirectionnelle entre les deux appareils.

Appuyé pour connecter avec Bluetooth

Figure III.47 : Bouton pour connecter avec Bluetooth.

Figure III.48: Message de l'activation de Bluetooth.

➢ **Dans l'onglet Blocks :**

L'onglet *Blocks* de MIT App Inventor permet de définir le comportement de l'application mobile en associant des actions aux différents éléments de l'interface. Dans ce projet, deux fonctionnalités principales ont été programmées : l'activation du Bluetooth et le contrôle directionnel du véhicule. Lors du démarrage de

l'application, un bloc vérifie si le Bluetooth du Smartphone est activé, puis permet à l'utilisateur de se connecter au module HC-05 via un bouton et une liste des périphériques appariés. Une fois la connexion établie, des boutons directionnels permettent de piloter le véhicule à distance. Chaque bouton (Avancer, Reculer, Gauche, Droite, Stop …etc.) envoie une commande spécifique sous forme de texte au module Bluetooth, qui est ensuite interprétée par le système embarqué pour contrôler les moteurs. Ce fonctionnement assure une interaction simple, intuitive et en temps réel entre l'utilisateur et le véhicule.

Chaque bouton a un bloc spécial :

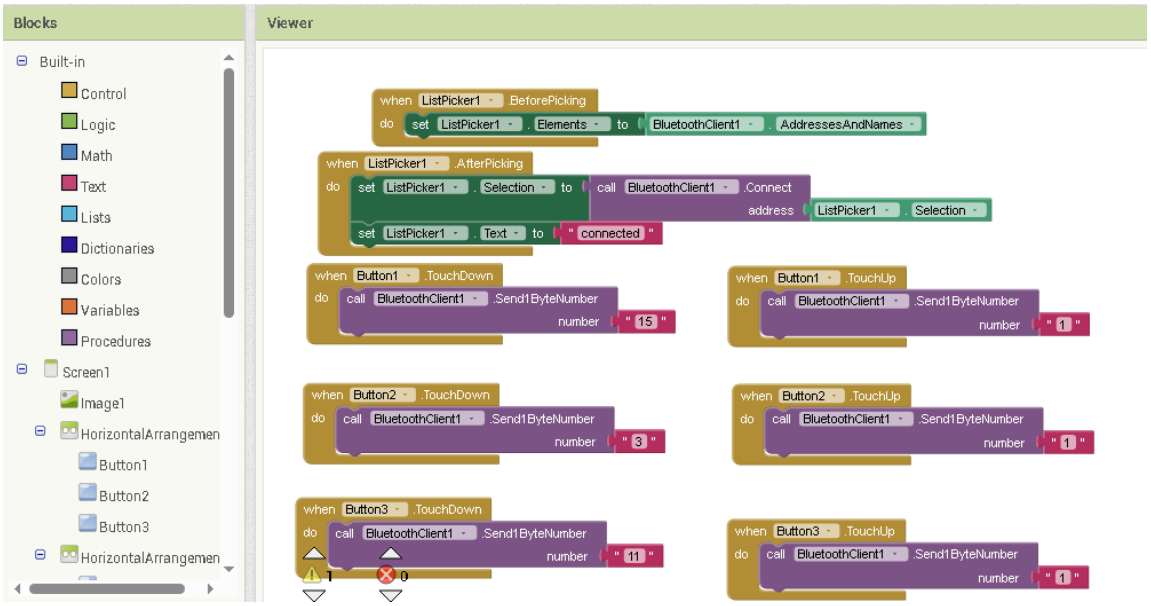


Figure III.49 : Blocs de l'application de contrôle du véhicule.

- **Tester l'application**
  - Téléchargez l'application MIT AI2 Compagnon sur votre Smartphone
  - Scannez le QR code avec l'application pour tester en temps réel

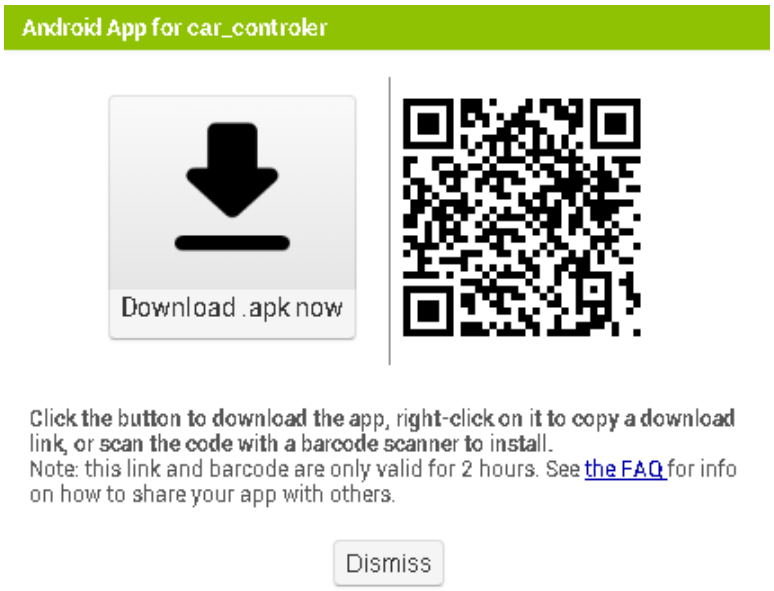


Figure III.50 : Code QR pour le téléchargement.

- L'application de notre projet :

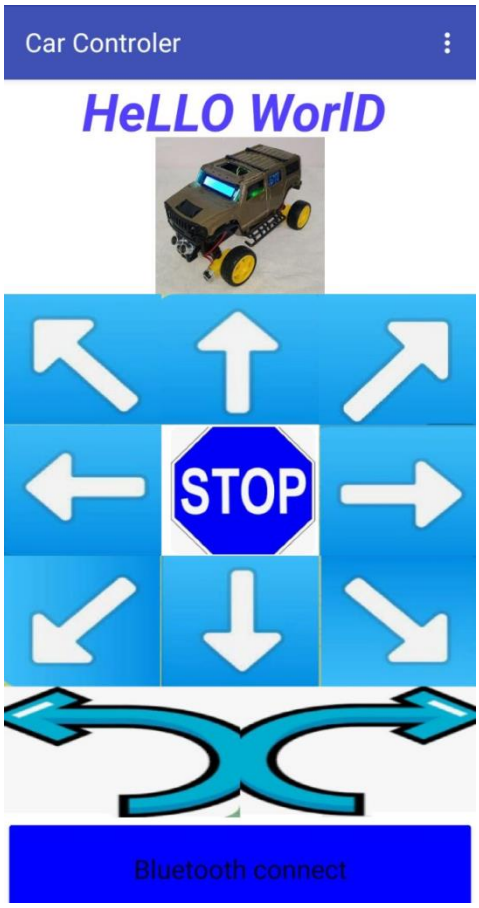


Figure III.51: L'application de la manipulation de notre véhicule.

III.7 Utilisation d'un FPGA/VHDL pour la réalisation du véhicule autonome

L'utilisation d'un FPGA programmé en VHDL pour la réalisation d'un véhicule autonome permet d'exploiter la flexibilité et la parallélisation des circuits logiques pour la base des applications d'intelligence artificielle embarquée. Dans ce projet, le véhicule évolue de manière autonome en analysant son environnement à l'aide d'un capteur à ultrasons. Lorsqu'un obstacle est détecté à une distance critique, le buzzer s'active et le système, en réponse, interrompt la progression du véhicule pour lui faire effectuer automatiquement une rotation vers la gauche. Cette capacité de réaction rapide face à l'environnement, intégrée entièrement en logique matérielle, illustre le potentiel des FPGA dans le développement de systèmes intelligents embarqués, réactifs et efficaces.

III.7.1 Sources VHDL

III.7.1.1 MOTOR_DC

Le module VHDL **MOTOR_DC** permet de contrôler la vitesse et le sens de rotation d'un moteur à courant continu (DC) à l'aide d'un signal PWM. Il prend en entrée une horloge CLK et un vecteur de 3 bits SW pour configurer le comportement du moteur. Les bits SW(0) et SW(1) déterminent la vitesse du moteur en sélectionnant une valeur de rapport cyclique PWM, tandis que SW(2) contrôle le sens de rotation. Le module génère les sorties PWM1 et PWM2, qui commandent un pont en H pour faire tourner le moteur dans un sens ou l'autre. Un compteur COUNT cadencé par CLK permet de créer le signal PWM, et selon la comparaison entre COUNT et la valeur SPEED, les sorties PWM sont activées ou non. Ce système permet un contrôle simple mais efficace de la vitesse et du sens de rotation d'un moteur DC.

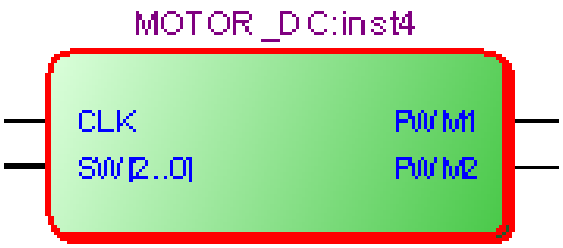


Figure III.52 : RTL du programme VHDL « MOTOR_DC».

III.7.1.2 Auto

Le module auto implémente une logique de contrôle autonome basée sur la mesure de distance par un capteur ultrasonique. Il génère un signal Trigger cyclique pour déclencher la mesure de distance et écoute le signal echo en retour pour estimer la durée du trajet aller-retour de l'onde ultrasonique. Cette durée est convertie en une distance numérique via une table de correspondance (distance_bits). En fonction de cette distance, le système contrôle divers composants : un buzzer (probablement pour alerter en cas de proximité dangereuse), et quatre moteurs (sw1 à sw4) représentés par des vecteurs à 3 bits chacun. La conversion du temps d'écho en distance est codée de façon discrète générée à partir de cette valeur. Bien que le code soit inachevé (partie decenas coupée), il illustre clairement une logique d'évitement d'obstacle par la mesure de distance et la gestion de la réponse du robot en conséquence.

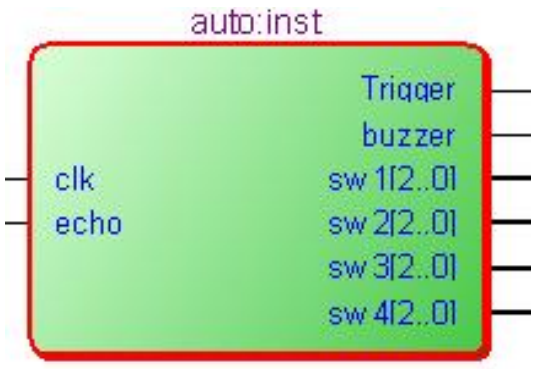


Figure III.53 : RTL du programme VHDL « auto ».

III.7.1.3 Schéma fonctionnel RTL du code principal du véhicule autonome

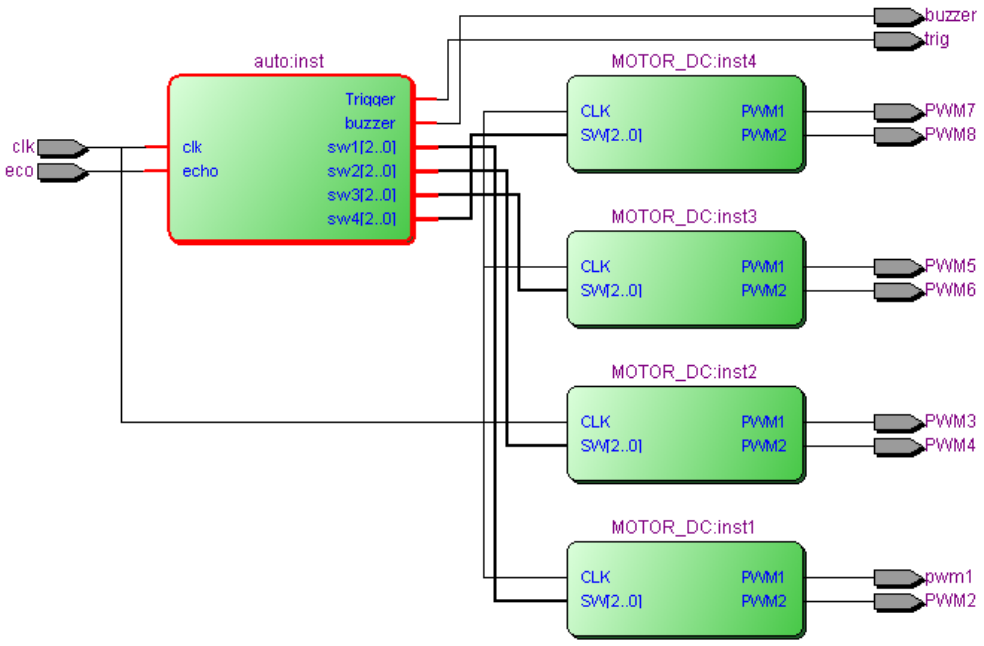


Figure III.54 : Schéma fonctionnel RTLdu véhicule autonome.

III.7.2 Affectation des pins

Les numéros des pins utilisés dans cette application sont résumés dans le tableau suivant :

Tableau III.7 : Numéros des pins entrées/sorties utilisés pour le véhicule autonome [III.1].

Entrées	Pins	Sorties	Pins	Sorties	Pins
CLK	Pin_23	Trig	Pin_80	PWM4	Pin_31
Eco	Pin_76	Buzzer	Pin_110	PWM5	Pin_42
		PWM1	Pin_51	PWM6	Pin_44
		PWM2	Pin_49	PWM7	Pin_33
		PWM3	Pin_28	PWM8	Pin_38

III.7.3 Résultat de conception et réalisation de notre véhicule autonome

La figure III.55 représente la photo réelle de résultat de conception et réalisation de notre véhicule autonome avec des roues omnidirectionnelles et le système embarqué FPGA.

Figure III.55: Photo du véhicule autonome avec le système embarqué FPGA.

III.8 Utilisation d'un FPGA/VHDL pour l'ouverture de la porte de notre véhicule en utilisant la technologie RFID

L'intégration des technologies d'identification par radiofréquence (RFID) dans les systèmes embarqués modernes permet de renforcer la sécurité et l'automatisation des accès. Dans le cadre de notre projet, nous avons mis en œuvre une solution permettant l'ouverture automatique d'une porte de véhicule à l'aide d'un lecteur RFID PN532 et d'un servomoteur SG90, le tout contrôlé par un FPGA programmé en VHDL.

III.8.1 Sources VHDL

III.8.1.1 access_control_uid

Ce module VHDL gère un système d'authentification par identifiant unique (UID), typiquement utilisé dans des applications RFID. Il lit séquentiellement un UID composé de 4 octets lorsque le signal uid_ready est actif. Une fois les 4 octets collectés, l'UID est comparé à une valeur autorisée (AUTH_UID). Si l'UID correspond, le module actif un signal PWM (pwm_enable), allume une LED d'accès (led_access). En cas de non-correspondance, un buzzer est activé pendant 1 seconde pour indiquer un refus (denied). Le module fonctionne avec une horloge de 50 MHz et intègre une logique d'état (idle, collecting, granted, denied) pour gérer les différentes étapes du processus d'authentification.

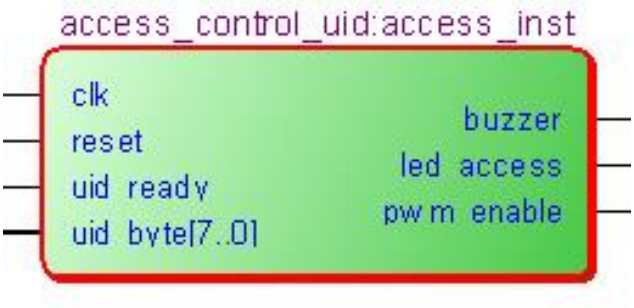


Figure III.56 : RTL du programme VHDL « access_control_uid ».

III.8.1.2 pn532_parser

Ce module VHDL est un parseur de trame de communication UART destinée au module RFID PN532, utilisé pour extraire l'UID (identifiant unique) d'un badge RFID. Il fonctionne comme une machine à états qui détecte les séquences caractéristiques d'une trame PN532 (preambuleB3B104, longueur, commande 04, etc.) et isole les 4 octets de l'UID. L'UID est stocké dans un registre temporaire puis émis octet par octet via la sortie uid_byte avec un indicateur de validité uid_ready. Le signal uart_valid déclenche la lecture d'un nouvel octet entrant (uart_data). Le module remet à zéro l'état après chaque trame et gère le décalage de l'UID jusqu'à sa transmission complète.

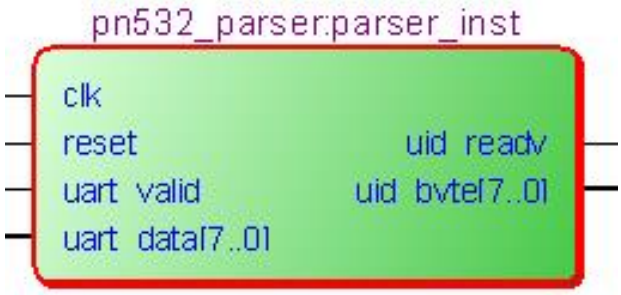


Figure III.57 : RTL du programme VHDL « pn532_parser ».

III.8.1.3 pwm_generator

Ce module VHDL génère un signal PWM (Pulse Width Modulation) adapté au contrôle de servomoteurs, en se basant sur une fréquence d'horloge de 50 MHz. La période PWM est fixe à 20 ms (50 Hz), ce qui est standard pour les servos. L'entrée duty_cycle (variant de 50 à 100) détermine la largeur d'impulsion, correspondant à un rapport cyclique de 5 % à 10 %, soit une durée d'impulsion de 1 ms à 2 ms. Cela permet de contrôler précisément la position d'un servomoteur. Le signal pwm_out passe à '1' pendant la durée d'impulsion, puis à '0' pour le reste du cycle. Le module remet le compteur à zéro à chaque fin de période

.

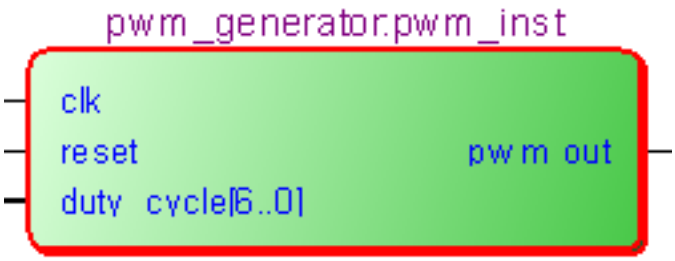


Figure III.58 : RTL du programme VHDL « pwm_generator **».**

III.8.1.4uart_receiver

Ce module VHDL implémente un récepteur UART asynchrone compatible avec un débit standard de 9600 bauds, en utilisant une horloge système de 50 MHz. Il décode les trames série reçues sur la ligne rx en capturant les bits de données à intervalles précis (déterminés par TICKS_PER_BIT). Le module suit une machine à états (idle, start_bit, data_bits, stop_bit) pour détecter le bit de départ, lire les 8 bits de données, puis vérifier le bit de stop. Une fois un octet complet reçu et validé, il est stocké dans data_out et signalé comme prêt via data_ready. Ce module permet ainsi d'intégrer une communication série fiable avec un périphérique comme un lecteur RFID PN532.

Figure III.59 : RTL du programme VHDL « uart_receiver **».**

III.8.1.5top_level (code principale)

Ce module VHDL représente l'architecture hiérarchique principale du système complet de contrôle d'accès RFID. Il connecte et orchestre les sous-modules suivants :

- uart_receiver : décode les trames UART reçues via la broche rx.
- pn532_parser : analyse les trames du lecteur RFID PN532 et extrait l'UID du badge.
- access_control_uid : compare l'UID reçu avec un UID autorisé ; active un pwm, une LED d'accès (led_access) ou un buzzer selon le résultat de l'authentification.
- pwm_generator : génère un signal PWM avec un duty cycle fixe (70 % ici) pour piloter un servomoteur (via pwm_out) lorsque l'accès est autorisé.

Le signal pwm_out est activé uniquement lorsque pwm_enable est à '1', assurant que le servomoteur ne s'actionne que si le badge est reconnu. Ce module constitue le point d'intégration de l'ensemble du système RFID sécurisé.

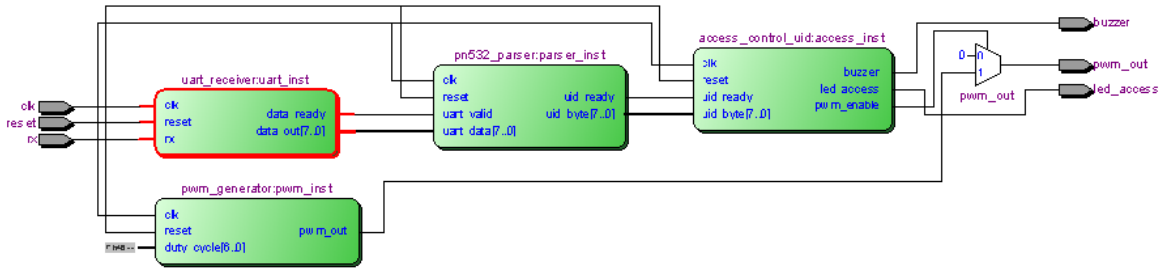


Figure III.60 : RTL du programme VHDL « top_level » code principale.

III.8.2Affectation des pins

Les numéros des pins utilisés sont présentés dans le tableau III.8.

Tableau III.8 : Numéros des pins entrées/sorties utiliséspour l'ouverture d'une porte de véhicule par le servomoteur via un RFID PN532 [III.1]

Entrées	Pins	Sorties	Pins
Clk	PIN_23	Buzzer	PIN_110
Reset	PIN_25	Led_access	PIN_84
Rx	PIN_115	Pwm_out	PIN_59

III.8.3 Résultat d'ouverture de la porte

Les deux figures suivantes présentes le principe de fonctionnement du système de contrôle d'accès RFID intégré à notre véhicule, piloté par un FPGA. Ce système a pour objectif de sécuriser l'ouverture de la porte du véhicule en autorisant uniquement l'accès aux utilisateurs disposant d'un tag RFID valide.

Dans la figure III.61, l'utilisateur présente un tag RFID non autorisé. Ce tag est lu par le module RFID PN532 en mode UART, puis comparé aux identifiants enregistrés dans la mémoire interne du système. Étant donné que le tag ne correspond à aucun identifiant autorisé, la porte du véhicule reste fermée. Aucun signal d'ouverture n'est envoyé au servomoteur, et une indication sonore, telle que l'activation d'un buzzer, signale le refus d'accès. Cette figure illustre donc un refus d'accès suite à une tentative d'authentification échouée.

Dans la figure III.62, un tag RFID autorisé est présenté. Après lecture et identification, le tag est reconnu comme valide par le système. Cela déclenche l'activation du servomoteur, qui entraîne l'ouverture mécanique de la porte du véhicule. Simultanément, une LED verte peut être allumée pour indiquer que l'accès est autorisé. Cette figure représente ainsi une authentification réussie, suivie d'une action concrète sur le système (ouverture de la porte).

Ces deux figures permettent de visualiser concrètement le comportement du système RFID embarqué dans deux situations opposées (accès refusé et accès accordé), et soulignent l'efficacité de l'intégration entre la lecture RFID, la logique de traitement sur FPGA, et le contrôle physique de la porte du véhicule.

Figure III.61 : Refus d'accès – Tag RFID non autorisé, porte du véhicule fermée.

Figure III.62 : Accès autorisé – Tag RFID reconnu, ouverture de la porte du véhicule.

III.9 Conclusion

Dans ce chapitre, nous avons développé et testé le mode de contrôle manuel du véhicule omnidirectionnel à travers une interface Bluetooth connectée à une application mobile conçue spécifiquement pour ce projet. Un buzzer est également activé lorsque la distance détectée est inférieure à un seuil critique, assurant ainsi un niveau de sécurité minimal contre les collisions. Ce mode permet à l'utilisateur de diriger le véhicule en temps réel en envoyant des commandes simples via son téléphone.

Nous avons commencé par la conception de l'affichage du message "WELCOME" sur un écran LCD au démarrage du système, confirmant que le dispositif est opérationnel. La mesure de distance à l'aide du capteur ultrasonique HC-SR04 permet de détecter des obstacles ; la valeur mesurée est ensuite affichée en temps réel sur l'écran LCD, ce qui apporte une information utile et accessible à l'utilisateur.

Dans le mode autonome, le véhicule est programmé pour avancer en ligne droite de manière continue, sans intervention de l'utilisateur. Cette progression est régulée par le capteur HC-SR04, connecté au FPGA, qui mesure en temps réel la distance entre le véhicule et les obstacles. Lorsque cette distance devient inférieure à 30 cm, une réaction automatique est déclenchée : le buzzer s'active pour signaler un danger, et le véhicule effectue une rotation vers la gauche afin d'éviter l'obstacle. Une fois l'obstruction contournée, le véhicule reprend automatiquement sa trajectoire initiale. Ce comportement repose sur une logique conditionnelle entièrement codée en VHDL, illustrant la capacité du FPGA à orchestrer plusieurs actions simultanées et synchronisées (lecture capteur, contrôle moteurs, affichage, alarme sonore).

En complément de ces fonctionnalités, nous avons intégré un système de contrôle d'accès RFID basé sur le module PN532. Ce module permet au véhicule d'identifier une carte ou un tag RFID. Lorsque l'identifiant détecté est reconnu comme valide, le FPGA déclenche un servomoteur pour ouvrir automatiquement la porte du véhicule. Cette fonction de sécurité, également implémentée en VHDL, repose sur une communication UART entre le FPGA et le module RFID,

démontrant l'intégration fluide de périphériques externes dans une architecture embarquée.

Ce chapitre a donc permis de concrétiser la construction d'un système embarqué complet, combinant interaction utilisateur, automatisation intelligente, et sécurisation par identification, soulignant la puissance et la polyvalence des FPGA pour le développement de systèmes robotiques évolutifs.

REFERENCES BIBLIOGRAPHIQUES -CHAPITRE III-

[III.1]https://onedrive.live.com/?authkey=%21ABZ%5Fj27B4HZ3Adg&id=B2CDC3A30980D5BD%2182722&cid=B2CDC3A30980D5BD
[III.2] MANSOUR Amir,« Etude et réalisation d'un système de sécurité basé sur module GSM Sim 900 via ATMEGA 328 », Mémoire de Fin d'Etudes De MASTER ACADEMIQUE en Génie électrique, UNIVERSITE Mouloud MAMMERI DE TIZI-OUZOU, 2018.
[III.3]https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf
[III.4] BENSALEM Rania BENDIF Nour El Houda KHEZAZNA Safa, « Intitulé Communication entre deux Arduinos via Bluetooth », MEMOIRE Présenté en vue de l'obtention du diplôme de : LICENCE en Télécommunications, Université BADJI MOKHTAR ANNABA, 2019.
[III.5]https://robotix.ah-oui.org/user_docs/dos11/sg90-data.pdf
[III.6]https://www.elechouse.com/elechouse/images/product/PN532_module_V3/PN532_%20Manual_V3.pdf
[III.7]https://www.mecs-press.org/ijem/ijem-v8-n2/IJEM-V8-N2-4.pdf
[III.8]Mohamed TaherRouissi, « SUIVI DE TRAJECTOIRES D'UN ROBOT MOBILE OMNIDIRECTIONNEL EN UTILISANT LE RÉSEAU DE NEURONES », MÉMOIRE PRÉSENTÉ COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN INGÉNIERIE, UNIVERSITÉ DU QUÉBEC EN ABITIBI-TÉMISCAMINGUE, 2022.
[III.9] SIRAT Abdelhak& BAHLOULI Hichem, « Génération de signaux vidéo VGA avec un FPGA pour les systèmes de vision artificielle », mémoire pour l'obtention du Diplôme de MASTER Électronique, Université Saida Dr Tahar Moulay, 2024.
[III.10] 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.

CONCLUSION GENERALE

CONCLUSION GENERALE

L'objectif principal de ce travail était de programmer un FPGA de la famille Cyclone IV d'Altera, en utilisant le langage de description matérielle VHDL, afin de concevoir et réaliser un véhicule omnidirectionnel intelligent adapté aux applications de l'intelligence artificielle embarquée. Pour ce faire, nous avons utilisé une carte de développement contenant un FPGA EP4CE10E22C8, en mettant en œuvre des fonctions de mobilité, de perception de l'environnement, de communication sans fil et d'accès sécurisé.

Dans le premier chapitre, nous avons présenté les circuits logiques programmables, en insistant sur les avantages des FPGA, notamment leur flexibilité, leur capacité de reconfiguration, et leur aptitude à s'adapter à une grande variété d'applications. Ces propriétés font des FPGA des composants idéaux pour les systèmes embarqués modernes.

Le deuxième chapitre a été consacré au langage VHDL, que nous avons décrit dans ses aspects structurels et comportementaux. Ce langage nous a permis de représenter efficacement les architectures numériques nécessaires à notre projet. Nous y avons également introduit la technologie RFID, en expliquant ses principes de fonctionnement et son importance croissante dans les systèmes d'identification et de sécurité sans fil.

Le troisième chapitre a présenté la phase pratique de notre projet. Nous avons commencé par une série d'implémentations de base en VHDL, comme l'affichage d'un message d'accueil sur un écran LCD 16x2, puis la mesure et l'affichage de distance à l'aide d'un capteur ultrasonique HC-SR04. Ensuite, nous avons conçu et réalisé un véhicule omnidirectionnel, piloté soit en mode manuel via Bluetooth avec l'application mobile développée, soit en mode autonome, où le véhicule détecte les obstacles et réagit en conséquence (activation d'un buzzer, changement de direction…).

Une fonctionnalité avancée a également été implémentée à travers l'utilisation d'un module RFID PN532 connecté en UART au FPGA. Ce module permet de sécuriser l'ouverture d'une porte motorisée par servomoteur, en ne l'autorisant qu'aux utilisateurs munis d'un badge RFID reconnu. Ce mécanisme renforce la sécurité et démontre l'intégration réussie d'une technologie d'identification sans contact au sein d'un système embarqué FPGA.

Les résultats obtenus ont validé la fiabilité et l'efficacité de l'architecture mise en œuvre. Ils confirment l'intérêt des FPGA dans la conception de robots intelligents, évolutifs et performants, tout en mettant en évidence les avantages du langage VHDL dans le développement de solutions embarquées complexes.

Ce projet constitue donc une base solide pour le développement futur de systèmes encore plus autonomes et intelligents, tels que des véhicules dotés de vision artificielle, de navigation optimisée, ou de reconnaissance d'objets via réseaux neuronaux embarqués.

En résumé, cette réalisation illustre parfaitement le potentiel des FPGA dans l'ère de l'intelligence artificielle embarquée, et ouvre des perspectives concrètes vers des applications industrielles, logistiques ou robotiques avancées.