REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA



Faculté de Technologie

Mémoire de fin d'études de Master

Filière : Electronique

Spécialité : Instrumentation

Etude et implémentation d'un système numérique en utilisant la reconfiguration partielle dans un circuit FPGA

Présenté par

M. ANNANE Abdessalem

M. STAMBOULI Mohamed

Proposé et Dirigé par

M. TOUBAL Abdelmoughni

M. MAAZOUZ Mohamed

Promotion 2020

Remerciements

Avant tout nous remercions ALLAH de nous avoir accordé des connaissances de la science et de nous avoir aidés à réaliser ce travail.

Nous adressons aussi nos remerciements à nos familles pour tout l'aide, qu'elles nous ont apportées tout le long de nos études.

Nous tenons à remercier chaleureusement et respectueusement notre promoteur M. Toubal Abdelmoughni et notre Co promoteur M.Maazouz Mohamed pour son aide et son soutien tout au long de notre projet.

Nous remercions aussi tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire de fin d'étude, sans oublier d'adresser nos remerciements à tous les enseignants du département Génie électrique notamment la spécialité instrumentation.

Nos vifs remerciements aux membres de jury pour nous avoir fait l'honneur et le plaisir d'accepter de juger notre travail.

Mohamed et Abdessalem

Dédicaces

Je dédie ce travail. Tout d'abord je tiens à remercier Dieu le tout puissant de m'avoir donné la santé, la volonté et le courage.

Ensuite, A ma très chère mère Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit.Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source De force pour affronter les différents obstacles.

A mon très cher père

Tu as toujours été à mes coté pour me soutenir et m'encourager,

Que ce travaille traduit ma gratitude et mon affection.

A mes très chères frères el Hacen et el Houcien et mes belle-sœur Fella et kawter.

A mes très chères cousin Rafik et Salaheddin et a tout la famille Annane.

A Mohamed Stambouli, Noureddine, Salah, Ibrahim.

Puisse dieu vous donne santé, bonheur, courage et surtout réussite.

Abdessalem

Dédicaces

Je dédie ce travail. Tout d'abord je tiens à remercier Dieu le tout puissant de m'avoir donné la santé, la volonté et le courage.

Ensuite, à mon père et ma mère pour leur soutien et leurs encouragements et qu'aucune dédicace ne pourra compenser leur sacrifice pour mon bien être et mon bonheur.

A mes chers frères Abdallah et Mahmoud pour lequel j'ai beaucoup d'estimes.

A mes sœurs Asma et Romayssa que je lui souhaite une longue vie pleine de joie et de réussite.

Une Spécial Dédicace A Cette Personne Qui Compte Enormément Pour Moi, Et Pour Qui Je Porte Beaucoup De Tendresse Et De Respect. A TOI Aya

A mes oncles, mes tantes et toute ma famille.

A mes amis, en particulier mon collègue du présent projet Abdessalem Annane qui n'a jamais manqué de travailler et ceux de la promotion de la 2ème année master systèmes de instrumentation.

A notre encadreur M.Toubal Abdelloughni Et à tous ceux qui nous ont soutenus pour l'achèvement de ce projet avec réussite.

MOHAMED

الملخص

يمكن للبنى القابلة لاعادة التشكيل المستندة الى مصفوفة البوابات المنطقية القابلة للبرمجة (FPGA) تقديم حلول مناسبة للعديد من التطبيقات لأنها تسمح بتغيير أداء جزء من الهندسة الكلية للدائرة المنطقية بينما تستمر بقية الدائرة في العمل بشكل طبيعي.

الهدف من هذا المشروع هو تصميم نظام أجهزة / برامج لاختبار اعادة التكوين الديناميكي الجزئي. لقد تم استخدام اعادة التكوين الديناميكي الجزئي لتغيير وضائف الأزرار المتواجدة على منصة (FPGA) من نوع (Nexys4 DDR) للتحكم في زاوية و سرعة محرك السيرفو.

الكلمات المفتاحية : الهياكل القابلة لإعادة التكوين ,FPGA , اعادة التكوين الديناميكي الجزئي , محرك السيرفو.

Résumé

Les architectures reconfigurables à base de FPGA sont capables de fournir des solutions adéquates pour plusieurs applications vues qu'elles permettent de modifier le comportement d'une partie du FPGA pendant que le reste du circuit continue de s'exécuter normalement.
L'objectif de ce projet est de concevoir un système matériel/logiciel pour tester la reconfiguration dynamique partielle. La reconfiguration dynamique partielle est utilisée pour changer dynamiquement les configurations des boutons poussoirs de la carte FPGA de type Nexys4 DDR pour commander l'angle et la vitesse de rotation d'un servomoteur.

Mots clés : Les architectures reconfigurables, FPGA, Reconfiguration dynamique partielle, Servomoteur.

Abstract

FPGA-based reconfigurable architectures can provide appropriate solutions for several applications as they allow for changing the configuration of a part of the FPGA while the rest of the circuit continues to run normally.
The objective of this project is to design a hardware/software system to test dynamic and partial reconfiguration. In this project, Dynamic partial reconfiguration is used to dynamically change the push button configurations of the Nexys4 DDR FPGA board to control the angle and the speed of rotation of a servo motor.

Keywords: reconfigurable architectures, FPGA, Dynamic Partial Reconfiguration (DPR), Servomotor.

TABLE DES MATIERS

LISTE DES FIGURES V
LISTE DES TABLEAUX XII
LISTE DES ACRONYMES.....XIII
Introduction générale.....1

Chapitre I : Généralités sur les circuits FPGAs

1. Introduction.....3
2. Les circuits Programmables FPGA 3
   2.1. Généralités.....3
   2.2. Critères de choix du circuit programmable FPGA 5
3. Architecture interne des FPGA 6
   3.1. Logique des FPGA.....8
   3.2. Interconnexion 10
   3.3. Entrée et sortie 11
   3.4. Horloge.....12
   3.5. Configuration.....13
   3.6. Consommation d'énergie 14
4. Différents domaines d'application des FPGA.....15
5. Technologies de programmation des FPGA 16
   5.1. Technologie à base de RAM (XILINX et ALTERA).....16
   5.2. Technologie à base d'EEPROM ou FLASH (LATTICE et ACTEL).....17
   5.3. Technologie à base d'ANTI-FUSIBES(ACTEL).....17
6. Etude des FPGA de la famille Xilinx 17
   6.1. Présentation et généralités sur les FPGAs de la famille Xilinx**Erreur ! Signet non défini.**
7. Conclusion 18

Chapitre II : La reconfiguration partielle dans les circuits FPGAs

1. Introduction 19
2. Configuration d'un FPGA 19
3. La reconfiguration des FPGA 20
3.1. Reconfiguration statique 20
3.2. Reconfiguration dynamique 21
3.2.1. *Reconfiguration dynamique globale* 22
3.2.2. *Reconfiguration dynamique partielle* 23
4. Processus de conception de la reconfiguration partielle 24
5. Les avantages de la reconfiguration partielle 26
6. conclusion 27

Chapitre III : Implémentation de la commande d'un servomoteur sur FPGA en utilisant la reconfiguration dynamique partielle

1. Introduction 28
2. Présentation du projet 28
2.1. Principe du fonctionnement 29
2.2. L'affichage à sept segments 31
3. Composants matériels 31
3.1. La carte Nexys4 DDR 31
3.2. Le servomoteur SG-90 34
3.3. Organisation des programmes VHDL 36
3.3.a. La partie statique 36
3.3.b. La partie reconfigurable 38
4. L'implémentation de l'architecture 40
4.1. Vivado Design Suite 40
4.2. Création du projet 42
4.2.a *Définition des partitions reconfigurables* 43
4.2.b *Finalisation de la structure du projet de reconfiguration partielle* 45

4.2.c *Edition de modules reconfigurables* 45

4.2.d *Edition des exécutions de configuration* 46

4.2.e *Mise en œuvre de la conception des RP* 47

4.2.f *Génération de bitstreams* 47

4. Configuration FPGA 48

4.1. Bitstream 49

4.2. Configuration JTAG 50

5. Expérimentations et résultants 50

5.1. Installation expérimental 50

5.2. L'implémentation sur la carte 53

6. Conclusion 53

Conclusion générale 54

Références bibliographiques : 55

Liste des figures

Figure I.1. *Structure interne du FPGA*....4
Figure I.2. *Critères de choix du circuit logique programmable FPGA*....5
Figure I.3. *Architecture interne d'un FPGA*....6
Figure I.4. *Différentes architectures des FPGA*....8
Figure I.5. *Une cellule logique est l'élément de base d'un FPGA*....9
Figure I.6. *Bloc d'entrée / sortie générique*....12
Figure I.7. *Réseaux de distribution d'horloge. À gauche : structure en H ; à droite : colonne*....13
Figure I.8. *Configuration de FPGA basée sur la RAM statique*....14
Figure I.9. *Inverseur CMOS*....15
Figure I.10. *Caractéristiques de technologie SRAM*....16
Figure I.11. *Caractéristiques de technologie FLASH*....17
Figure II.1. *Mécanisme de la reconfiguration statique d'un FPGA.*....21
Figure II.2. *Mécanisme de la reconfiguration dynamique d'un FPGA*....22
Figure II.3. *Reconfiguration dynamique globale d'un FPGA.*....23
Figure II.4. *Reconfiguration dynamique partielle d'un FPGA.*....23
Figure II.5. *Différence entre (a) Configuration complète des bitstreams (b) Reconfiguration*....24
Figure II.6. *Méthodes de configuration pour la conception des RP (a) ICAP (b)*....25
Figure III.1. *La commande d'un Servomoteur A Base d'FPGA*....28
Figure III.2. *Plage de Fonctionnement du servomoteur*....30
Figure III.3. *Le fonctionnement des boutons poussoirs*....30
Figure III.4. *L'affichage à sept segments*....31
Figure III.5. *Légende de la fonction Nexys4 DDR*....33
Figure III.6. *Fonctionnement du servomoteur*....35
Figure III.7. *Servomoteur*....35

Figure III.8. *Organigramme simplifie de la partie statique*.....37
Figure III.9. *Organigramme de RM1* 38
Figure III.10. *Organigramme de RM2* 39
Figure III.11. *Organigramme de RM3* 40
Figure III.12. *Les bitstreams et la reconfiguration dynamique partielle de notre projet* 42
Figure III.13. *Activation de la reconfiguration partielle*.....43
Figure III.14. *Création d'une partition reconfigurable* 44
Figure III.15. *Définition de la partition reconfigurable et du premier module reconfigurable*.....44
Figure III.16. *L'assistant de reconfiguration partielle dans le navigateur de flux*.....45
Figure III.17. *Création d'un nouveau module reconfigurable*.....46
Figure III.18. *Configuration automatique* 46
Figure III.19. *Propriétés de reconfiguration partielle sur un Pblock (7 séries)* 47
Figure III.20. *Génération des bitstream* 48
Figure III.21. *Options de configuration du Nexys4 DDR*.....49
Figure III.22. *Analog Discovery2* 51
Figure III.23. *Les signaux de commande du servomoteur*.....52
Figure III.24. *Installation du système*.....52

LISTE DES TABLEAUX

Table III.1. *Les composants de la carte Nexys4 DDR* 33
Table III.2. *Description des fils de connexion du servomoteur* 35

LISTE DES ACRONYMES

ASCI : Application Specific Integrated Circuit

CLB : Configurable Logical Bloc

CMOS : Complementary Metal Oxide Semiconductor

DPR : Dynamic Partial Reconfiguration

DSP : Digital Signal Processing

EEPROM : Electrically-Erasable Programmable Read-Only Memory

FPGA : Field Programmable Gate Array

HDL : Hardware Description Language

IC : Integrated Circuit

ICAP : Internal Configuration Access Port

IOB : In Out Bloc

ISE : Integrated Synthesis Environment

JTAG : Joint Test Action Group

LE : Logic Element

LED : l ight emitting diode

LC : Logic Cell

LUT : Look Up Table

MAC : Media Access Control

PCAP : Processor Configuration Access Port

PCI : Peripheral Component InterConnect

PIP : Programme Interconnecte Points

PR **:** Partial Reconfiguration

PS **:** Process System

RAM **:** Random Access Memory

RISC **:** Reduced Instruction Set Computing

ROM **:** Read-Only Memory

RVB **:** Rouge Vert Bleu

SOC **:** System on a Chip

SPI **:** Serial Peripheral Interface

SRAM **:** Static Random-Access Memory

USB **:** Universal Serial Bus

VGA **:** Video graphics array

VHDL **:** VHSIC Hardware Description Language

VLSI **:** Very Large-Scale Integration

Introduction générale

Avec la convergence de l'électronique, de l'informatique et des télécommunications, les technologies de l'information occupent une place préférentielle dans notre quotidien. En effet, l'évolution des systèmes embarqués et des systèmes sur puces permettent de développer des fonctions de plus en plus complexes sans négliger la vitesse de traitement et la consommation d'énergie. Les circuits intégrés spécifiques à une application appelés ASIC (Application-Specific Integrated Circuit) représentent une solution optimale sur le plan du taux d'intégration au détriment d'un coût de développement et de fabrication important, et d'un degré de flexibilité loin d'être idéal. Par ailleurs, avec la réduction de l'échelle de gravure des circuits, la consommation statique augmente considérablement ce qui devient un problème majeur pour les applications embarquées. Les ASICs atteignent une nouvelle limite et ils ne sont viables que pour certaines applications dédiées. Ainsi toute modification ou mise à jour d'un ASIC se soldera par la fabrication d'un nouveau circuit. Cependant, les systèmes électroniques embarqués ont besoin de circuits capables de répondre aux exigences de coût, de performance, de flexibilité et de consommation. Les circuits logiques programmables de type FPGA (Field-Programmable Gate Array) sont une solution qui dispose de performances acceptables et d'un temps de mise en œuvre réduit par rapport aux ASICs. En outre, les FPGAs représentent une solution idéale pour implémenter la reconfiguration dynamique partielle du fait de leur grande flexibilité par rapport aux SoCs (System on a Chip) traditionnels. La reconfiguration dynamique partielle (RDP) est un type de reconfiguration supporté par certains FPGAs et permet de reconfigurer une partie du FPGA sans perturber le fonctionnement du reste. Ce type de reconfiguration permet par exemple de remplacer un module matériel, réalisant une tâche de l'application par un autre. Cela permet de cibler de larges applications en utilisant un nombre limité de ressources. Un tel mécanisme permet d'améliorer la performance globale en faisant se chevaucher l'exécution et la reconfiguration des différentes portions du système.

Grâce à la reconfiguration dynamique partielle, on peut supprimer certaines tâches matérielles du FPGA et charger de nouvelles tâches à leurs places. L'implémentation de ces nouvelles tâches matérielles sur des régions reconfigurables permet d'augmenter l'adaptabilité du système. Les tâches matérielles doivent être entièrement portables afin d'être facilement insérées dans n'importe quel système, mais elles peuvent être de tailles différentes. Par conséquent, le placement des tâches matérielles de tailles différentes sur des régions

reconfigurables doit être réalisé au mieux pour optimiser l'utilisation des ressources FPGA.

L'objectif principale de ce projet et la maîtrise de la reconfiguration dynamique des circuits FPGA et non pas la commande du servomoteur.

Ce mémoire de fin d'étude est élaboré en trois chapitres, le premier chapitre présente une explication détaillée sur les circuits logiques programmables FPGA, après on donnera des généralités sur les circuits FPGA : description des critères de choix, et explication de leurs architectures interne. On abordera aussi dans ce chapitre les domaines d'applications des circuits FPGA et la technologie de programmation de ces derniers, on terminera par une étude sur les FPGA de la famille Xilinx.

Le deuxième chapitre sera entièrement dédié aux techniques de reconfiguration des FPGA. Après une introduction, une présentation des modes de reconfiguration des FPGA, puis un aperçu sur la reconfiguration dynamique partielle.

Le troisième chapitre est consacré à une implémentation d'une architecture sur circuit FPGA qui utilise la reconfiguration dynamique partielle. Cette architecture, qui est basée sur une carte FPGA de type Nexy4 DDR, sert à la commande d'un servomoteur en utilisant la reconfiguration dynamique partielle.

Dans ce dernier chapitre on va aborder les moyennes matérielles et logicielles ainsi que les méthodes utilisés dans notre projet. On abordera aussi la conception et l'implémentation sur la carte FPGA. Pour finir, des expérimentations et des résultats vont être présentées.

Enfin, nous finissons notre travail par une conclusion générale, résumant notre travail.

Chapitre I :

Généralités sur les circuits FPGAs

1. Introduction

Durant ces dernières années l'évolution de l'électronique et l'industrie a connu une évolution gigantesque grâce à la concurrence entre les fabricants, ce qui a conduit en parallèle à un énorme développement de divers types de circuit logique programmable, les FPGA (Field Programmable Gate Array) sont l'un des circuits logiques ,ces derniers sont capables de réaliser plusieurs fonctions logiques complexes avec la souplesse et la flexibilité apportée par la logique programmable et ça dans un seul circuit, ils sont les réseaux logiques programmables les plus performants qui existent en ce moment.

Dans ce chapitre nous allons commencer par une explication détaillée sur les circuits logiques programmables FPGA, après on abordera des généralités : description de critère de choix, explication de la configuration, la reconfiguration des FPGA et leurs architectures interne.

On abordera aussi dans ce chapitre les domaines d'application des circuits FPGA et la technologie de programmation de ces derniers, on terminera par une étude sur les FPGA de la famille Xilinx.

2. Les circuits Programmables FPGA

2.1. *Généralités*

Les premiers circuits intégrés reconfigurables de type FPGA ont été vendus par Xilinx en 1985.Depuis, d'autres fabricants sont apparus sur le marché, comme ALTERA, ACTEL, Texas Instrument …etc. La complexité des FPGA s'est développée par conséquent leurs possibilités d'utilisation ont augmenté jusqu'à concurrencer les circuits spécifiques pour atteindre de petites tailles de lots (jusqu'à plusieurs milliers de composants), c'est pourquoi l'utilisation des FPGAs est envisageable dans les applications aéronautiques et spatiales.
Ils offrent une grande facilité d'utilisation et réduisent le coût de développement des équipements électroniques embarqués. Les FPGAs reconfigurables basés sur SRAM (Static Random Access Memory) offrent également la possibilité d'optimiser la fonctionnalité de ces circuits, ce qui permet la maintenance à distance (que n'autorisent pas les circuits spécifiques). [1]

Les FPGA sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs, 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.

Le progrès de ces technologies permet de faire des composants toujours plus rapides et à plus haute intégration, ce qui permet de programmer des applications importantes, cette technologie permet d'implémenter un grand nombre d'applications et offre une solution d'implémentation matérielle à faible coût pour des compagnies de taille modeste car le coût de développement d'un circuit intégré spécifique implique un trop lourd investissement [2].

En une seule puce de circuit intégré (IC) des milliers de composant discret incorporant des millions de portes logiques peuvent être substitués par un seul FPGA, une matrice de blocs logiques configurables (CLB) entourée d'une périphérie de bloc d'E/S est les constituants de ressources internes d'une puce FPGA montré dans la figure I.1.

Au sein de la matrice FPGA les signaux sont acheminés par des commutateurs d'interconnexion programmable et des routes filaires [3].

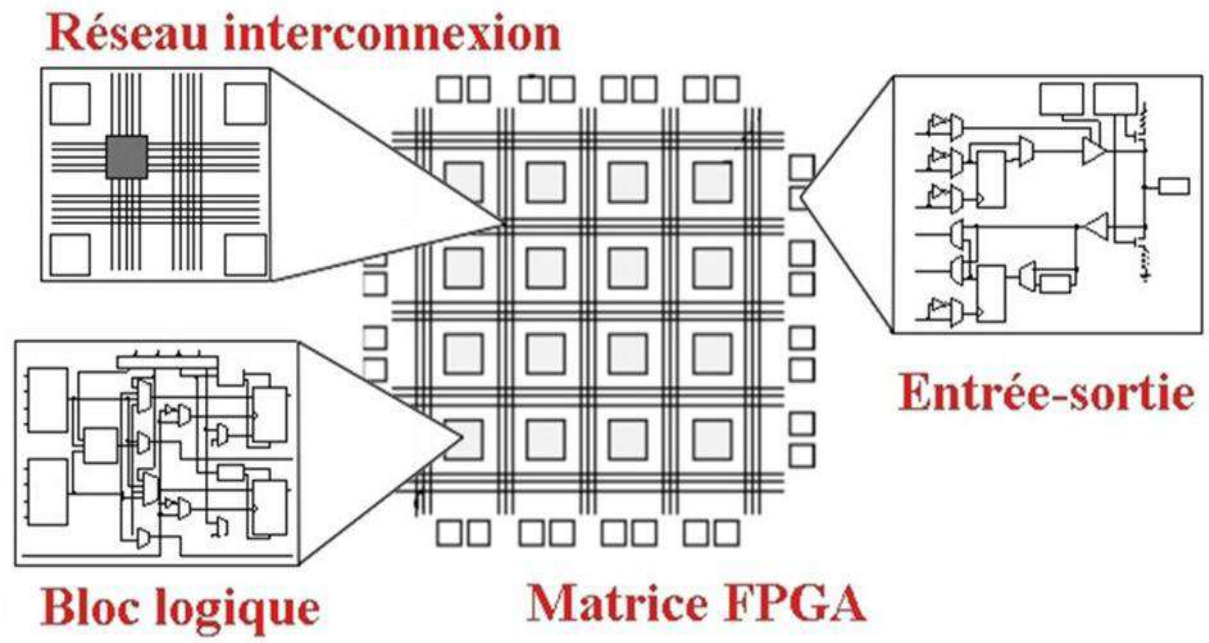


Figure I.1. *Structure interne du FPGA [3]*

Dans un FPGA, les blocs logiques sont implémentés à l'aide de portes d'entrée multiples, ce qui lui confère une conception plus compacte par rapport à une implémentation avec une logique AND-OR à deux niveaux, FPGA fournit à son utilisateur un moyen de configurer :

L'intersection entre les blocs logiques AND et la fonction de chaque bloc logique. [3]

2.2. *Critères de choix du circuit programmable FPGA*

Grâce aux progrès de la technologie VLSI (Very-Large-Scale Integration), des FPGAs ont récemment été développés, l'émergence de ce type de circuit est une révolution dans les systèmes digitaux et a ouvert l'horizon au traitement numérique inaccessible auparavant. Le premier lot de circuits FPGA est apparu à la fin des années 1980, ce sont des circuits intégrés configurables dans un laps de temps relativement court pour implémenter à moindre coût toute fonction logique "câblée" en programmant leurs cellules logiques et leur interconnexion, en évitant d'épuiser les ressources FPGA. En général, les circuits FPGAs à haute densité peuvent contenir jusqu'à des millions d'éléments programmables [4].

Pour réussir dans les applications basées sur FPGA et obtenir un système plus efficace avec la plus faible consommation d'énergie, un certain nombre de règles doivent être observées, telles que :

- Comprendre les caractéristiques du FPGA ciblé pour s'assurer qu'il répond aux besoins du projet.
- Développer des méthodes de conception.
- Maîtriser les outils de mise en œuvre et sélectionner des outils de qualité.

La conception de circuits FPGA est un défi, son objectif est de trouver un compromis approprié entre densité, flexibilité et performances temporelles.

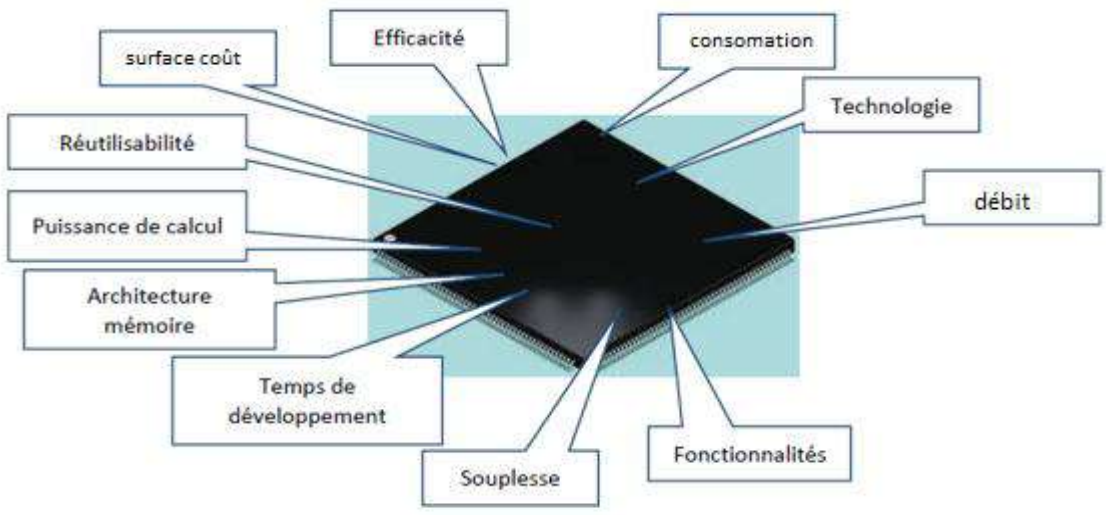


Figure I.2. *Critères de choix du circuit logique programmable FPGA [4]*

3. Architecture interne des FPGA

La figure I.3 illustre la structure et les composants de base d'un FPGA à usage général. La logique programmable se compose d'un ensemble de blocs à grain fin utilisés pour implémenter la logique d'application. Il est parfois appelé la structure d'un FPGA, les blocs logiques sont généralement basés sur une architecture de table de consultation (LUT : Look-Up Table), ce qui leur permet d'implémenter n'importe quelle fonction arbitraire de l'entrée. Les blocs logiques sont généralement carrelés dans une structure de grille et interconnectés par une matrice de routage programmable, permettant à ces blocs d'être connectés dans n'importe quelle configuration. Les blocs d'entrée et de sortie (IOB) établissent une interface entre les composants internes ou les cœurs FPGA et les périphériques externes. Le routage signifie que presque tous les signaux peuvent être routés vers n'importe quelle broche d'E/S sur l'appareil. En plus de ces fonctions de base, la plupart des FPGA fournissent également une certaine forme de synchronisation d'horloge pour contrôler la synchronisation des signaux d'horloge par rapport aux sources externes. [3]

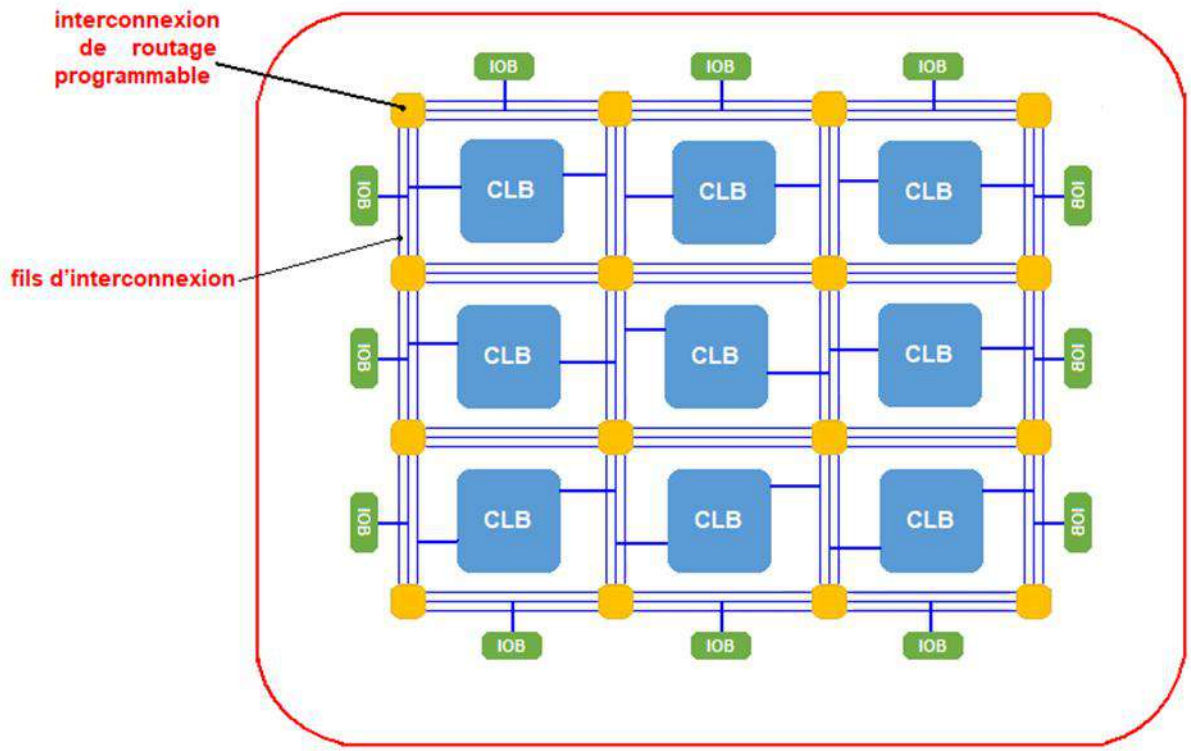


Figure I.3. *Architecture interne d'un FPGA [3]*

On peut configurer un bloc logique d'un FPGA d'une façon qu'il nous donne une fonction simple et similaire à celle d'une porte logique ou fonction complexe similaire à celle d'un microprocesseur, il a pour but aussi de mettre en œuvre différentes combinaisons de fonction logique combinatoire et séquentielle. [3]

- Les macro-cellules internes sont appelées :
  - **CLB** qui est la dénomination adoptée par Xilinx et abréviation anglaise de (**C**onfigurable **l**ogic **B**loc), signifiant bloc logique configurable.
  - **LC** qui est le nom choisi par cyprès et abréviation anglaise de (Logic **C**ell), signifiante cellule logique.
  - **LE** qui est l'appellation d'Altéra abréviation anglaise de (Logic **E**lement) signifiant élément logique.
- Les macro-cellules sur la périphérie sont appelées : **IOB** abréviation anglaise de (**I**nput **O**utput **B**loc), signifiant bloc logique d'entrées sorties.
- L'ensemble des points de connexion est appelé PIP, abréviation anglaise de (Programme Interconnecte Points). La granularité des FPGA par les macrocellules CLB nous permet d'implémenter des fonctions logiques (combinatoires ou séquentielles) complexes car chaque CLB est constitué d'une partie combinatoire et d'une partie séquentielle. Chaque fonction est décomposée en petites fonctions booliennes qui peuvent être contenues par de petites cellules élémentaires SLICES.

Ces dernières comportent des LUTS pour la partie combinatoire et une ou des bascules (généralement du type D) pour la partie séquentielle. [6]

Les architectures existantes peuvent être regroupées en trois grandes catégories suivant la manière dont les blocs logiques sont organisés comme le montre la figure suivante :

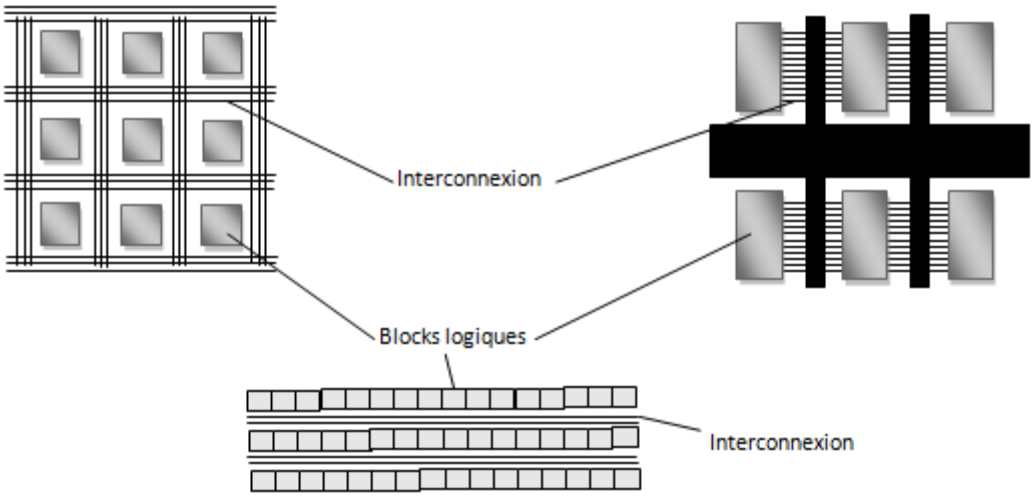


Figure I.4. *Différentes architectures des FPGA [6]*

Le réseau de distribution d'horloge fournit des signaux d'horloge à toutes les parties du FPGA, tout en limitant le décalage d'horloge entre les différentes parties de la conception, il existe également une logique dédiée pour charger la configuration sur le FPGA, cette logique ne fait pas directement partie de la conception utilisateur, mais de la charge supplémentaire nécessaire pour la programmation et la compatibilité FPGA, fournissant ainsi un mécanisme pour charger la conception utilisateur sur l'FPGA. Les sections suivantes décrivent en détail chacun de ses composants. [3]

3.1. *Logique des FPGA*

La cellule logique est la plus petite unité logique d'un FPGA, elle se compose généralement d'une petite table de conversion LUT (table de consultation) et d'un verrou de sortie. Comme le montre la figure I.5. Le verrou est utilisé pour enregistrer la sortie. Sur certains FPGA, deux sorties peuvent être fournies, comme le montre la figure 5, d'un autre côté, un multiplexeur peut être utilisé pour sélectionner la sortie d'un signal enregistré ou d'un signal non enregistré.

Les registres peuvent être utilisés pour construire des machines à états finis, des compteurs et enregistrer des données dans le système de pipeline. La LUT peut implémenter n'importe quelle fonction arbitraire de son entrée. En connectant la sortie X à l'entrée de la couche suivante pour mettre en cascade plusieurs niveaux de la LUT, des fonctions plus complexes peuvent être obtenues. Les fabricants de FPGA doivent faire des compromis lors du choix de

la taille de la LUT. Envisager d'utiliser deux entrées pour implémenter une simple porte AND. Pour les LUT de grande taille, la majeure partie de la logique ne sera pas utilisée; un LUT plus petit permettra d'utiliser plus de ressources. Cependant, lors de l'utilisation d'une petite LUT pour implémenter des fonctions complexes, plusieurs unités logiques doivent être mises en cascade, ce qui augmentera considérablement le délai de propagation. Pour les fonctions complexes, il est préférable d'utiliser une table de correspondance plus grande. Pour équilibrer cela, la surface de silicium de la table augmentera de façon exponentielle à mesure que le nombre d'entrées augmente, ce qui la rend plus efficace lors de la division de fonctions complexes en plusieurs LUT.

Un FPGA uniforme se compose d'un grand nombre de blocs logiques identiques. Cependant, la plupart des FPGA actuels n'utilisent plus cette structure homogène, mais mélangent des blocs logiques à grain fin avec des blocs à grain grossiers pour accélérer les opérations courantes. Cela présente des avantages et des inconvénients de conception.

Les systèmes hétérogènes sont généralement plus rapides parce que le coût des blocs à gros grains ne sera pas le même que ceux des blocs à gros grains, mais les appareils hétérogènes sont également plus difficiles à programmer, et si le mélange de blocs spéciaux est incorrect, de nombreuses fonctions peuvent ne pas être adéquates utilisation. [3]

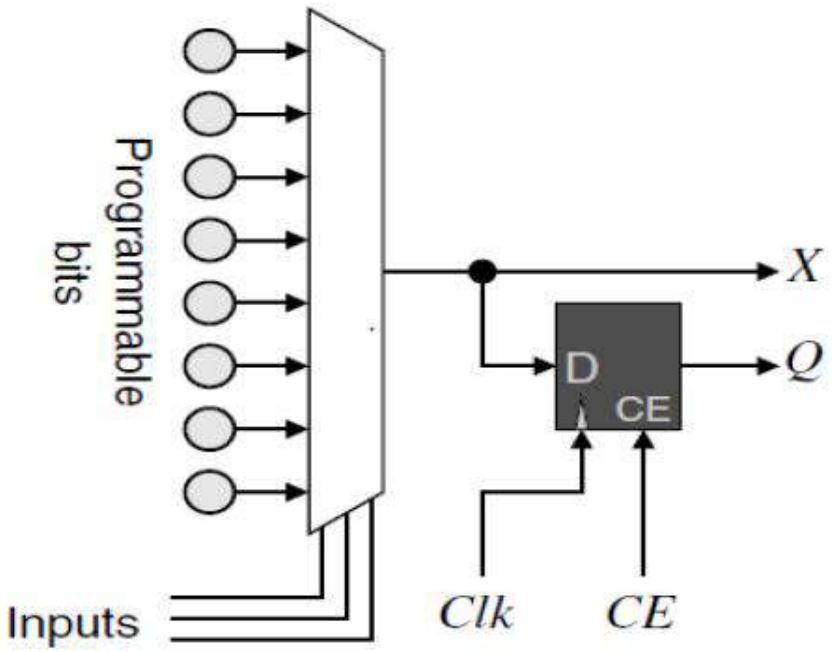


Figure I.5. *Une cellule logique est l'élément de base d'un FPGA [8]*

3.2. *Interconnexion*

L'interconnexion programmable a pour but de permettre l'utilisation de divers blocs logiques dans une connexion flexible pour réaliser les fonctions requises, il est impossible de fournir une connexion directe dédiée de toutes les sorties possibles à toutes les entrées possibles. Dans une application donnée, seule une petite partie de ces connexions est nécessaire, cependant, il peut ne nécessiter aucune connexion potentielle, la solution consiste à utiliser un plus petit ensemble de ressources de connexion (lignes de routage). Partage flexible pour créer la connexion nécessaire pour une application donnée. Étant donné que l'interconnexion se produit sur le silicium, il y a un compromis, si seules quelques lignes de routage sont fournies, l'espace logique disponible est plus grand, mais en raison de ressources d'interconnexion insuffisantes, il est impossible de construire un circuit compliqué. Sinon, si trop de lignes de routage sont fournies, de nombreuses lignes de routage ne seront pas utilisées, ce qui réduit la zone disponible pour la logique. Il y a un choix optimal entre ces deux extrêmes, qui est de fournir suffisamment des lignes de routage pour la plupart des applications sans prendre trop de place. La plupart des réseaux de connexion FPGA sont basés sur une structure de grille, similaire à la structure illustrée à la figure 3. À chaque intersection de grille, une forme de commutateur latéral permet des connexions programmables entre les lignes de routage horizontal et vertical.

Un autre facteur à considérer est le retard de propagation à travers le réseau d'interconnexion, chaque commutateur par lequel passe le signal augmente le retard, cela conduit à une structure segmentée dans laquelle toutes les lignes de routage ne sont pas échangées à chaque nœud, de plus, des connexions directes spécifiques peuvent être fournies entre des blocs logiques adjacents pour réduire le nombre de signaux qui doivent traverser la matrice d'interconnexion.

Bien que disposer de diverses ressources d'interconnexion puisse augmenter la flexibilité et la vitesse, cela peut compliquer le processus de recherche de bonnes routes, en particulier pour les conceptions qui utilisent la plupart de la logique disponible sur les FPGA.

L'important est qu'une seule sortie entraîne la ligne de routage. Si vous connectez les deux sorties ensemble, l'une haute et l'autre basse, l'alimentation sera effectivement court-circuitée, endommageant la puce. L'un des rôles des outils de programmation FPGA est de s'assurer que cela n'arrive jamais, les appareils utilisant une structure de routage de bus doivent limiter le courant du transistor de commande, ou utiliser des circuits de drain ouvert et pull-up passif.

Les deux méthodes affectent la vitesse et la consommation d'énergie [3].

3.3. *Entrée et sortie*

Les blocs d'entrée et de sortie permettent des connexions de signaux entre l'FPGA et les périphériques externes, La flexibilité des FPGAs les oblige à pouvoir se connecter à divers autres composants, la plupart des broches d'entrée / sortie peuvent être programmées comme entrées, sorties ou les deux, et peuvent être directement connectées à la mémoire logique interne du FPGA, ou via des registres. Les périphériques d'E / S universels peuvent fournir cette gamme de fonctions, comme le montre la figure I.6, chaque multiplexeur est contrôlé par un bit de configuration programmable.

La principale complexité des E/S provient du grand nombre de signaux standard différents utilisés par différents appareils et assure la synchronisation correcte entre le FPGA et les composants hors puce.

Lors de la connexion de signaux à haute vitesse, il est important de prendre en compte l'impact sur la ligne de transmission. Généralement, toute connexion plus longue qu'un huitième du temps de montée doit avoir une terminaison appropriée pour minimiser la réflexion du signal. Comme les FPGA sont devenus plus grands et plus rapides, il est devenu courant de fournir une terminaison programmable sur la puce, s'il n'est pas fourni, vous devez vous assurer que la terminaison correcte est établie hors puce afin de commuter rapidement les signaux. De nombreux protocoles de communication avancés ne nécessitent qu'un grand nombre de ressources FPGA pour les protocoles de gestion. Bien que cela puisse être implémenté dans une structure FPGA, la conception est généralement très complexe et il est d'utiliser une puce d'interface dédiée. Ces puces gèrent généralement l'accès aux médias (MAC) et la signalisation physique (PHY), et elles ne nécessitent que des signaux logiques standards dans l'interface avec le FPGA.

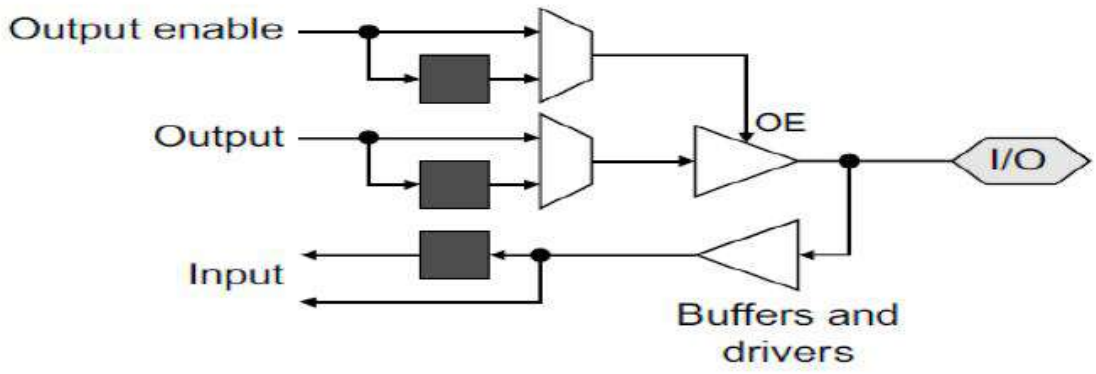


Figure I.6. *Bloc d'entrée / sortie générique [3]*

Certains FPGA intègrent des circuits MAC et PHY dédiés aux protocoles de communication courants, tels que PCI Express et Gigabit Ethernet. La plupart des FPGA fournissent une gamme de connexions d'E / S conventionnelles et dédiées.

3.4. *Horloge*

Les FPGA sont principalement conçus comme des appareils synchrones, ils nécessitent donc un signal d'horloge. Bien que plusieurs horloges puissent être utilisées dans la conception, les registres, la mémoire synchrone et les modules d'E / S ne peuvent être contrôlés que par une seule horloge. Le domaine d'horloge comprend tous les circuits contrôlés par une horloge spécifique. Étant donné que l'horloge peut piloter de nombreuses entrées sur l'ensemble du FPGA, il existe deux options spéciales. Les caractéristiques des lignes d'horloge qui les distinguent des autres interconnexions FPGA. Tout d'abord, il est important que tous les registres du domaine d'horloge soient synchronisés en même temps, le décalage d'horloge est la différence de temps entre un signal d'horloge et un registre continu de graphiques, tout écart réduira la marge de synchronisation et affectera la vitesse d'horloge maximale de la conception. Bien que l'utilisation de réseaux de distribution d'horloge en arbre H-tree ou en arbre X-tree puisse minimiser l'asymétrie, ils ne sont pas très cohérents avec les structures de grille FPGA. Au lieu de cela, les structures de la colonne vertébrale et des côtes sont couramment utilisées (figure I.6), où les épines sont utilisées pour affecter l'horloge à la ligne, puis l'horloge est utilisée pour affecter l'horloge aux éléments de la ligne. Côtes levées

(Lamoureux et Wilton, 2006), Une série de mémoires tampons sont utilisées dans le réseau de distribution pour gérer les grands fan-out, en fait, c'est beaucoup plus compliqué car le réseau doit prendre en charge plusieurs domaines d'horloge et de nombreux FPGA ont également un mélange d'horloges locales et globales [3].

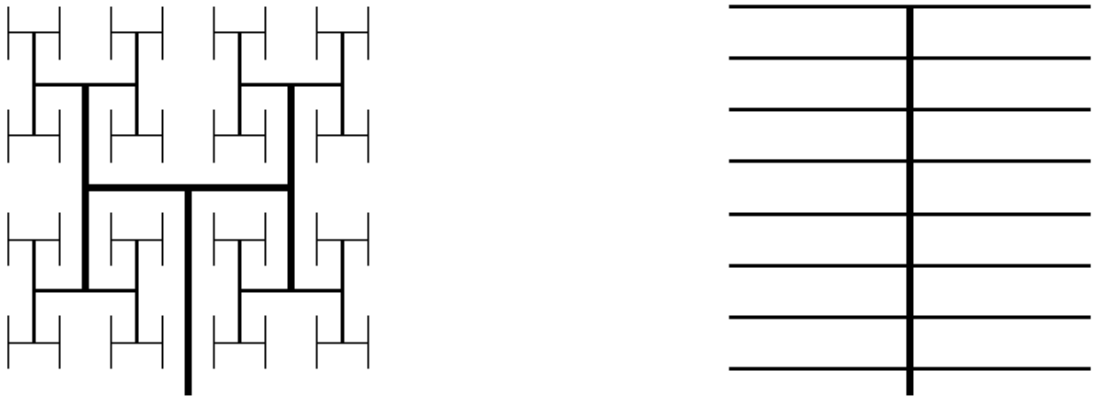


Figure I.7. *Réseaux de distribution d'horloge. À gauche : structure en H ; à droite : colonne [3]*

3.5. *Configuration*

Tous les commutateurs, structures et options FPGA sont contrôlés par des bits, plus généralement, les données de configuration sont contenues dans l'unité de stockage statique du FPGA (SRAM), bien que certains fournisseurs proposent des dispositifs avec programmation de mémoire flash, les FPGA programmés avec une mémoire flash ne sont pas volatils et conservent leur configuration après la mise hors tension afin de pouvoir commencer à fonctionner immédiatement après la mise sous tension, l'FPGA à mémoire statique perdra sa configuration à la mise hors tension, ce qui nécessite de recharger la configuration du fichier à chaque mise sous tension du système, l'avantage de la mémoire statique est qu'elle peut être reprogrammée indéfiniment.

La mémoire flash a généralement un nombre limité de cycles d'écriture. Configurer l'FPGA en chargeant le fichier de configuration dans la mémoire interne, en règle générale, cela est obtenu en transférant le fichier de configuration vers le FPGA en série ou en parallèle 8,16 ou parfois 32 bits à la fois.

La mémoire externe illustrée dans la figure I.8, a sa carte configurée pour configurer

automatiquement le FPGA au démarrage. Le FPGA peut également être programmé en série via l'interface JTAG (Joint Test Action Group), qui est généralement utilisée lors de la programmation du FPGA à partir du système hôte, chaque fois que le système est mis sous tension, la configuration doit être transférée vers le FPGA, qui expose le fichier de configuration, ce qui rend la propriété intellectuelle qu'il contient vulnérable aux attaques de pirates ou aux flips d'ingénierie.

Les FPGA haut de gamme surmontent cette limitation en chiffrant les fichiers de configuration et en déchiffrant les fichiers de configuration après leur placement dans le FPGA, certains FPGA prennent également en charge la lecture des fichiers de configuration depuis le FPGA, cette fonctionnalité est utile pour déboguer les conceptions avec les outils appropriés, mais n'est généralement pas utilisée en fonctionnement normal [3].

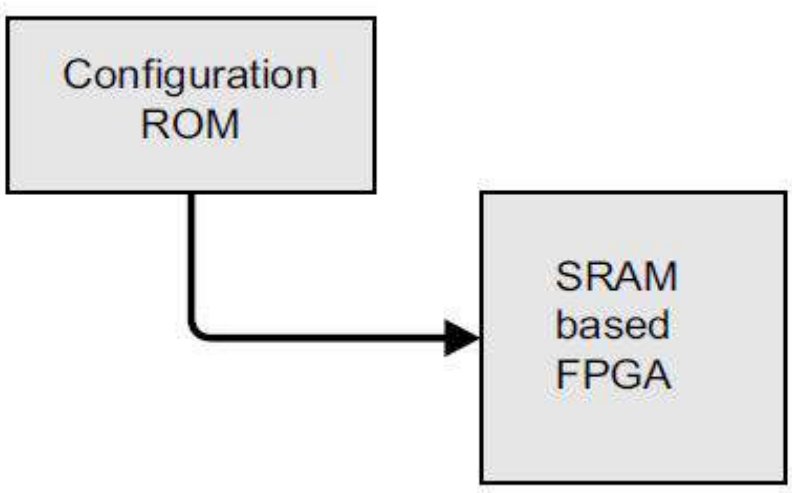


Figure I.8. *Configuration de FPGA basée sur la RAM statique [3]*

3.6. *Consommation d'énergie*

Dans les applications intégrées, la consommation d'énergie du système, en particulier du FPGA est critique, la consommation électrique du FPGA peut être divisée en deux parties : la consommation électrique statique et la consommation électrique dynamique.

La puissance dynamique est la puissance requise pour commuter le signal de 0 à 1 (ou vice versa) Au contraire, juste parce que l'appareil est allumé, l'énergie statique est l'énergie qui peut être consommée sans commutation. La plupart des FPGA sont basés sur la technologie CMOS, l'onduleur CMOS est le circuit CMOS le plus élémentaire, comme le montre la figure

I.9, il se compose d'un transistor de rappel à canal P et d'un transistor à faible drain à canal N, un seul transistor est passant à la fois, et parce que les transistors MOS ont une résistance aux chocs très élevée, les circuits CMOS ont généralement une très faible consommation d'électricité statique [3].

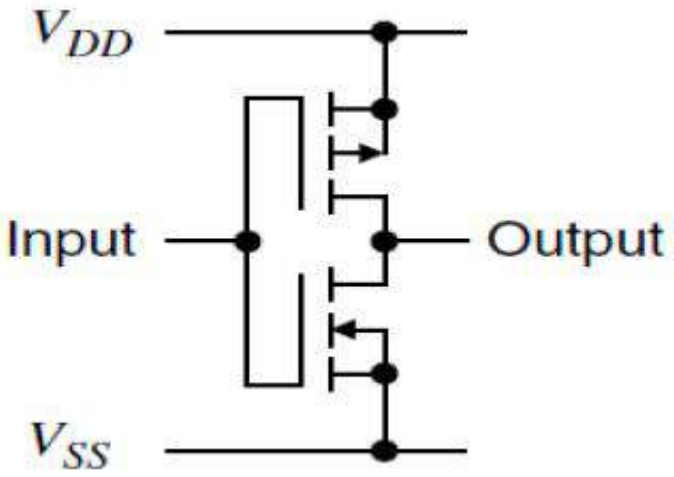


Figure I.9. *Inverseur CMOS [3]*

4. Différents domaines d'application des FPGA

Les FPGA ont fait révolutionner certains domaines de contrôle numérique et de plus en plus utilisés pour intégrer des architectures numériques complexes, ils sont devenus les plus populaires en matière d'implantation et de prototypage des circuits numériques après leur apparition sur le marché en 1984. La clé maîtresse de leurs réussites est l'aspect de programmation de ces derniers. Leurs utilisations actuelles couvrent les domaines civils et militaires. Parmi ces applications nous citons [7]:

1. Informatique : Périphériques spécialisés.
2. Machineries industrielles : Contrôleur pour machines.
3. Télécommunications : Traitement d'images, filtrage.
4. Instrumentations : équipement médical, prototypage.
5. Transports : Contrôle d'avions et métros.
6. Aérospatiales : Satellites.
7. Militaires : Radar, Communication protégée, la détection ou la surveillance…etc.

5. Technologies de programmation des FPGA

Pour franchir les inconvénients susmentionnés des mémoires, et dans le but de faire un ensemble de technologies complémentaires adaptable suivant l'environnement des cahiers de charges, il existe trois types d'FPGA reprogrammables suivant la technologie de mémorisation pour répondre aux différentes applications [5].

Ces trois principales technologies d'FPGA sont :

- Technologie de programmation par **RAM.**
- Technologie de programmation par **EEPROM ou FLASH.**
- Technologie de programmation par **ANTI-FUSIBLE**

5.1. *Technologie à base de RAM (XILINX et ALTERA)*

Cette technologie permet d'avoir une reconfiguration rapide des FPGA, les points de connexions sont des ensembles de transistors commandés, l'inconvénient majeur de cette technologie c'est qu'elle nécessite beaucoup de place et il est nécessaire de sauvegarder le design du FPGA dans une autre mémoire Flash.

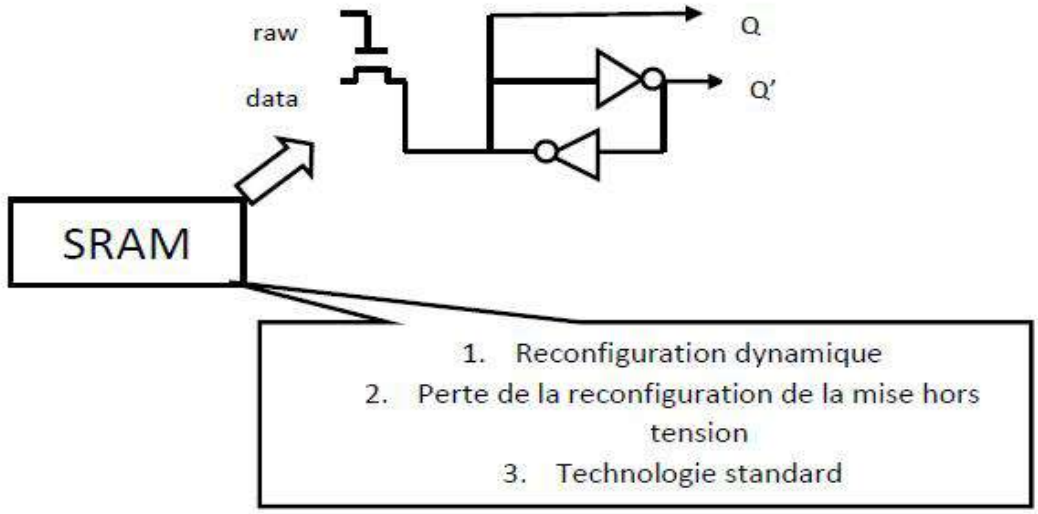


Figure I.10. *Caractéristiques de technologie SRAM [5]*

5.2. *Technologie à base d'EEPROM ou FLASH (LATTICE et ACTEL)*

Par rapport à SRAM, la technologie peut garder sa configuration inchangée, mais le nombre de configurations est limité et la vitesse de configuration est plus lente.

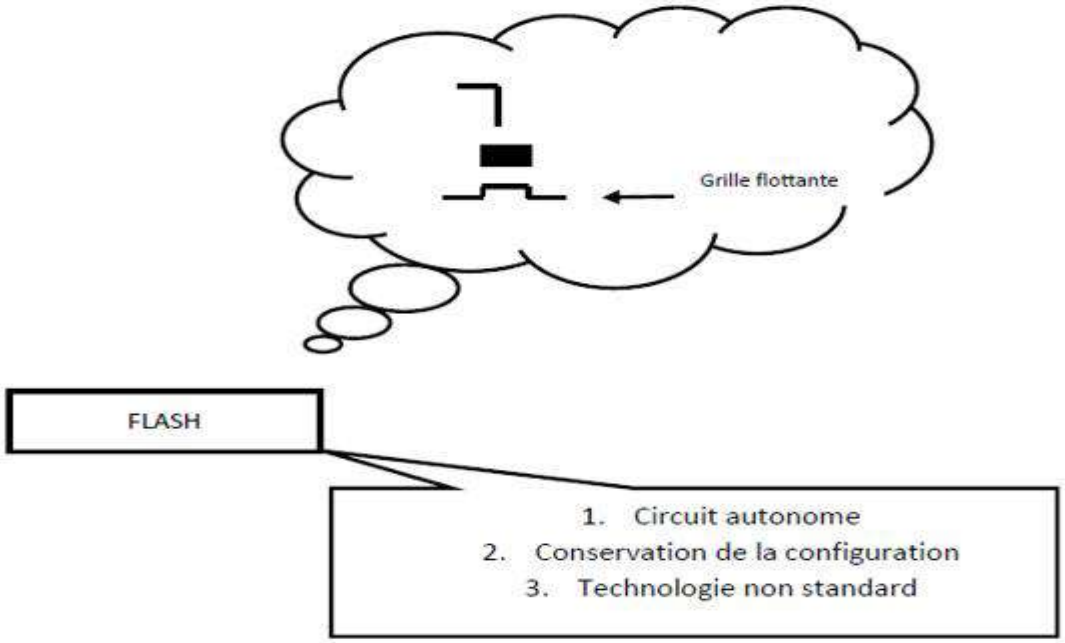


Figure I.11. *Caractéristiques de technologie FLASH [5]*

5.3. *Technologie à base d'ANTI-FUSIBES(ACTEL)*

Les points de connexions sont du type ROM, c'est-à-dire que la modification du point est irréversible. Pour comprendre le mécanisme de connexion sans rentrer dans les détails des semi-conducteurs, on considère que le point de connexion est le point de rencontre de deux segments conducteurs ou lignes conductrices, le nom anti-fusible vient du fait que l'état initial du fusible ou la couche isolante est présent et il n'y a pas de contact pour l'établir il faut détruire le fusible ce qui est contradictoire au fonctionnement habituel d'un fusible, des composants moins génériques mais plus petits et plus rapides ont été développés.

6. Etude des FPGA de la famille Xilinx

Le plus gros constructeur du marché est XILINX qui a introduit la série XC2000 (de 600 à 1500 portes) entre 1984 et 1985 avec des fréquences de fonctionnement pouvant atteindre les 420MHZ, depuis d'autres séries sont apparues comme les XC3000, XC4000, XC5200 et XC6200, ainsi que les XC9500 et la mise sur le marché du 1er FPGA XILINX en 1985 Puis

l'apparition en 1992 du premier FPGA du constructeur ALTERA qui est le concurrent le plus important de XILINX, avec un type de circuits assez différent le FLEX 8000 (15 000 portes max).

Rapidement l'exploitation de la technologie EEPROM un an plus tard en 1993 puis le lancement du VIRTEX II /XILINX (jusqu'à 10 millions de portes) en 2001 et des FPGAs de capacités supérieures à 50 millions de portes fonctionnant à des fréquences dépassant les 500 MHz en 2005. [8]

Le principe des FPGA de XILINX est de stocker la configuration dans une mémoire vive statique SRAM, aujourd'hui des blocs des fonctionnalités supplémentaires dans quelques versions évoluées sont ajoutés et dédiées à des applications spécifiques, ces fonctionnalités supplémentaires qui sont : Mémoire RAM, Petits multiplieurs, Blocs DSP, Cœurs de processeurs RISC et arbres de distribution d'horloge pour générer différent domaines d'horloge pour un bon synchronisme en cas d'FPGA à grande capacité.

Xilinx est le premier fournisseur mondial de technologies et d'appareils programmables, la société va au-delà de la logique programmable traditionnelle pour permettre la programmation matérielle et logicielle, intégrer des fonctions de signaux mixtes analogiques et numériques et permettre de nouveaux niveaux d'interconnexion programmable dans les circuits intégrés 3D monolithiques et multiples. Les produits Xilinx sont associés à un environnement de conception de nouvelle génération et à l'IP pour répondre à un large éventail de besoins des clients, de la logique programmable à l'intégration de systèmes programmables. [3]

7. Conclusion

Dans ce premier chapitre, nous avons abordé des définitions et quelques notions de base sur les circuits logiques reprogrammables ce qui nous permettra d'entamer le deuxième chapitre intitulé la reconfiguration partielle dans les circuits FPGA.

Chapitre II :

La reconfiguration partielle dans les circuits FPGAs

1. Introduction

La reconfiguration est un terme composé du préfixe « re » qui exprime la réitération (refaire). La reconfiguration est une réitération d'une configuration, par conséquent, c'est une nouvelle modification ou un nouveau réglage de paramètres qui permet l'optimisation du fonctionnement d'un système. Pour les FPGA la reconfiguration désigne le fait que l'architecture matérielle du système soit capable de s'adapter en changeant de forme ou de fonction. Un FPGA reconfigurable support le changement de sa configuration à n'importe quel moment. Cette reconfiguration peut être dynamique, car le processus de réitération permet d'avoir à chaque nouvelle configuration une nouvelle action. On peut donc définir les FPGA comme des systèmes reconfigurables et c'est leur exécution qui est dynamique. Tout système permettant une modification de son architecture grâce à l'anticipation et/ou à la réaction est auto-adaptatif. Et toute architecture matérielle permettant une modification de forme ou de fonction à tout moment est adaptable dynamiquement. Par conséquent, un système auto-adaptatif à base d'FPGA utilise la reconfiguration dynamique pour s'auto-adapter à son environnement. Pour réaliser cette adaptation et avoir un système flexible, il faut utiliser des techniques de conception et des architectures adéquates. Toutes ces architectures et ces techniques de conception sont en constante évolution pour améliorer la flexibilité des FPGA reconfigurables.

2. Configuration d'un FPGA

Pour configurer un FPGA, il faut un bitstream de configuration. C'est un fichier binaire dans lequel les informations de configuration pour un périphérique particulier sont stockées. Les bitstreams peuvent être partiels ou complets. Un bitstream complet (Merged bitstream) configure la mémoire de configuration de l'ensemble du dispositif. Il est utilisé pour la conception statique ou au début de l'exécution d'un système de reconfiguration dynamique afin de définir l'état initial des cellules SRAM. Le bitstream partiel configure une partie seulement du dispositif il est utilisé pour la reconfiguration partielle. Un FPGA fournit différents moyens d'interfaces de configuration, parmi eux il y a le port JTAG, l'interface SelectMAP pour la configuration de plusieurs FPGA, ou encore le port de configuration interne (ICAP) utilisé pour la reconfiguration. [9]

3. La reconfiguration des FPGA

Quelques années auparavant, les algorithmes qui pouvaient être mises en œuvre sur une puce FPGA unique été très peu répandue. En 1995, par exemple, les plus grands FPGA qui pouvaient être programmés disposaient d'environ 15 000 portes logiques au maximum. Plus récemment, les FPGA ont atteint des millions de portes logiques. Ensuite, les chercheurs et les concepteurs ont vu la possibilité de réaliser des systèmes auto adaptatifs grâce aux FPGA. Pour cela le circuit logique doit être capable de procéder à une modification de sa fonctionnalité en cours de fonctionnement. Les FPGA basiques ne le permettent pas, il est nécessaire d'arrêter le système pour y implémenter une nouvelle configuration. Les sociétés ATMEL et Xilinx à concevoir des architectures capables de procéder à une modification de leurs ressources sans arrêter le système. Cela donne naissance aux premières architectures reconfigurables dynamiquement. La capacité de l'architecture à s'adapter de manière dynamique à son environnement est caractérisée par le temps que met le système à se reconfigurer. Cette caractéristique est directement liée au temps d'accès de la mémoire SRAM de configuration. Le temps de reconfiguration dépend des technologies FPGA et des tâches matérielles utilisées, il peut varier de quelques microsecondes à quelques secondes. Les systèmes qui utilisent les architectures reconfigurables sont souvent utilisés pour réaliser des accélérateurs matériels pour traiter des tâches gourmandes en ressources et en temps de calcul.

La mise en œuvre de ce type de fonctionnement permet de reconfigurer autant de fois que nécessaire l'architecture matérielle du système tout au long de son exécution. Les tâches matérielles sont alors obtenues par un découpage de l'application sous forme de partitions exécutées séquentiellement. Les ressources sur lesquelles sont exécutées ces partitions sont le résultat du découpage spatial du circuit reconfigurable. La manière dont un algorithme sera implémenté sur une architecture reconfigurable dynamiquement va déterminer d'une part le niveau de performance dans le traitement des données et d'autre part le niveau de dynamicité dont cette architecture pourra faire preuve.

3.1. *Reconfiguration statique*

Le FPGA peut modifier ses fonctionnalités matérielles, pour cela il doit arrêter son exécution, une nouvelle configuration matérielle est chargée sur le FPGA (il est aussi possible de charger un nouveau programme logiciel sur le processeur) et il faut redémarrer le système. Dans ce cas, la reconfiguration est un procédé indépendant de l'exécution de l'application. Ce

procédé est appelé la reconfiguration statique, le contenu de la configuration reste statique pendant toute la durée de vie de l'application. La figure II.1 illustre le cycle de fonctionnement et de reconfiguration pour une architecture reconfigurable statiquement, il est à remarquer que les demandes d'arrêt et de reprise des opérations sont réalisées à la demande de l'utilisateur. La reconfiguration statique est similaire à l'utilisation d'un ASIC d'un point de vue de l'application finale et de son utilisation. La reconfiguration statique n'est pas capable de répondre aux besoins dans le domaine des systèmes adaptatifs. Concrètement, la reconfiguration statique est utilisée pour les tests, pour le prototypage, pour les systèmes avec application figée et pour l'initialisation d'un système. Parmi les systèmes utilisant la reconfiguration statique, nous trouvons principalement les accélérateurs matériels dont le rôle est de réaliser des tâches gourmandes en temps de calcul et qui nécessitent une exécution rapide qui ne peut pas être réalisée sur des systèmes à base de processeur. [9]

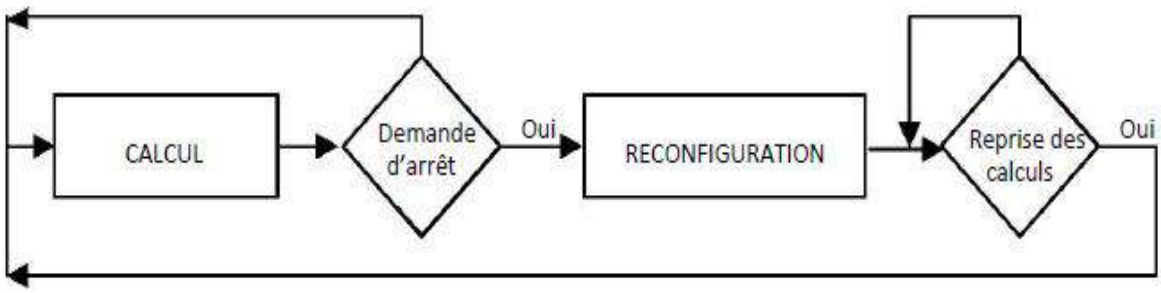


Figure II.1. *Mécanisme de la reconfiguration statique d'un FPGA [9]*

3.2. *Reconfiguration dynamique*

Une approche différente à la première est celle qui considère la reconfiguration du FPGA dans le cadre de l'application elle-même, en lui donnant la capacité d'adapter son matériel en configurant les ressources selon les besoins d'une situation particulière au cours d'exécution (figure II.2). Ce cas particulier est appelé la reconfiguration dynamique. Dans le cas de la reconfiguration dynamique, le processus de reconfiguration est considéré comme faisant partie de l'exécution de l'application. La reconfiguration dynamique est caractérisée par une application qui a un nombre de configurations au moins égales à deux. La mise en œuvre d'une telle application se traduit par une première configuration du système suivie par des reconfigurations qui permettront de modifier l'application autant de fois que nécessaire. Le système se comporte comme un processeur dont les ressources matérielles s'adaptent au cours du temps. La reconfiguration dynamique permet principalement d'accélérer des tâches pour lesquelles un processeur ne serait pas adapté. Un système reconfigurable dynamiquement doit

prendre en compte les temps de reconfiguration de celui-ci. Notamment pour les applications temps réel où les temps perdus pendant les phases de reconfigurations ne doivent pas impacter sur le temps de traitement du système.

Un des intérêts de la reconfiguration dynamique est la possibilité de modifier l'architecture matérielle de l'FPGA à l'infini pendant l'exécution d'une application comme le montre la figure II.2. De plus, la reconfiguration dynamique est un moyen d'améliorer le principal handicap des FPGA par rapport aux ASICs. Le principal handicap des FPGA est qu'ils sont beaucoup moins denses en nombres de portes logiques que les ASIC. Grâce à la reconfiguration dynamique, une petite surface logique peut supporter une grande application ou plusieurs petites applications. La reconfiguration dynamique peut être divisée en deux modes de fonctionnement : la reconfiguration dynamique globale ou la reconfiguration dynamique partielle. [9]

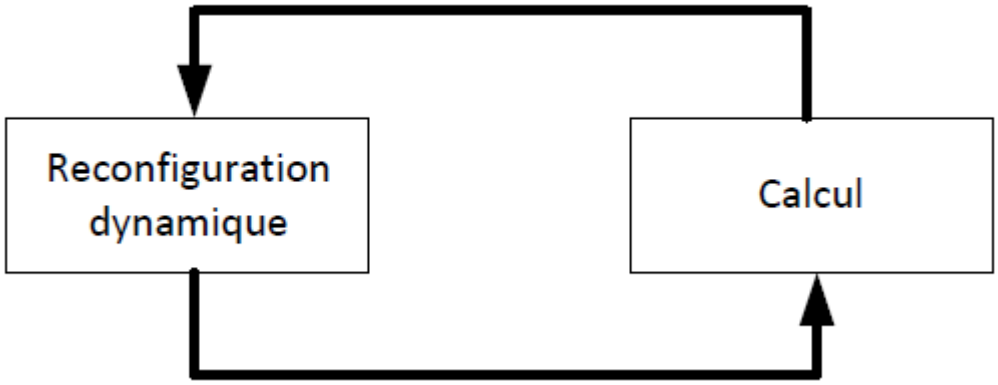


Figure II.2. *Mécanisme de la reconfiguration dynamique d'un FPGA [9]*

3.2.1. *Reconfiguration dynamique globale*

La reconfiguration dynamique globale permet de modifier la totalité du contenu de la mémoire de configuration. Dans ce type de reconfiguration, le FPGA est entièrement modifié pour accueillir une nouvelle implémentation. La figure II.3 illustre ce mode de fonctionnement qui consiste à reconfigurer la totalité du FPGA en passant d'une application à une autre. Sur cette figure, le FPGA est configuré avec une première architecture « A », ensuite il est entièrement reconfiguré pour passer à l'architecture « B ». La reconfiguration dynamique globale est une évolution de la reconfiguration statique. Cette évolution permet au FPGA de supporter une plus grande flexibilité et une bonne adaptation à son environnement. Actuellement avec les évolutions technologiques des FPGA nous pouvons implémenter de plus en plus d'applications dans un même circuit. Par conséquent, le problème de la taille des FPGA est de moins en moins contraignant, mais il reste toujours présent, car le coût en

ressources logiques d'un FPGA est important. [9]

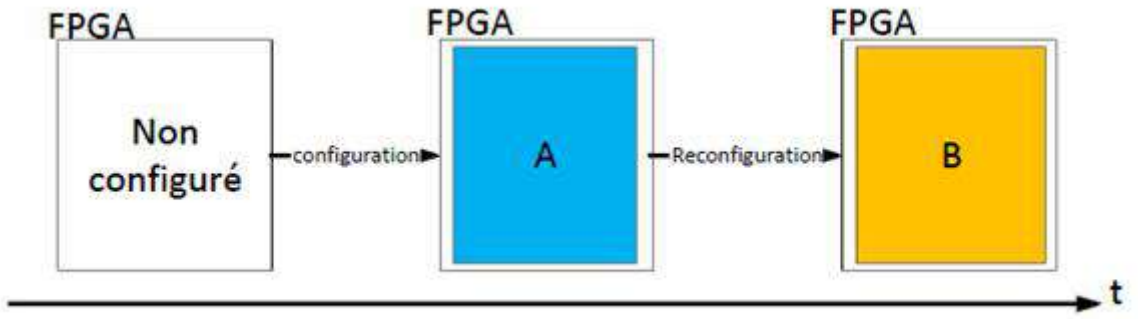


Figure II.3. *Reconfiguration dynamique globale d'un FPGA [9]*

3.2.2. *Reconfiguration dynamique partielle*

Le principal inconvénient de la reconfiguration dynamique globale est l'obligation de modifier la totalité du circuit même si certaines parties sont communes. Pour résoudre cet inconvénient, il existe une solution appelée la reconfiguration dynamique partielle. Elle permet de modifier une ou plusieurs parties du circuit sans interrompre le fonctionnement de celui-ci. La figure II.4 illustre ce mode de fonctionnement qui consiste à reconfigurer certaines régions. Dans un premier temps, le circuit FPGA est configuré avec une première configuration « A, A1, A2 ». Après une reconfiguration, la nouvelle configuration est « A, B1, B2 ». « A1 et A2 » ont respectivement été remplacés par « B1 et B2 » tandis que « A » n'est pas modifiée et est toujours en cours d'exécution. [9]

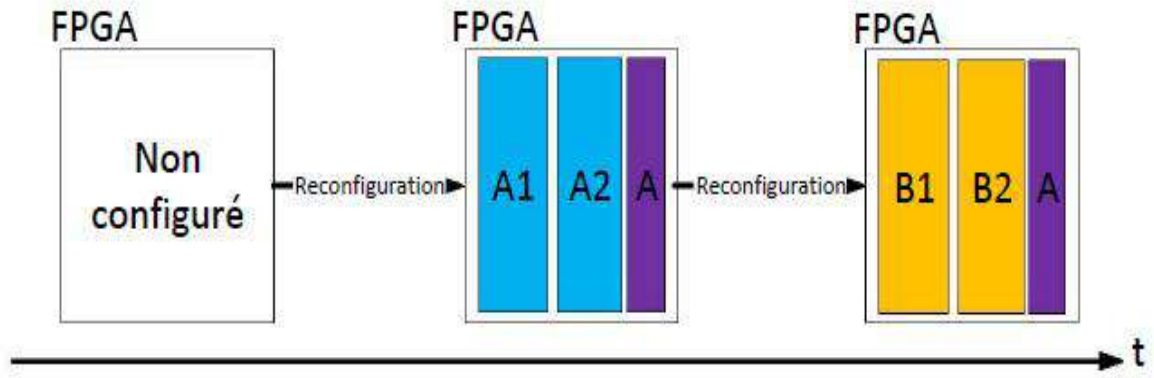


Figure II.4. *Reconfiguration dynamique partielle d'un FPGA [9]*

4. Processus de conception de la reconfiguration partielle

La figure II.5 montre la différence de contenu entre les bitstreams partiels et complets. Dans la configuration complète, un bitstream décrivant tout le FPGA est chargé même si la reconfiguration n'apporte que quelques modifications sur le système, un bitstream complet se compose d'un en-tête, de données de configuration et d'une somme de contrôle. Il s'agit de vérifier l'intégrité du bitstream et configurer la structure FPGA complète avec la conception logique. À la fin de la configuration, l'FPGA émet le signal "DONE" et entre en mode utilisateur à partir du mode de configuration où la conception commence à fonctionner, sauf si le bitstream a été corrompu, Cela garantit que les conceptions incorrectes ne commencent jamais à fonctionner sur le FPGA. [10]

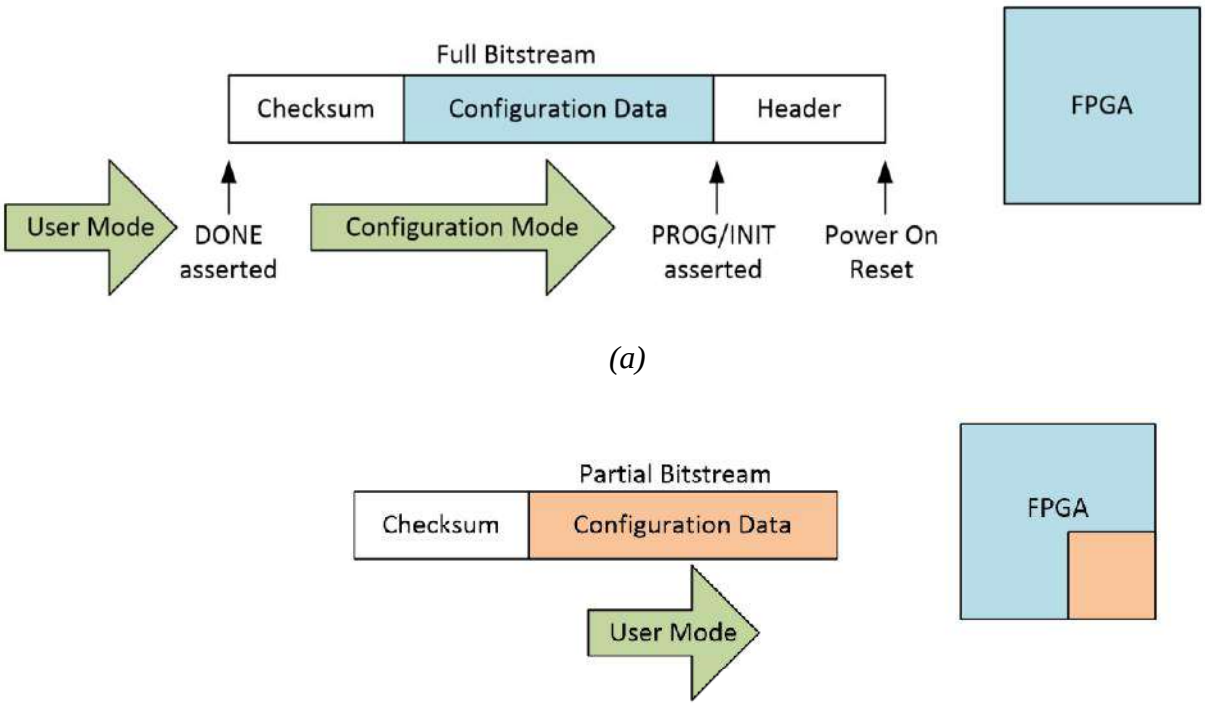


Figure II.5. *Différence entre (a) Configuration complète des bitstreams (b) Reconfiguration [10]*

Les bitstreams partiels, de l'autre côté, ne contiennent que les données de configuration et la somme de contrôle, contrairement à la méthode de configuration complète où l'FPGA est déjà en mode utilisateur avec une conception de fonctionnement lorsque le processus de reconfiguration a lieu, il n'est pas essentiel d'affirmer que le signal DONE ici. Le processus de reconfiguration dure jusqu'à ce que le bitstream partiel soit entièrement envoyé au port de configuration. Les défaillances des sommes de contrôle indiquent des bitstreams corrompus et

en cas des bitstreams partiels, ils sont plus préoccupants car si l'échec de la somme de contrôle est dû à des données de configuration erronées, la conception incorrecte est isolée dans la partition reconfigurable, cela peut être corrigé au moyen du chargement d'un autre bitstream partiel pour la partition respective, par contre, si l'échec est dû à une adresse de bloc erronée de la partition reconfigurable, alors la logique statique peut être corrompue à la suite de la reconfiguration. Pour éviter ces problèmes, il est conseillé de vérifier leur intégrité avant de les diriger vers les ports de configuration. [10]

Les ports de configuration sont responsables à la configuration de la structure FPGA après avoir vérifié l'intégrité des bitstreams, généralement un processeur ou une machine d'état est utilisé pour récupérer les bitstreams partiels de la mémoire non volatile (carte SD, mémoire QSPI, etc.) et les diriger à des ports de configuration.

Ainsi, le fait de disposer d'un système de traitement (PS) sur le FPGA SoC peut présenter l'avantage de transférer un bitstream partiel pour configurer la partition reconfigurable (RP), sans qu'il soit nécessaire d'ajouter un processeur sur puce supplémentaire sur le matériel. Différents ports de configuration peuvent être utilisés pour effectuer la reconfiguration partielle d'un FPGA. [10]

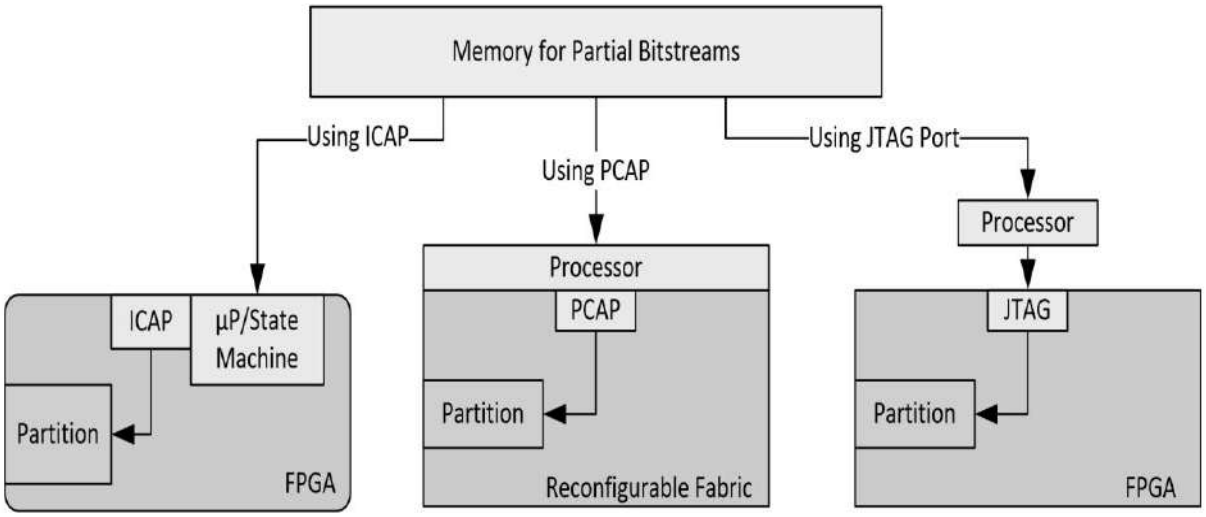


Figure II.6. *Méthodes de configuration pour la conception des RP (a) ICAP (b) [10]*

Les trois méthodes de reconfiguration les plus couramment utilisées sont :

- Port d'accès à la configuration interne (ICAP): Ce port permet une reconfiguration partielle au sein du FPGA, permettant ainsi des conceptions de FPGA à configuration automatique.

-Port de configuration du processeur (PCAP) : Cette interface de configuration est utilisée par les architectures reconfigurables qui intègrent un noyau de processeur dur, le processeur configure l'architecture reconfigurable à l'aide de ce port. L'ICAP et le PCAP sont des méthodes de configuration interne pour charger le bitsream partiel.

- Port JTAG : il s'agit d'une interface permettant de tester rapidement la reconfiguration partielle en utilisant la méthode du port externe, des processeurs peuvent être utilisés pour extraire les bitstreams de la mémoire et en les dirigeants vers le port du JTAG. Des outils performants sont disponibles à cet effet, la figure II.6 montre l'utilisation des trois ports de configuration mentionnés.

5. Les avantages de la reconfiguration partielle

La mise en œuvre d'une conception basée sur RP ouvre de nouveaux horizons une utilisation flexible dans une grande variété d'applications. Les avantages d'utilisation de conceptions basées sur une reconfiguration partielle sont les suivants :

- Réduction de la consommation de ressources et d'énergie :

Avec la reconfiguration partielle, il est possible de multiplexer dans le temps plusieurs modules matériels, cela réduit le besoin total de ressources pour la mise en œuvre de toute conception matérielle, ce qui se traduit directement par une réduction de la consommation d'énergie et des coûts économies.

Pour les systèmes utilisant plusieurs FPGA, une reconfiguration partielle permet d'intégrer la conception dans un nombre inférieur de circuits intégrés du FPGA. Pour ces systèmes, des économies d'énergie peuvent être obtenues non seulement grâce à la réduction du nombre de FPGA, mais aussi grâce à la réduction de la communication sur la puce. [10]

- Amélioration des performances et flexibilité :

Avec la reconfiguration partielle, la capacité de calcul du système peut être adaptée au moment de l'exécution, par exemple dans les systèmes où des noyaux matériels différents sont utilisés l'un après l'autre, les ressources supplémentaires peuvent être utilisées pour accélérer le fonctionnement du noyau ou pour créer un plus grand nombre de noyaux pour effectuer l'opération en parallèle.

- Amélioration de la tolérance aux pannes et de la fiabilité :

La tolérance aux pannes est un critère très important pour les systèmes de sécurité critiques, en particulier dans les industries aérospatiales et de défense. Les pannes de système dues à des défauts matériels peuvent être fatales dans ces systèmes, avec la reconfiguration partielle, la tolérance aux pannes et la fiabilité des systèmes peuvent être améliorées au moyen de techniques telles que la diversification des modules et l'épuration de la configuration. [10]

- Conceptions de matériel auto-adaptatif :

Avec des architectures matérielles de la reconfiguration partielle on peut s'adapter à l'évolution des conditions d'exploitation et d'environnement si nécessaire en fonction sur l'intelligence artificielle et l'apprentissage.

6. conclusion

La reconfiguration dynamique partielle est un mécanisme puissant qui permet de reconfigurer des portions du FPGA sans affecter le fonctionnement du reste du système, ce mécanisme offre une grande flexibilité qui permet d'adapter que la partie concernée du système suite à des changements durant l'exécution qui peuvent être liés à l'environnement, aux entrées des utilisateurs ou des contraintes internes du système (performance, consommation, contraintes liées aux données, etc.). La prise en compte de ces différentes sources d'adaptation pourrait engendrer une grande complexité de conception pour des systèmes de grande taille.

L'objectif de ce Chapitre est de donner une explication détaillée concernant la reconfiguration partielle en général et la reconfiguration dynamique partielle en particulier.

Dans le chapitre suivant, des travaux pratiques sur la reconfiguration dynamique partielle seront réalisés en se basant sur l'une des méthodes abordées dans ce chapitre.

Chapitre III :

Implémentation de la commande d’un servomoteur sur FPGA en utilisant la reconfiguration dynamique partielle.

1. Introduction

Les systèmes reconfigurables permettent d'implémenter plusieurs fonctions sur une même région. Grâce à la reconfiguration dynamique partielle, on peut substituer certaines fonctions (tâches matérielles ou IP) du dispositif FPGA par de nouvelles fonctions à leur place. La question de la communication est l'un des défis dans la mise en place des tâches matérielles sur les régions reconfigurables.

Le but de ce travail est de concevoir un système matériel/logiciel pour tester la reconfiguration dynamique partielle sur le circuit FPGA. Notre objectif est de commander un servomoteur via une carte FPGA en utilisant la reconfiguration dynamique partielle. Le projet est composé de deux partie une partie statique et une partie dynamique.

2. Présentation du projet

Dans ce travail, la reconfiguration dynamique partielle est utilisée pour changer dynamiquement les configurations et les fonctions des boutons poussoirs de la carte Nexys4 DDR pour commander l'intervalle et la vitesse de rotation d'un servomoteur.

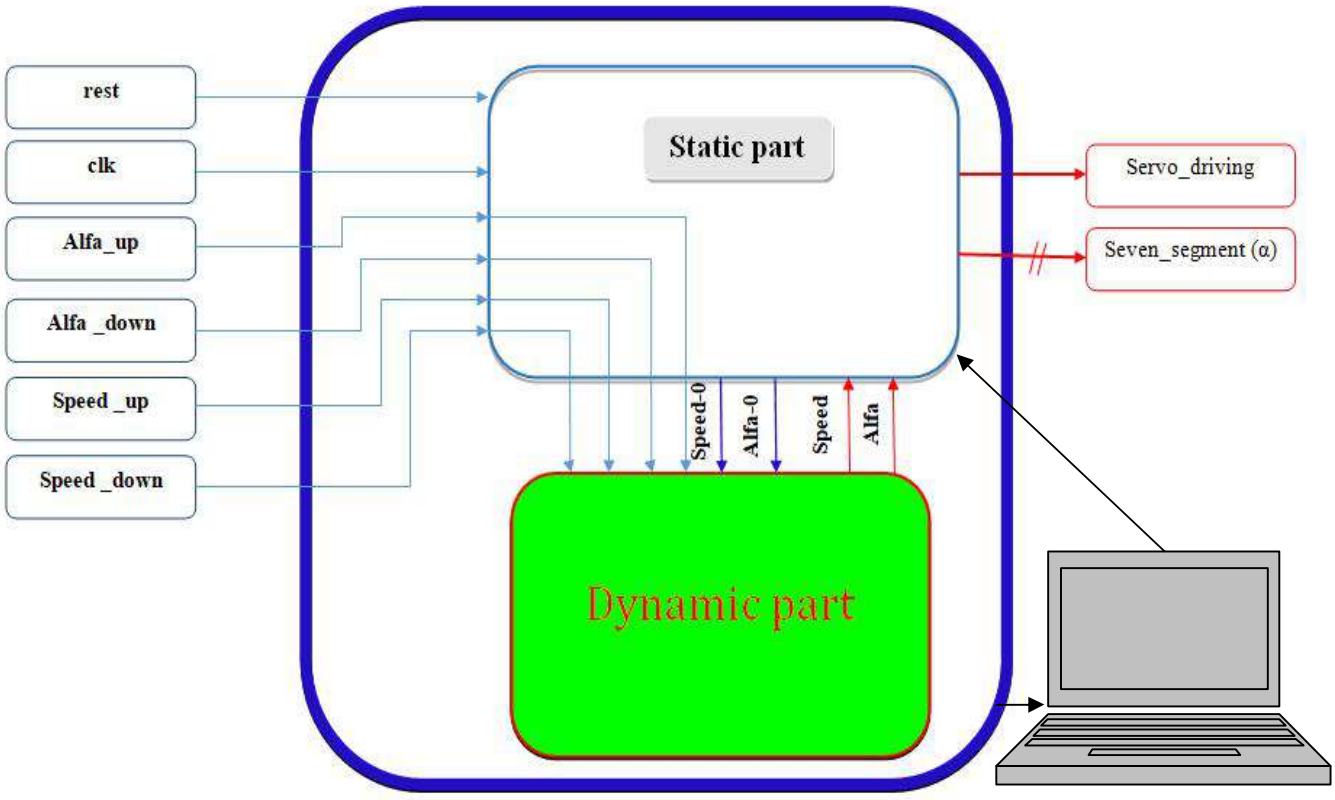


Figure III.1. *La commande d'un Servomoteur A Base d'FPGA*

Le changement de fonctions des boutons poussoirs doit se fait sans interrompre le fonctionnement du servomoteur. Une partie de la conception matérielle est mise en œuvre sous forme de partition reconfigurable. La logique matérielle de chaque configuration de boutons poussoirs est stockée sous forme des bitstreams partiels de modules reconfigurables. La reconfiguration dynamique partielle, nous donne la possibilité d'ajouter des fonctionnalités supplémentaires en ajoutant de nouveaux modules reconfigurables.

2.1. *Principe du fonctionnement*

La figure (Figure III.2) illustre le principe du fonctionnement du servomoteur, une plaque ronde graduée avec une aiguille ont placées sur un servomoteur de type SG-90 pour une visibilité claire et pour un bon contrôle du champ de rotation du servomoteur.

Le servomoteur doit pouvoir faire tourner son arbre de +Alfa à -Alfa avec une vitesse contrôlée. Pour contrôler la vitesse du servomoteur, nous devons l'entraîner vers des positions intermédiaires (step by step) avec un temps de retard, donc la variation de sa vitesse est la variation de la valeur de **Step**.

- L'aiguillé du servomoteur déplace de **–Alfa** a **+Alfa** et vice versa. L'angle instantané est **Beta.**
- L'angle **Beta** varie de **–Alfa** a **+Alfa** avec une vitesse de **1 Step/500ms.**
- Les entrées **Alfa_up** et **Alfa_down** nous permet d'augmenter ou de diminuer la valeur d'**Alfa de 0** à **90°**.
- Les entrées **Speed_up** et **Speed_down** nous permet d'augmenter ou de diminuer la valeur de **Step de 0** à **90°**, ce qui nous permet de contrôler la vitesse du servomoteur (Figure III.3).

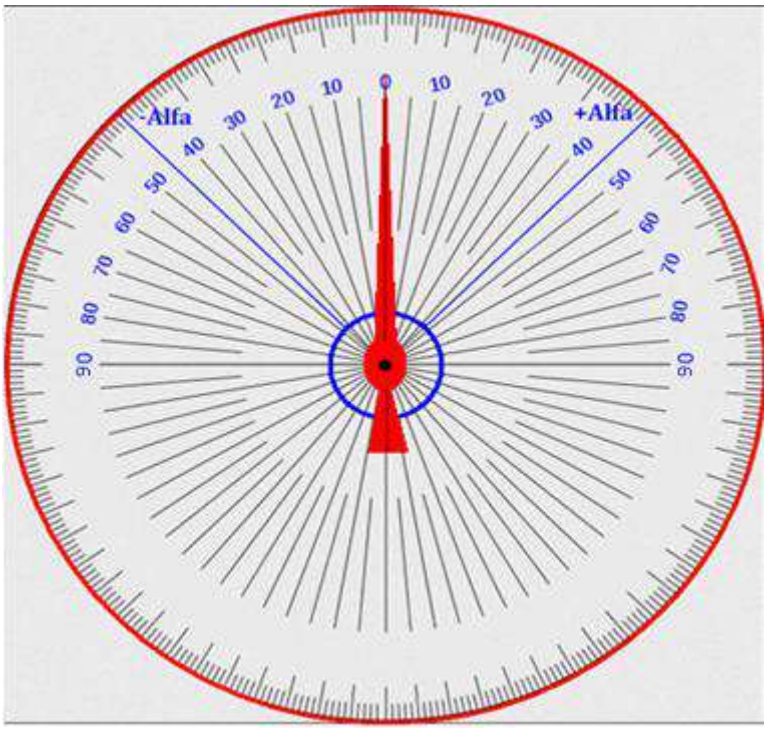


Figure III.2. *Plage de Fonctionnement du servomoteur*

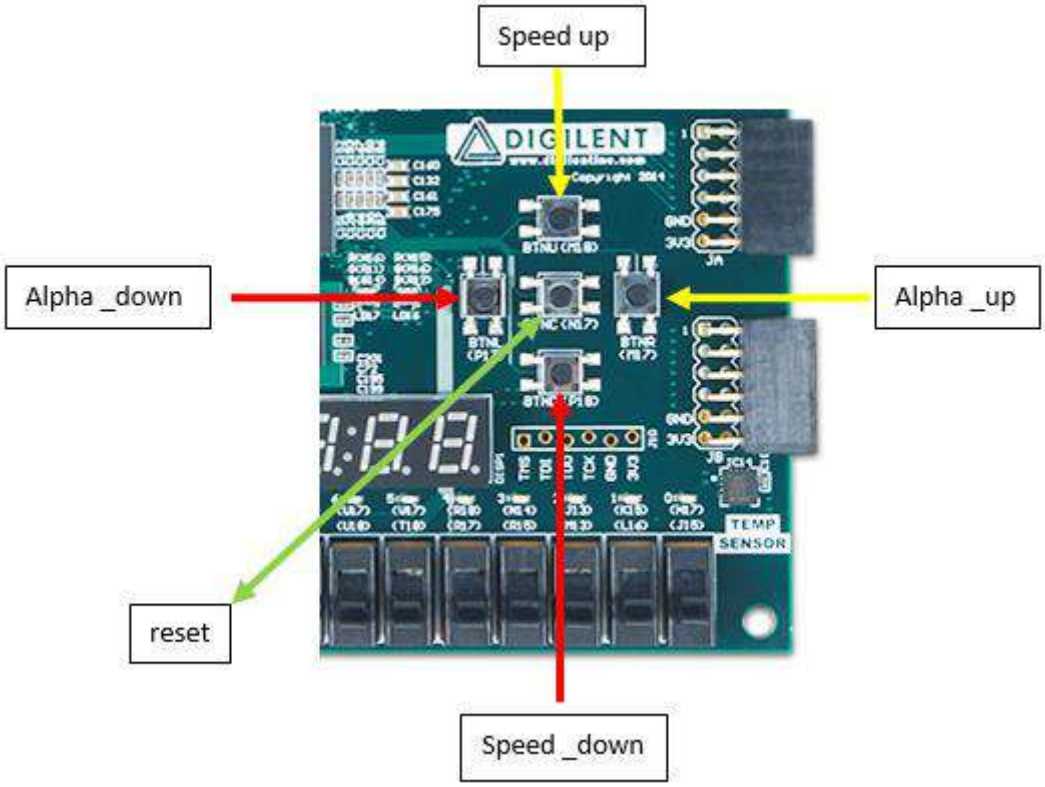


Figure III.3. *Le fonctionnement des boutons poussoirs*

2.2. *L'affichage à sept segments*

Comme la carte Nexys4 DDR est composée de deux écrans LED à quatre chiffres on a préféré de les divise à quatre parties, comme indique la figure (Figure III.4):

- La partie (1) est utilisée pour afficher l'angle alfa.
- La partie (2) est utilisée pour afficher la vitesse de l'arbre du servomoteur.
- La partie (3) est utilisée pour afficher le signe de l'angle instantané.
- La partie (4) est utilisée pour afficher l'angle instantané.

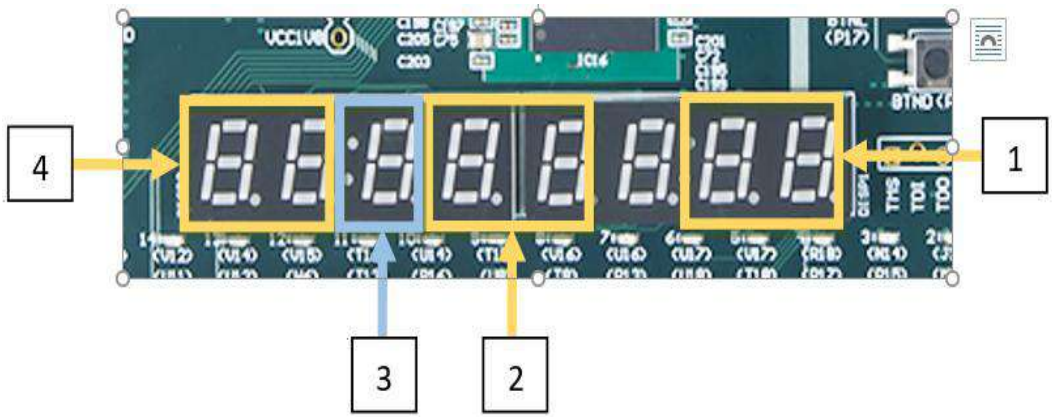


Figure III.4. *L'affichage à sept segments*

3. Composants matériels

3.1. *La carte Nexys4 DDR*

La carte Nexys4 DDR est une plate-forme de développement de circuits numériques complète et prête à l'emploi, basée sur le dernier Field Programmable Gate Array (FPGA) Artix-7 ™ de Xilinx®. Avec son grand capacité FPGA (numéro de pièce Xilinx XC7A100T-1CSG324C), ses mémoires externes généreuses et sa collection de ports USB, Ethernet et autres, le Nexys4 DDR peut héberger des conceptions allant des circuits combinatoires d'introduction aux processeurs embarqués puissants. Plusieurs périphériques intégrés, dont un accéléromètre, un capteur de température, un microphone numérique MEMs, un amplificateur de haut-parleur et plusieurs périphériques d'E / S permettent au Nexys4 DDR d'être utilisé dans une large gamme de conceptions sans nécessiter d'autres composants. [11]

Les Caractéristiques principales de la carte Nexys4 DDR sont :

- Circuit FPGA : Artix-7
  - 15 850 Slices de logique programmable, chacune avec quatre LUT à 6 entrées et 8 bascules (8 150 slices)
  - 4,860 Kbits de RAM de bloc rapide
  - Six tuiles de gestion d'horloge, chacune avec une boucle à verrouillage de phase (PLL)
  - 240 slices DSP
  - Vitesse d'horloge interne supérieure à 450 MHz
  - Convertisseur analogique-numérique interne (XADC) à deux canaux, 1 MSPS
- Mémoire
  - 128 Mo DDR2
  - Flash série
  - Fente pour carte microSD
- Puissance
  - Alimenté par USB ou par n'importe quelle source d'alimentation externe 4.5V-5.5V
- Entrée / sortie utilisateur simple
  - 16 commutateurs
  - 16 LED
  - Deux LED RVB
  - Deux afficheurs à 7 segments à 4 chiffres
- Connecteurs d'expansion
  - Quatre connecteurs Pmod fournissant un total de 32 entrées / sorties FPGA.

Le tableau III.1 et la Figure III.5 représentent les composants de la carte Nexys4 DDR.

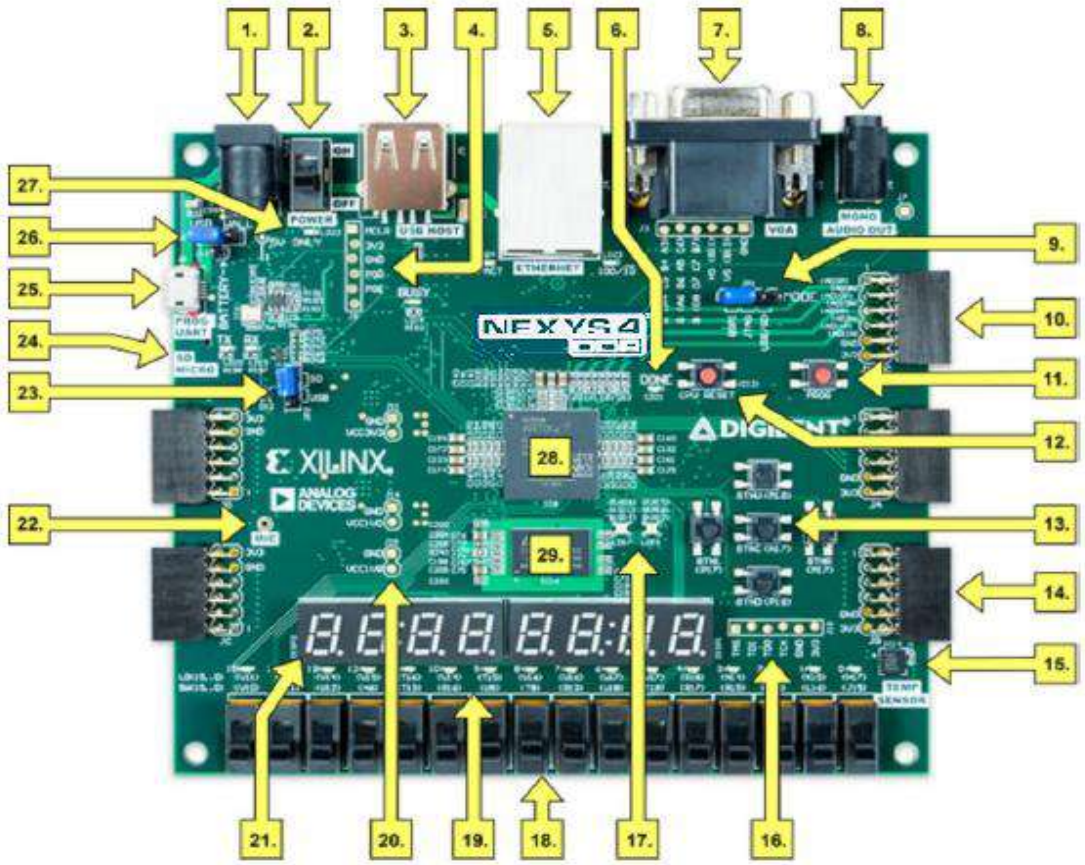


Figure III.5. *Légende de la fonction Nexys4 DDR*

Tableau III.1. *Les composants de la carte Nexys4 DDR*

Composant	Description de composant	Composant	Description de composant
1	Prise d'alimentation	16	Port JTAG pour câble externe (en option)
2	Interrupteur d'alimentation	17	LED tricolores (RVB)
3	Connecteur hôte USB	18	Commutateurs à glissière (16)
4	Port de programmation PIC24	19	LED (16)
5	Connecteur Ethernet	20	Point (s) de test d'alimentation
6	Programmation FPGA terminée LED	21	Affichage à sept segments de huit chiffres

7	Connecteur VGA	22	Microphone
8	Connecteur audio	23	Cavalier de configuration externe (SD / USB)
9	Cavalier de mode de programmation	24	Logement de carte MicroSD
10	Port Pmod de signal analogique (XADC)	25	Port USB partagé UART / JTAG
11	Bouton de réinitialisation de la configuration FPGA	26	Cavalier de sélection de l'alimentation et tête de la batterie
12	Bouton de réinitialisation de CPU (pour les noyaux souples)	27	Voyant Power-Good
13	Cinq boutons poussoir	28	FPGA Xilinx Artix-7
14	Port (s) Pmod	29	Mémoire DDR2
15	Capteur de température		

3.2. *Le servomoteur SG-90*

Dans ce projet, nous avons utilisé le servomoteur SG-90 qui est petit et léger avec une puissance de sortie élevée. Le servo peut tourner d'environ 180 degrés (90 dans chaque sens) et fonctionne comme les servo classiques [12].

Le temps, ou la fréquence, de l'impulsion détermine la position du servo. Chaque servo a sa propre gamme de fréquences, indiquée par le fabricant dans la fiche technique. La figure III.6 montre des valeurs comprises entre 1 et 2 ms.

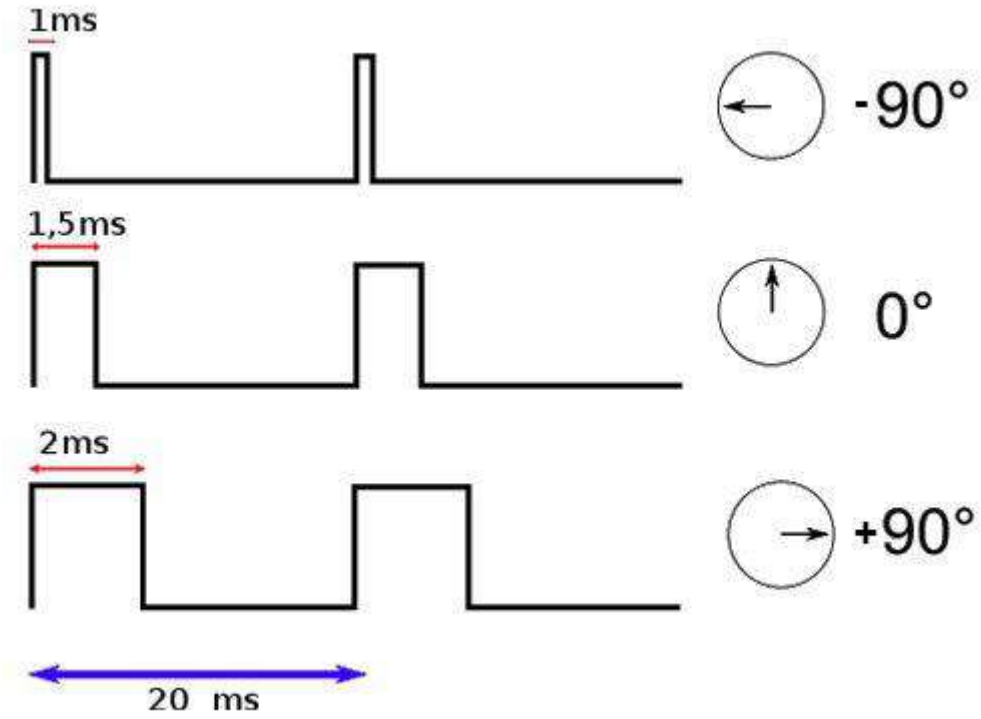


Figure III.6. *Fonctionnement du servomoteur*

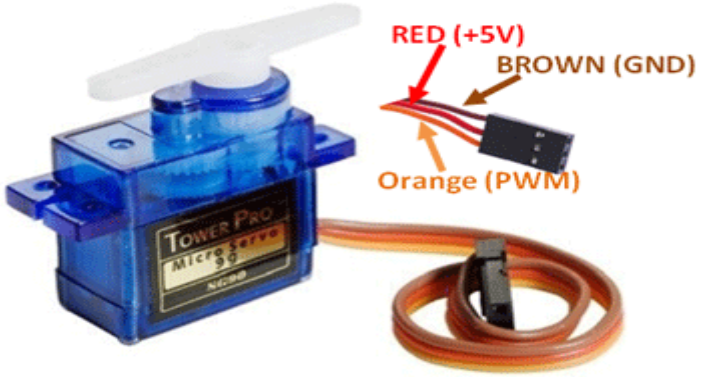


Figure III.7. *Servomoteur*

Tableau* III.*2. *Description des fils de connexion du servomoteur*

Numéro de fil	Couleur du fil	Description
1	Marron	La masse
2	Rouge	Le moteur est généralement alimenté en +5V
3	Orange	Le signal PWM est transmis par ce fil pour commander le moteur

3.3. *Organisation des programmes VHDL*

Le code VHDL a été organisé en quatre fichiers VHDL. Le programme principale (TOP.vhd) présente la partie statique de ce projet, il est chargé de générer les signaux de commandes pour contrôler la vitesse et la position de l'axe du servomoteur, de gérer les afficheurs sept segments et de communiquer avec la partie reconfigurable qui lit les entrées venantes des boutons poussoirs de la carte est qui augment ou diminue les valeurs consignes de la vitesse et l'angle de balayage du l'axe du servomoteur. La partie reconfigurable et chargée par défaut par le bitstream partiel qui correspond au programme RM.vhd et peut être reconfiguré ultérieurement par les bitstreams partiels qui correspondent aux programme RM2.vhd ou RM3.vhd. Le projet est finalement composé de quatre bitstreams représentent une partie statique et trois autres modules pour la partie reconfigurable. Chaque module reconfigurable (RM) a son propre fonctionnalité.

3.3.a. *La partie statique*

Le premier organigramme Figure III.8 représente la partie statique qu'on va implémenter sur la carte, et qui explique comment génère les signaux et commander le servomoteur selon les valeurs consignes actualisées chaque fois par la partie dynamique.

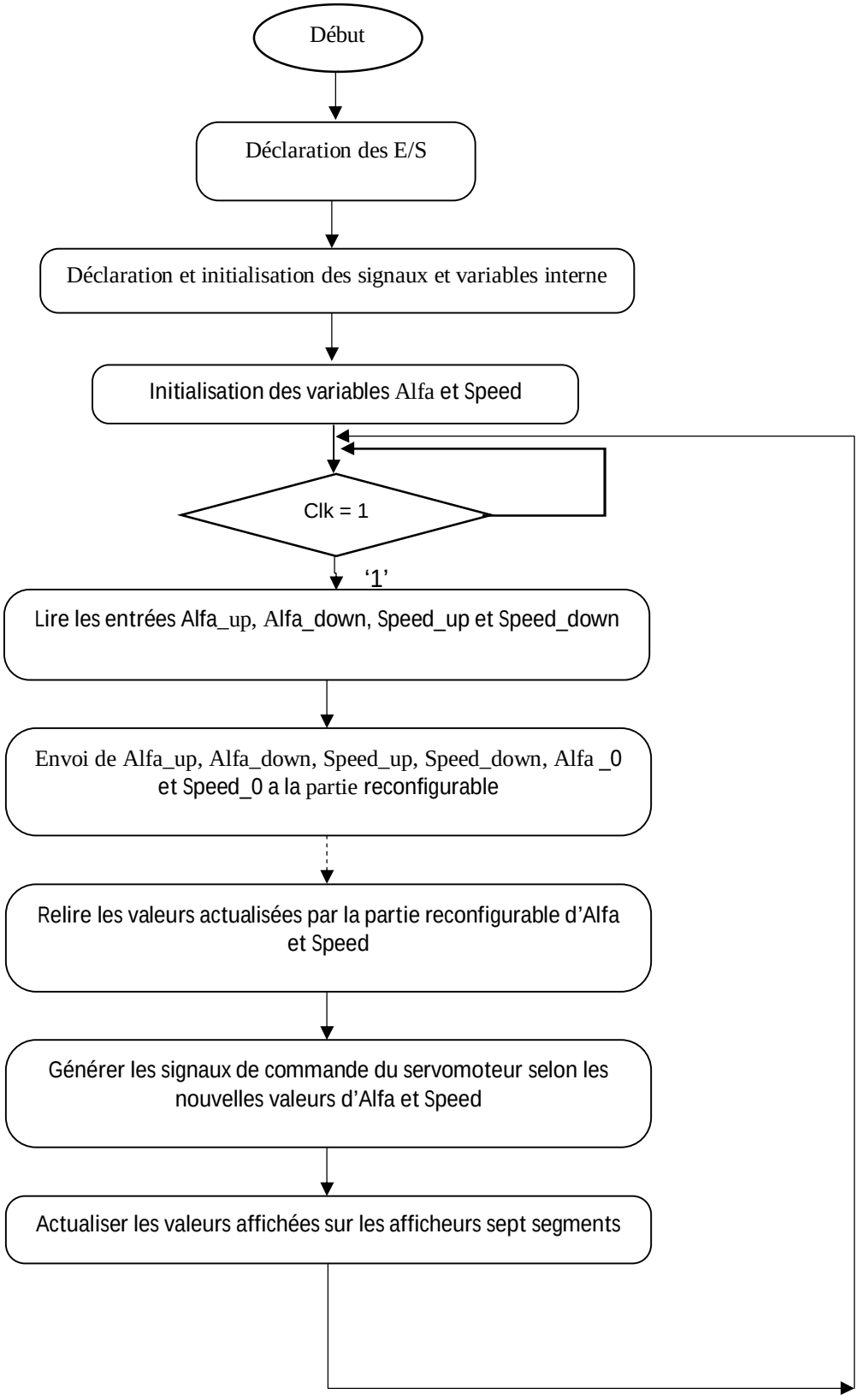


Figure III.8. *Organigramme simplifie de la partie statique*

3.3.b. *La partie reconfigurable*

Le deuxième organigramme Figure III.9 représente le deuxième programme qui explique les le fonctionnement du premier module reconfigurable (RM1) et qu'on va l'implémenter sur la partie reconfigurable de l'architecture sur le circuit FPGA.

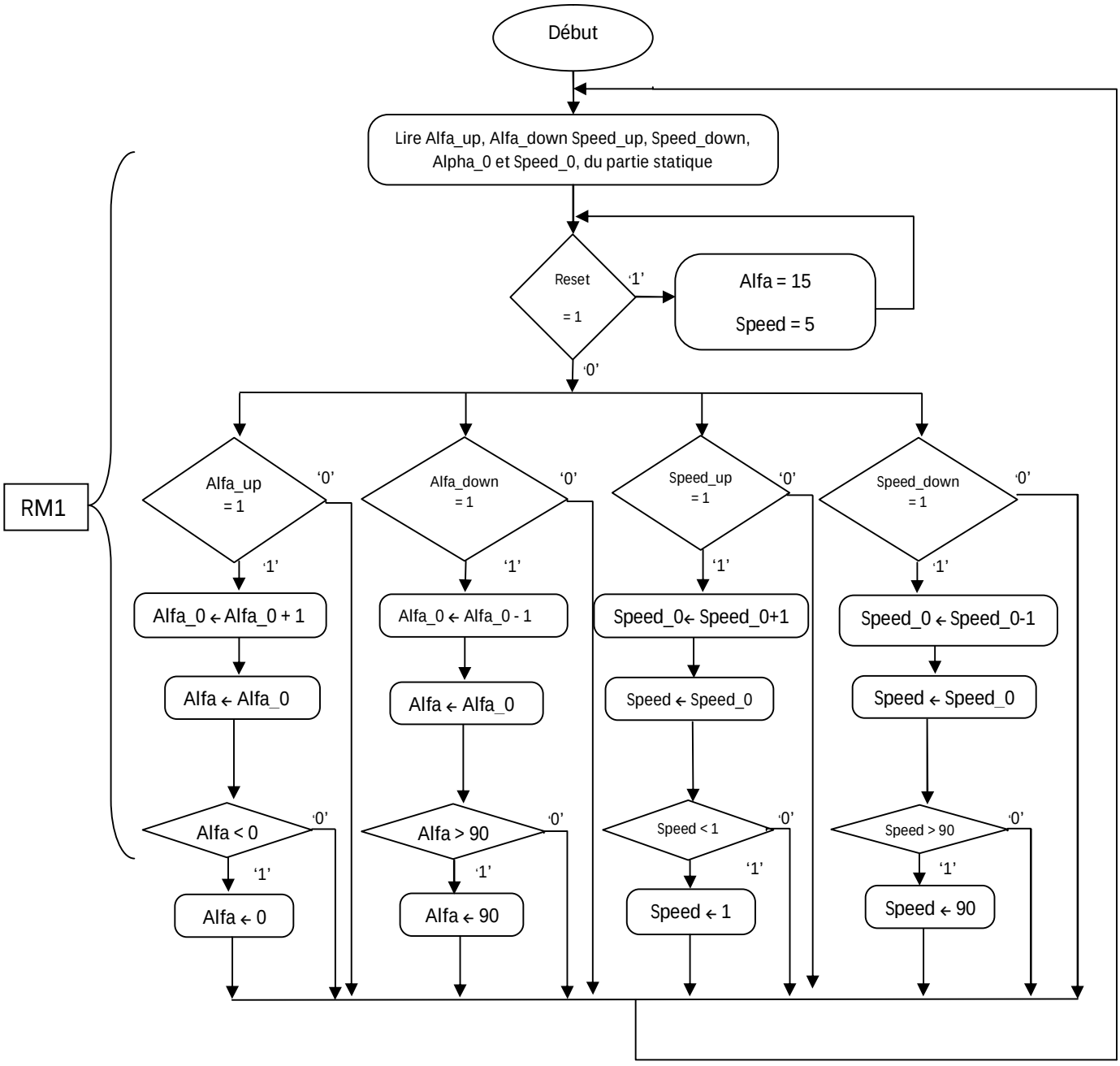


Figure III.9. *Organigramme de RM1*

Le troisième organigramme Figure III.10 représente le troisième programme qu'on va implémenter sur la carte, qui explique les étapes de fonctionnement du deuxième module reconfigurable (RM2) qui représente la partie dynamique (changeable) de l'architecture.

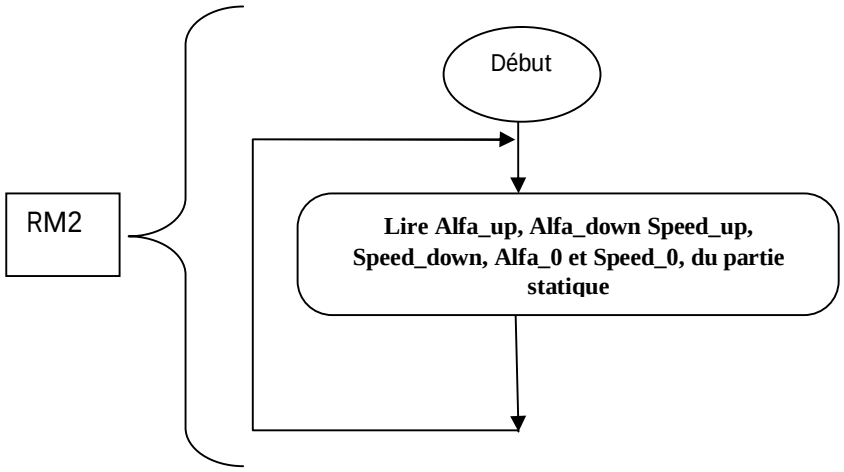


Figure III.10. *Organigramme de RM2*

Le quatrième organigramme Figure III.11 représente le quatrième bitstream qu'on va implémenter sur la carte, qui explique les étapes de fonctionnement du troisième module reconfigurable (RM3) qui représente la partie dynamique (changeable) de l'architecture.

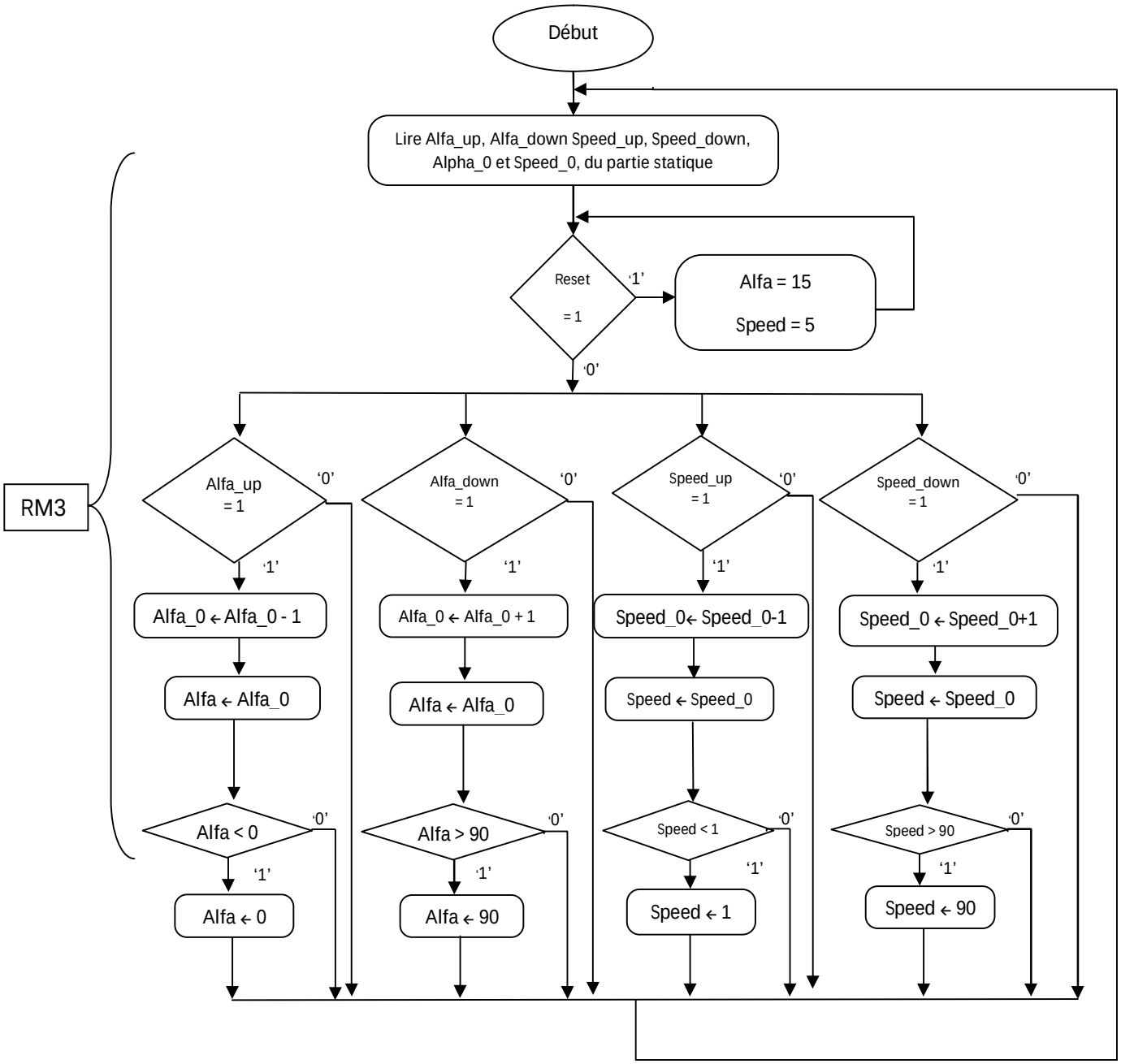


Figure III.11. *Organigramme de RM3*

4. L'implémentation de l'architecture

4.1. *Vivado Design Suite*

Le logiciel Vivado Design Suite, qu'on a utilisé pour le développement de ce projet, est une suite logicielle produite par Xilinx pour la synthèse et l'analyse de conceptions HDL,

remplaçant Xilinx ISE avec des fonctionnalités supplémentaires pour le développement de système sur puce et la synthèse de haut niveau. Vivado représente une réécriture et une refonte complète du flux de conception (comparé à ISE) et a été décrit par les critiques comme "bien conçu, étroitement intégré, rapide, évolutif, facile à gérer et intuitif ".

Contrairement à son prédécesseur ISE qui utilisait ModelSim pour la simulation, Vivado System Edition inclut un simulateur logique intégré. Vivado introduit également la synthèse de haut niveau, avec une chaîne d'outils qui convertit le code C en logique programmable. Vivado a été décrit comme un « outil EDA complet et à la pointe de la technologie, doté de toutes les dernières nouveautés en termes de modèle de données, d'intégration, d'algorithmes et de performances ». [13]

Le logiciel Vivado Design Suite supporte la technologie de reconfiguration dynamique partielle qui permet de reconfigurer des parties d'une conception matérielle après qu'elle ait été mise en œuvre sur une structure FPGA.

Une conception qui prend en charge la reconfiguration dynamique partielle du matérielle aura une ou plusieurs partitions reconfigurables dans la conception. Chaque partition reconfigurable sera associée à un ou plusieurs modules reconfigurables. Le reste de la conception connue sous le nom de logique fixe reste inchangé (figure III.12).

Les modules reconfigurables doivent être créés à l'avance et stockés sous forme de bitstream partiel. Les bitstreams partiels sont utilisés pour reconfigurer une partition reconfigurable selon les besoins. Les modules reconfigurables associés à une partition reconfigurable.

Peuvent être utilisés pour la reconfigurer ou modifier ses caractéristiques fonctionnelles en cours d'exécution.

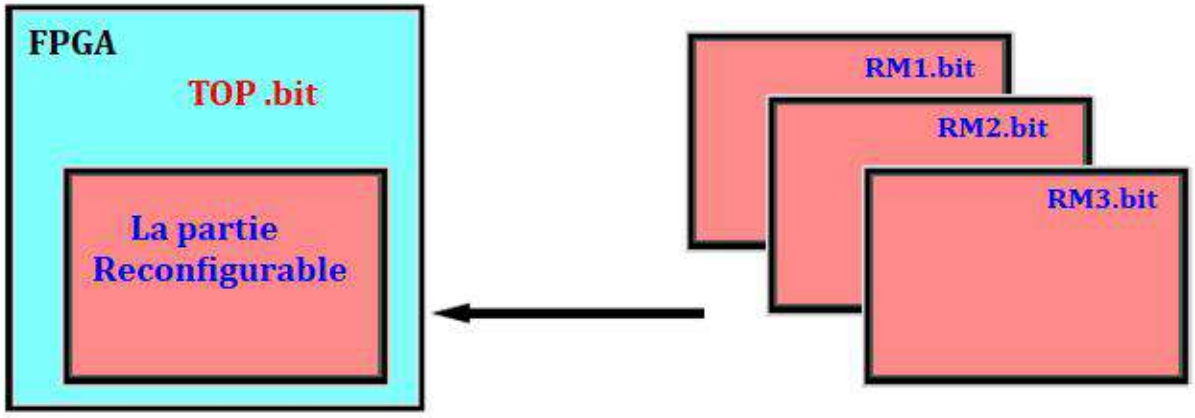


Figure III.12. *Les bitstreams et la reconfiguration dynamique partielle de notre projet*

La reconfiguration dynamique partielle du matérielle peut réduire la taille et le coût d'une conception. Elle offre également la possibilité d'ajouter de nouvelles fonctionnalités ou caractéristiques aux modèles déjà existants qui ont une partition reconfigurable. Les performances et l'efficacité des conceptions peuvent également être améliorées à l'aide de la technologie de reconfiguration dynamique partielle.

4.2. *Création du projet*

Comme départ on a créé un nouveau projet puis on a ajouté le fichier vhdl sources et les contraintes de conception, et défini tous les détails du projet principal (partie statique).

Une fois le projet créé, on doit défini comme un projet de Reconfiguration Partielle. Pour ce faire, on sélectionne Outils → Activer la reconfiguration partielle. Cela prépare le projet au processus de conception des RP. Une fois que cette option est définie, elle ne peut pas être annulée.

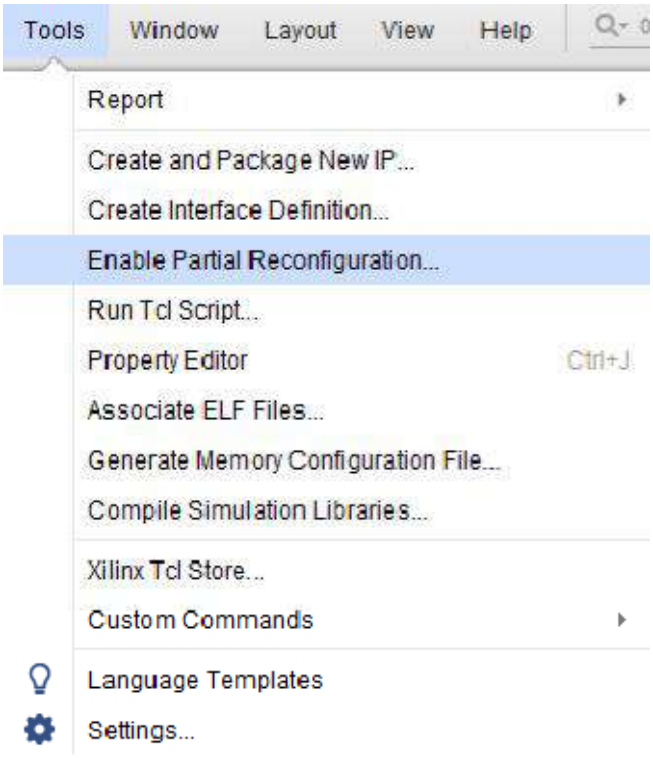


Figure III.13. *Activation de la reconfiguration partielle*

4.2.a *Définition des partitions reconfigurables*

Les définitions de partition ont été créé pour lancer le processus de création de partitions reconfigurables. Ce module peut être une boîte noire si les sources de conception n'existent pas encore.

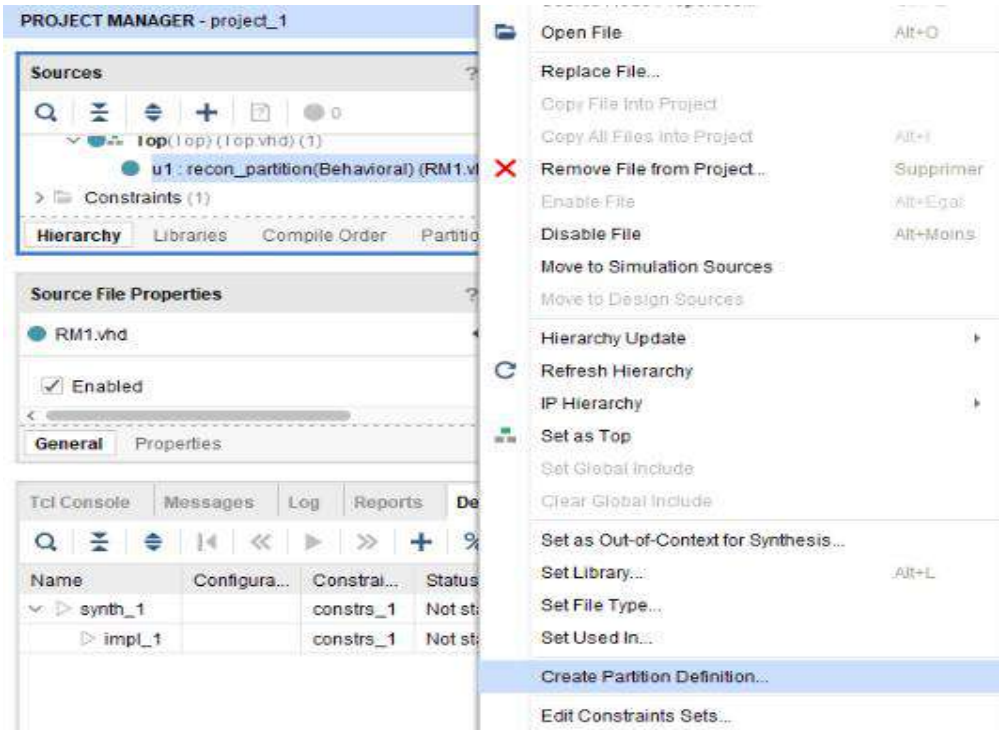


Figure III.14. *Création d'une partition reconfigurable*

Dans la boîte de dialogue qui apparaît ensuite, on donne un nom unique à cette définition de partition. Puis on définit un nom pour le premier module reconfigurable (RM). Ce RM est créé à partir des sources RTL ou netlist résidant actuellement à ce niveau de la hiérarchie, d'autres RM sont ajoutés ou créés plus tard dans le flux.

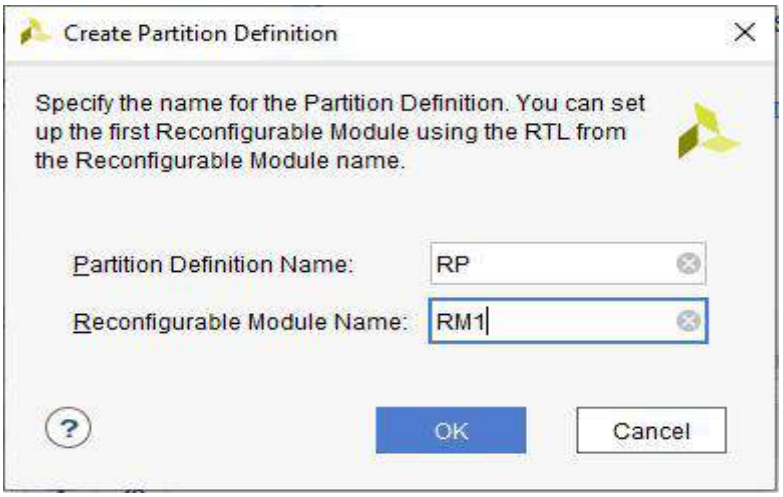


Figure III.15. *Définition de la partition reconfigurable et du premier module reconfigurable*

4.2.b *Finalisation de la structure du projet de reconfiguration partielle*

Après avoir défini les Partitions Reconfigurables, on a entré les détails complets du projet. Cela consiste à ajouter des modules reconfigurables pour chacune des partitions reconfigurables, à définir un ensemble complet de configurations qui combinent les RM avec la conception statique et à déclarer l'ensemble des exécutions qui seront utilisées pour mettre en œuvre toutes les configurations. Tous ces ajouts sont effectués dans l'assistant de reconfiguration partielle.

On ouvre l'Assistant de reconfiguration partielle en sélectionnant l'étape dans le Navigateur de flux ou dans le menu Outils.

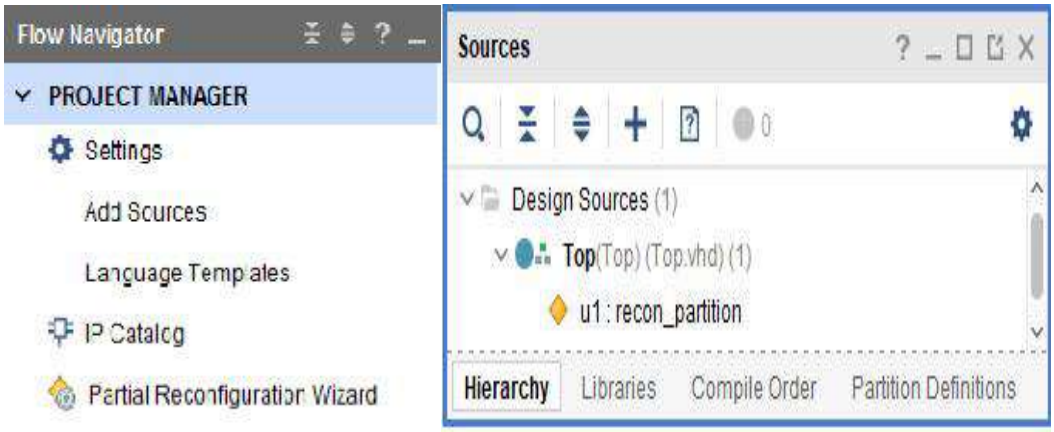


Figure III.16. *L'assistant de reconfiguration partielle dans le navigateur de flux*

4.2.c *Edition de modules reconfigurables*

Après avoir sélectionné « Suivant » pour progresser au-delà de l'introduction, la première page de contenu nous permet de définir de nouveaux modules reconfigurables pour toute définition de partition définie. Le premier RM pour chaque PD a déjà été inclus ici, après on ajoute les deux nouveaux modules reconfigurables RM2 et RM3.

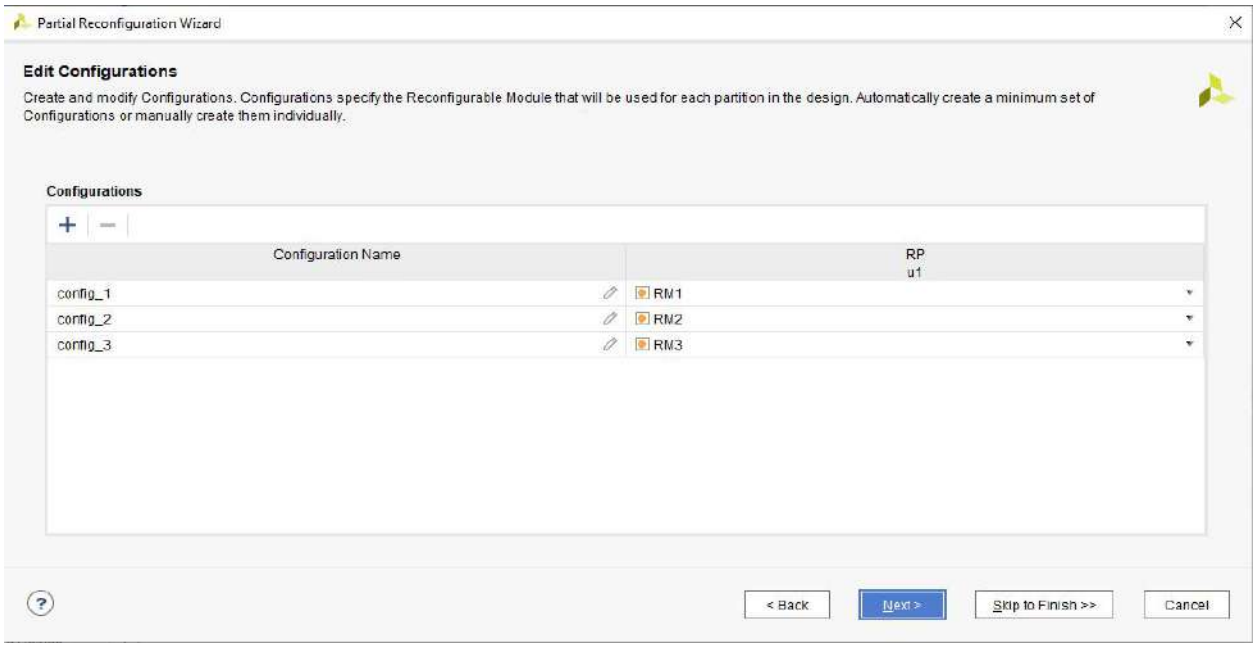


Figure III.17. *Création d'un nouveau module reconfigurable*

4.2.d *Edition des exécutions de configuration*

Une fois toutes les configurations définies, on passe à l'écran final pour gérer les cycles de configuration qui leur sont associés. Tout comme les configurations elles-mêmes, Vivado peut créer automatiquement un ensemble de Configuration Runs. La première configuration de la liste est définie comme le parent, et toutes les autres configurations sont définis comme des enfants de ce parent.

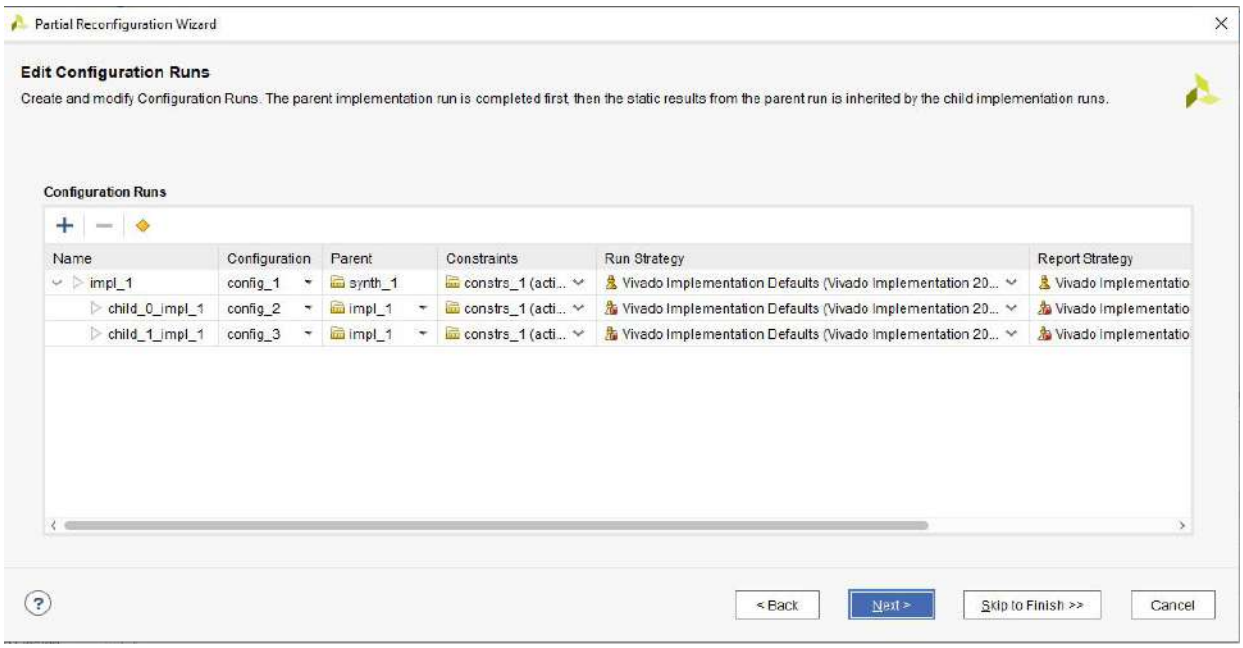


Figure III.18. *Configuration automatique*

4.2.e *Mise en œuvre de la conception des RP*

Une fois que toutes les configurations nécessaires ont été définies, la conception peut être synthétisée et mise en œuvre. Le navigateur de flux peut être utilisé pour passer par toutes les étapes de la synthèse, du lieu et de l'itinéraire, et même de la génération de bitstreams. Le Navigateur de flux fonctionne sur le run actif comme un flux standard, mais il lancera tous les runs child en plus du parent actif. Un détail requis pour les conceptions de reconfiguration partielle est un Pblock pour chaque partition reconfigurable.

Après avoir dessiné le Pblock pour une partition reconfigurable, ses propriétés peuvent être vues dans la fenêtre Propriétés du Pblock sous la vue Propriétés. Deux options propres aux partitions reconfigurables sont disponibles ici : RESET_AFTER_RECONFIG (série 7 uniquement) et SNAPPING_MODE. La vue Statistiques indique les ressources disponibles et utilisées pour le module reconfigurable actuellement chargé, il est donc important de prendre en compte les besoins des autres RM également.

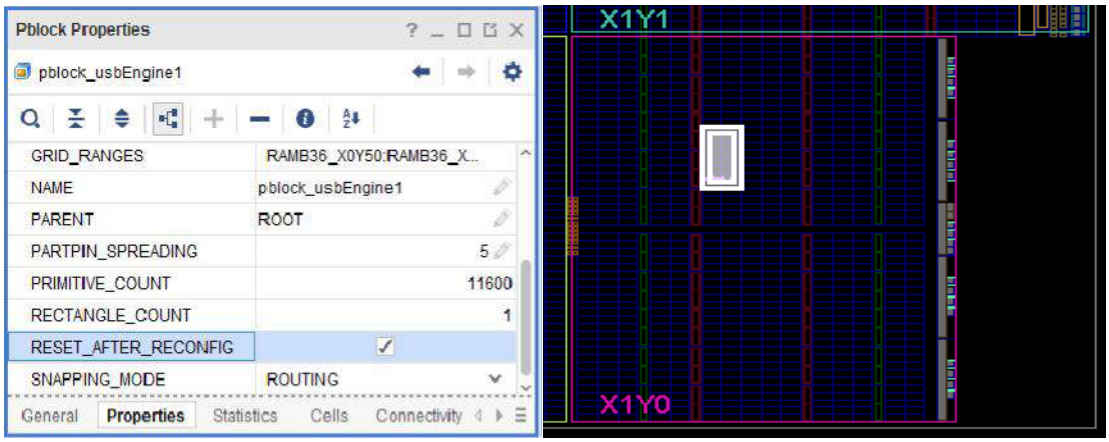


Figure III.19. *Propriétés de reconfiguration partielle sur un Pblock (7 séries)*

4.2.f *Génération de bitstreams*

Une fois que toutes les configurations souhaitées ont été placées et acheminées, des bitstreams peuvent être générés. Tout comme pour l'implémentation, le bouton Générer les bitstreams dans le navigateur peut être utilisé. Cela lancera write_bitstream pour toutes les exécutions enfant ainsi que pour le parent actif. Un appel local par clic droit pour write_bitstream sur n'importe quelle configuration est également disponible.

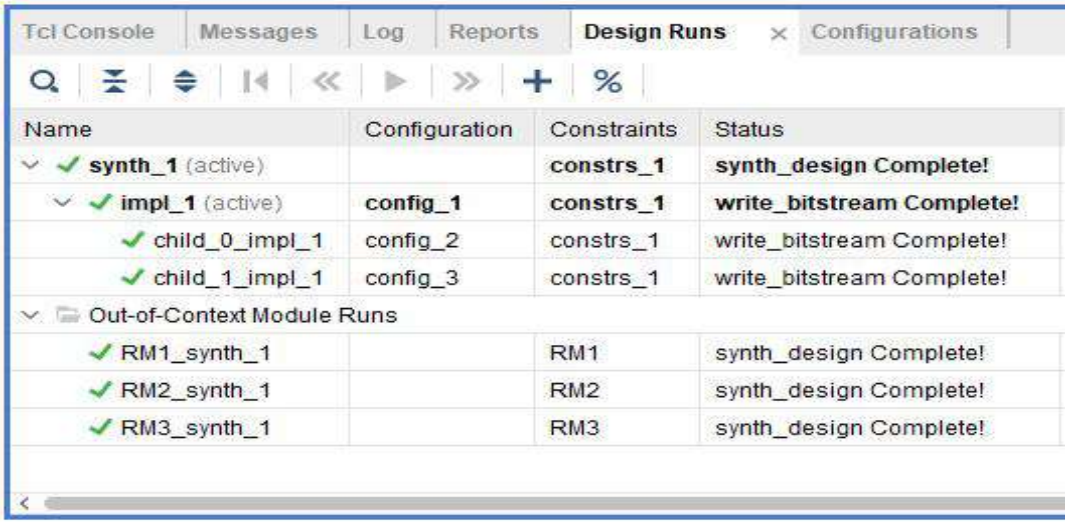


Figure III.20. *Génération des bitstream*

4. Configuration FPGA

Après la mise sous tension, le FPGA Artix-7 doit être configuré (ou programmé) avant de pouvoir exécuter toute fonction. Nous pouvons configurer le FPGA de l'une des quatre manières suivantes :

- Un PC peut utiliser le circuit Digilent USB-JTAG (port J6, intitulé « PROG ») pour programmer le FPGA à tout moment de la mise sous tension.
- Un fichier stocké dans le périphérique flash série non volatile (SPI) peut être transféré vers le FPGA à l'aide du port SPI.
- Un fichier de programmation peut être transféré sur le FPGA à partir d'une carte micro SD.
- Un fichier de programmation peut être transféré depuis une clé USB connectée au port USB HID.

La figure III.21, représente les différentes options disponibles pour la configuration du FPGA. Un cavalier « mode » intégré (JP1) et un cavalier de sélection de support (JP2) permettent de sélectionner les modes de programmation.

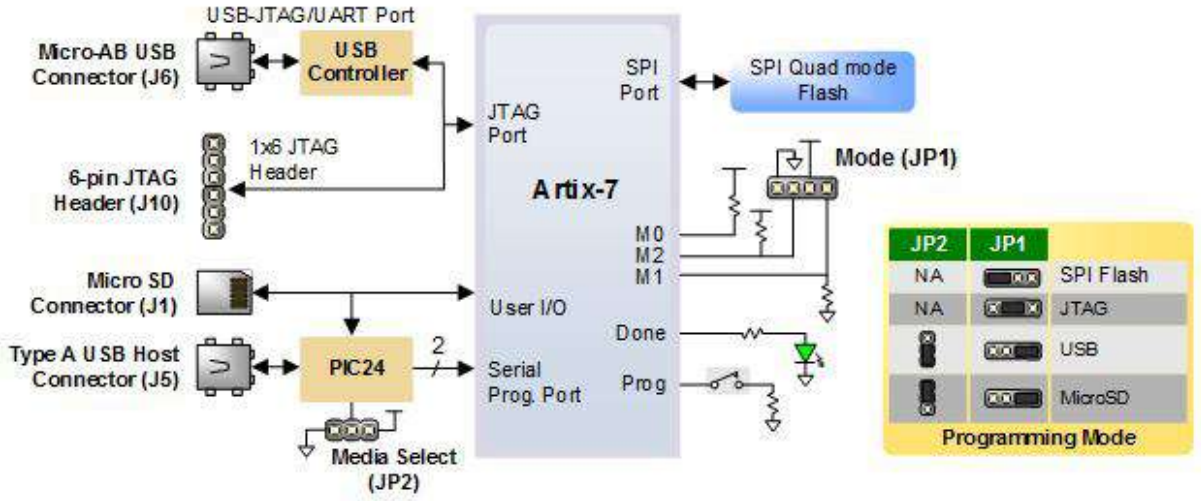


Figure III.21. *Options de configuration du Nexys4 DDR*

4.1. *Bitstream*

Les données de configuration FPGA sont stockées dans des fichiers appelés bitstreams d'extension de fichier : «.bit ». Les logiciels ISE ou Vivado de Xilinx peuvent créer des Bitstreams à partir de fichiers sources VHDL, Verilog ou schématiques (dans le jeu d'outils ISE, EDK est utilisé pour les conceptions à base de processeur intégré MicroBlaze).

Les bitsreams sont stockés dans des cellules de mémoire basées sur la mémoire SRAM au sein du FPGA. Ces données définissent les fonctions logiques et les connexions de circuit du FPGA. Elles restent valables jusqu'à ce qu'elles soient effacées en coupant l'alimentation de la carte, en appuyant sur le bouton de réinitialisation attaché à l'entrée PROG ou en écrivant un nouveau fichier de configuration à l'aide du port JTAG.

Le temps nécessaire à la programmation du Nexys4 DDR peut être réduit en compressant le bitstream avant la programmation, puis en permettant au FPGA de décompresser le bitstream lui-même lors de la configuration. En fonction de la complexité de la conception, des taux de compression de 10x peuvent être atteints. La compression de bitstream peut être activée dans les outils Xilinx (ISE ou Vivado) pour se produire lors de la génération. Après avoir été programmé avec succès, le FPGA provoquera le voyant "DONE" à éclairer. Nous appuyons sur le bouton "PROG" à tout moment pour réinitialiser la mémoire de configuration du FPGA. Après avoir été réinitialisé, le FPGA tentera immédiatement de se reprogrammer à partir de la méthode sélectionnée par les cavaliers en mode de programmation.

4.2. *Configuration JTAG*

Les outils Xilinx communiquent généralement avec les FPGA à l'aide de l'architecture Test Access Port et Boundary-Scan, généralement appelée JTAG. Pendant la programmation JTAG, un fichier « .bit » est transféré du PC vers le FPGA à l'aide du circuit Digilent USB-JTAG intégré (port J6) ou d'un programmeur JTAG externe, tel que le Digilent JTAG-HS2, connecté au port J10. Nous pouvons exécuter la programmation JTAG à tout moment après la mise en sous tension du Nexys4 DDR, quel que soit le type de cavalier défini pour le cavalier de mode (JP1). Si le FPGA est déjà configuré, la configuration existante est remplacée par le bitstream transmis via JTAG. Le paramétrage du cavalier de mode sur le paramètre JTAG (illustré à la figure III.21) est utile pour empêcher la configuration du FPGA à partir d'une autre source de bitstream jusqu'à ce qu'une programmation JTAG ait lieu.

La programmation du Nexys4 DDR avec un bitstream non compressé à l'aide du circuit USB-JTAG intégré prend généralement environ cinq secondes. La programmation JTAG peut être effectuée à l'aide du serveur matériel de Vivado ou de l'outil iMPACT inclus avec ISE et de la version Lab Tools de Vivado.

5. Expérimentations et résultants

5.1. *Installation expérimental*

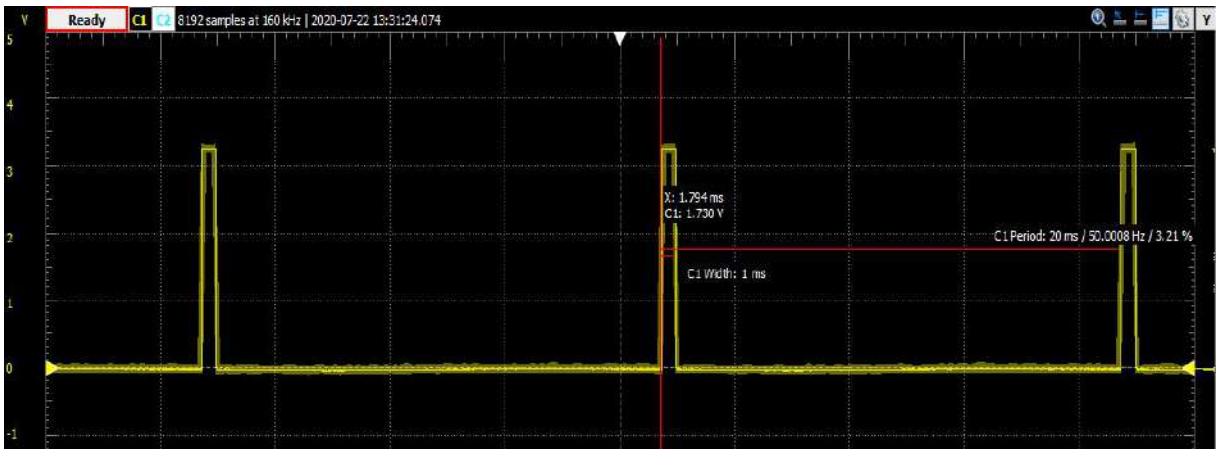
Le système matériel/logiciel de reconfiguration dynamique partielle du servomoteur a été testé afin d'étudier ses performances et la relation entre les différents paramètres techniques de la conception. L'implémentation a été faite avec succès, on a utilisé l'instrument de mesure Digilent Analog Discovery 2 pour visualisé les signaux de sortie de la carte FPGA.

Digilent Analog Discovery 2 est un oscilloscope USB, un analyseur logique et un instrument multifonctionnel qui permet aux utilisateurs de mesurer, visualiser, générer, enregistrer et contrôler des circuits à signaux mixtes de toutes sortes. Développé en conjonction avec Analog Devices et soutenu par le programme universitaire Xilinx. [14]

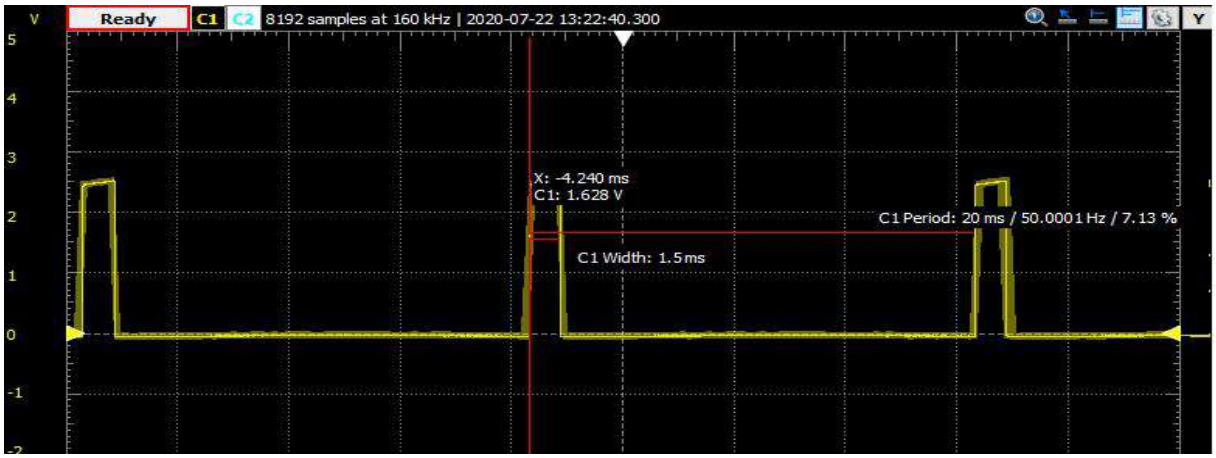


Figure III.22. *Analog Discovery 2*

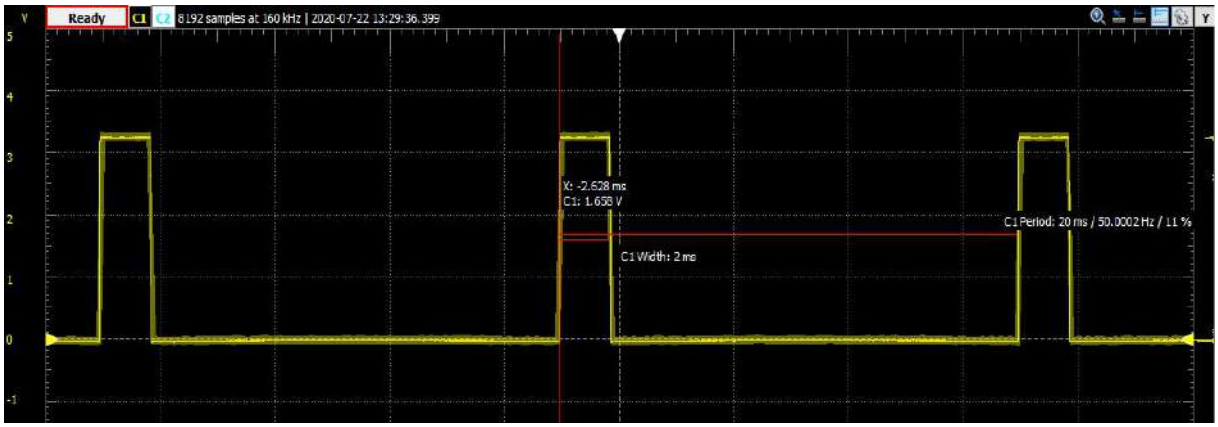
Comme indique la fiche technique du constructeur, le servomoteur fonctionne dans une plage de 20 ms, les figures III.23 (a, b, c) illustre les signaux du servomoteur de 0°,90°,180°.



(a) Position 0°



(b) position 90°



(c) position 180°

Figure III.23. *Les signaux de commande du servomoteur*

L'installation du système se compose d'un pc, une carte FPGA Nexys4 DDR, un servomoteur et d'une alimentation externe, l'installation est présentée sur la figure suivant :

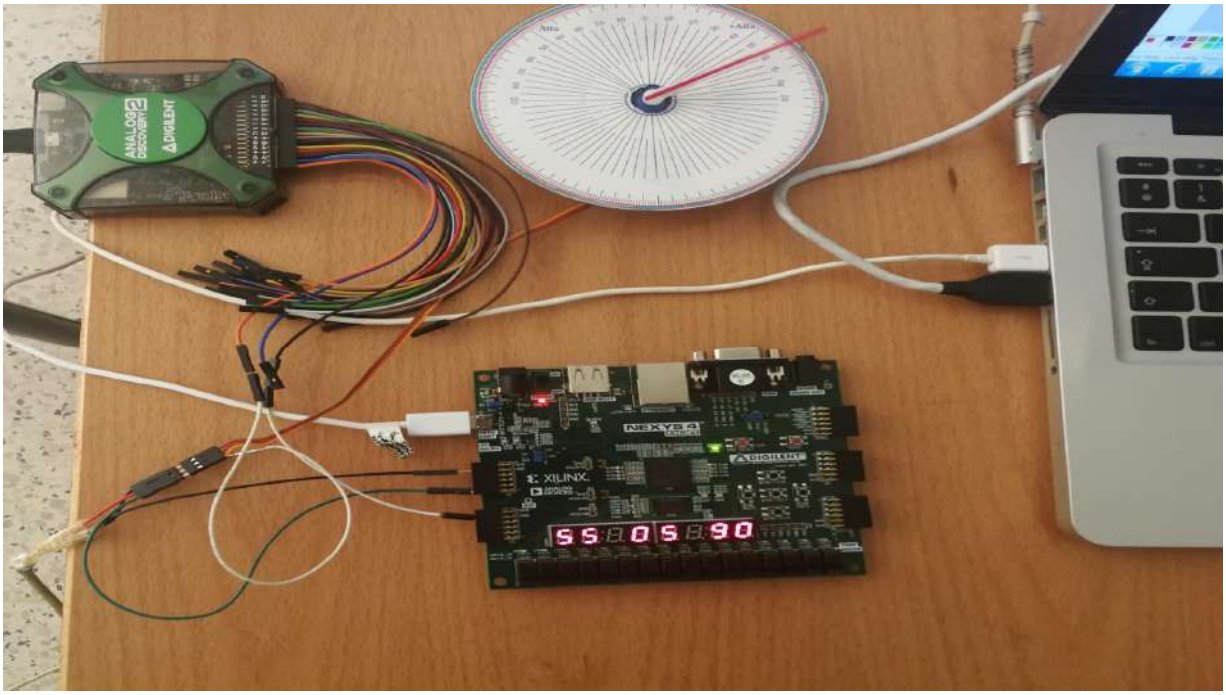


Figure III.24. *Installation du système*

5.2. *L'implémentation sur la carte*

Pour tester la reconfiguration dynamique partielle les deux premier bitstreams (la partie statique et RM1) doit être implémentés ensembles. Après l'implémentation le système démarre à fonctionner normalement suivant l'architecture désiré.

Pour le reconfigurer dynamiquement, nous laissons la partie statique en fonctionne et nous remplaçons le bitstream (RM1) par le Bitstream (RM2). Cela nous permet de changer le fonctionnement des boutons poussoirs pendant que le système reste en fonctionnement, ici le fonctionnement des boutons poussoirs est suspendu suivant l'architecture du RM2.

Afin de test et pour mieux confirmer que la reconfiguration dynamique partielle fonctionne normalement nous laissons la partie statique en fonctionne et nous implémentons le quatrième Bitstream (RM3) qui comporte une architecture inversée que le RM1, ici le fonctionnement des boutons poussoirs est inversé par rapport au bitstream RM1.

6. Conclusion

Dans ce chapitre, une plate-forme de commande matérielle d'un servomoteur a été conçue. La technologie de reconfiguration dynamique partielle du FPGA a été utilisée pour faire en sorte que la plate-forme s'adapte à différentes configurations de bouton poussoir pour faire commander l'aiguillé du servomoteur. La conception a été testée sur une carte Nexys4 DDR.

Au cours de ce chapitre on a réalisé le fichier principal et les modules reconfigurables qui ont été implémentés sur la carte FPGA de type Nexys4 DDR en utilisant le logiciel vivado.
En fin nous avons vérifié nos résultats expérimentaux sur la carte.

Conclusion générale

Les architectures reconfigurables à base d'FPGA sont capables de modifier le comportement d'une partie du système pendant que le reste du circuit continue de s'exécuter normalement.
La reconfiguration dynamique partielle permet d'augmenter l'efficacité d'un FPGA en allouant des ressources logiques à plusieurs tâches. Il est possible d'implémenter une application de taille plus grande que la mémoire physique disponible sur le FPGA. Par conséquent, un FPGA reconfigurable dynamiquement peut offrir une quantité plus importante de ressources logiques qu'il en dispose.
Un autre avantage qui suscite un intérêt croissant est la flexibilité introduite par la reconfiguration dynamique des FPGA qui se manifeste à deux niveaux. Le premier niveau permet au concepteur, d'adapter facilement son application pour faire face à de nouvelles contraintes ou pour remplacer un algorithme par un autre plus efficace. Cette flexibilité assure une certaine évolution du système. Le second niveau se situe sur le choix des tâches à exécuter, qui peut se faire dynamiquement en fonction des données à traiter. On peut donc tout à fait imaginer que les données d'entrée puissent influer en temps réel sur l'enchaînement des algorithmes. Le matériel devient extrêmement malléable et contrôlable par le logiciel.
L'objectif de ce projet est de commander un servomoteur à base d'une carte FPGA du type Nexys4 DDR en utilisant la reconfiguration dynamique partielle.

Toutefois, il est intéressent d'approfondir cette étude, à savoir par : l'utilisation du Port d'accès à la configuration interne (ICAP) pour faire la reconfiguration dynamique partielle automatiquement.

Bibliographie

[1] Dutrieux, L., & Demigny, D. (1997). *Logique programmable: architecture des FPGA et CPLD, méthodes de conception, le langage VHDL*. Eyrolles.

[2] M. Philippe LETENNEUR – Lycée Julliot de la Morandière *GRANVILLE***.** Les circuits logiques programmables.

[3] Bailey, D. G. (2011). *Design for embedded image processing on FPGAs*. John Wiley & Sons.

[4] Snaidero, S. (2004). Modélisation multidisciplinaire VHDL-AMS de systèmes complexes: vers le Prototypage Virtuel (Doctoral dissertation, Université Louis Pasteur-Strasbourg I).

[5] Smith, D. J. (2006). HDL Chip Design a Practical Guide for designing, synthesis & simulating ASICs & FPGAs using VHDL or Verilog. MGH. pp179-183 and, 198, 201.

[6] Deckers, D. (2008). Composants programmables. Cours de 3è Bachelier en Electronique appliquée, Institut Supérieur Industriel de Mons Haute Ecole de la Communauté française en Hainaut Unité Electronique, 2009.

[7] Guihal, D. (2007). Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires (Doctoral dissertation).

[8] Tavernier, C. (1996). *Circuits logiques programmables*. Dunod.

[9] Marques, N. (2012). *Méthodologie et architecture adaptative pour le placement efficace de tâches matérielles de tailles variables sur des partitions reconfigurables* (Doctoral dissertation).

[10] Shah, M. (2018). *Flexible Partial Reconfiguration Based Design Architecture for Dataflow Computation* (Doctoral dissertation).

[11] Programmable-Logic: Nexys-4-DDR: Reference-Manual.

https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual]

[12] TowerPro, S. Mini Micro Digital Servo 9g for RC Models.

[13]: Vivado Design Suite and User Guide, Release Notes, Installation, and Licensing: 2018, Xilinx.

[14] Analog Discovery 2.

https://store.digilentinc.com/analog-discovery-2-100msps-usb-oscilloscope-logic-analyzer-and-variable-power-supply/