République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de L'**E**nseignement **S**upérieur et de la A **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DE **G**ENIE **E**LECTRIQUE ET D' **I**NFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de Fin d'Etude de MASTER PROFESSIONNEL

Spécialité : **Electronique Industrielle**
Filière : **Génie électrique**

Présenté par

BOUGHANIME BACHIR
BOUDLAL LYES

Mémoire dirigé par **Hammiche Hamid** et co-dirigé par **Mellah Rabah**

Thème

commande force position d'une architecture de teleoperation a base d'une carte FPGA

Mémoire soutenu publiquement le 20 septembre 2017 devant le jury composé de :

Mr : Zermi Rachid,	Président
Mr : Tahanout Mouhamed,	Examinateur
Mr : Hammiche Hamid,	promoteur
Mr : Mellah Rabah	co-promoteur

Dédicaces

Je dédie ce modeste travail à :

Toute ma chère famille

Mes amis

Notre promotion de l'année d'étude 2016/2017

A la mémoire de mon cher ami IHOUT CHAFAA paix a son âme.

boudlal lyes .

Dédicaces

Je dédie ce modeste travail à :

Toute ma chère famille

Mes amis

Notre promotion de l'année d'étude 2016/2017

A la mémoire de mon cher ami IHOUT CHAFAA paix a son âme.

boughanime bachir

Sommaire

Introduction générale 01

Chapitre I : synthèse d'une architecture de télé-opération 03

I.1 Introduction 04
I.2 bref historique 05
I.3 Modélisation du système 06
I.3.1 Modélisation du comportement de l'opérateur humain 08
I.3.2 Modélisation de l'environnement 08
I.3.3 Modélisation des manipulateurs maître et esclave 09
I.3.4 Modélisation du réseau de communication 10
I.4 Les systèmes de télé opération avec retour d'effort 10
I.5 Stratégies de commande bilatérale 11
I.5.1 Asservissement symétrique bilatéral 11
I.5.2 Asservissement Asymétrique bilatéral 11
I.5.2.1 Principe 12
I.6 Performance et stabilité en télé opération 12
I.6.1 Stabilité 13
I.6.2 passivité 13
I.6.3 Transparence 14
I.7 couplage par les positions, couplage par les vitesses 14
I.8 influence des retards sur les systèmes avec retour d'efforts 15
I.8.1 Qualité de la télé présence 15
I.8.2 Asservissement Asymétrique 15
I.9 Structure à deux canaux Force-position 17
I.10 Conclusion **18**

Chapitre II : description du langage VHDL 19

II.1 Introduction 20
II.2 Historique 21
II.3 Définition du langage VHDL 21
II.4 La structure d'une description VHDL 22
II.4.1 Déclaration des bibliothèques 22
II.4.2 Déclaration de l'entité 22
II.4.2.1 Le signal d'entré/sortie 23
II.4.3 Les architectures 23
II.4.3.1 Description comportementale 24
II.4.3.2 Description structurelle 25
II.4.3.3 Description Mixte 26
II.5 Les instructions concurrentes et séquentielles 26
II.5.1 Les instructions concurrentes 26
II.5.1.1 L'affectation simple 27
II.5.1.2 L'affectation conditionnelle 27
II.5.1.3 L'affectation sélective 27
II.5.1.4 Les opérateurs 28

II.5.2 Les instructions séquentielles 30
II.5.2.1 L'affectation simple 31
II.5.2.2 L'instruction conditionnelle 31
II.5.2.3 L'instruction de choix 32
II.6 Définition de procès 34
II.7 Les fonctions et procédures 35
II.7.1 Rôle, principe et fonctionnement 35
II.7.2 La déclaration des fonctions et procédures 35
II.8 Les attributs 35
II.9 Paquetage 35
II.10 Les différences avec un langage de programmation 36
II.11 Les avantages du langage VHDL 37
II.12 Environnement ISE 38
II.13 Conclusion 38

Chapitre III : présentation de la carte FPGA 39

III.1 Introduction 40
III.2 Définition des FPGA 42
III.3 Architecture interne d'un FPGA 43
III.3.1 Les blocs logiques configurables (CLB 43
III.3.2 Blocs d'entrée-sortie (IOB 44
III.3.3 Ressources d'interconnexions 44
III.4 FPGA de la famille XLINX 46
III.5 Présentation et description des éléments de FPGA ML 501 47
III.6 conclusion 50

Chapitre IV : présentation de la commande de la télé-opération 51

IV.1 Introduction 52
IV.2 description du système 52
IV.3 modélisation du système 52
IV.4 élaboration de la commande 54
IV.5 résultat de simulation avec SIMULINK 55
IV.5.1simulation de l'architecture sur matlab SIMULINK 55
IV.6 Conversion du matlab au VHDL 58
IV.6.1 numérisation des régulateurs 58
IV.6.2 génération du code VHDL 63
IV.7 Conclusion 64

Conclusion générale : 65

Introduction générale [16]

Le mot (télé opération) a son origine dans le mot grec (têle), qui signifie loin, à distance, et le mot latin (operor), qui signifie travailler, œuvrer, accomplir. La notion de télé opération désigne donc une interaction entre l'opérateur et le robot. L'opérateur envoie une consigne et le robot l'exécute. Cette consigne peut prendre plusieurs formes : une destination à atteindre, un plan (suite d'actions) à exécuter, une vitesse à atteindre, un comportement à adopter etc.

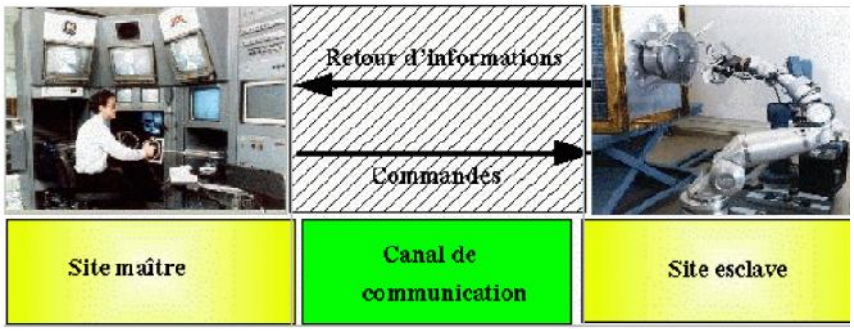
Les premiers télémanipulateurs étaient des systèmes mécaniques composés d'une partie maîtresse, tenue par l'operateur et d'une partie esclave, exacte réplique de la partie maîtresse et munie d'un outil à son extrémité. La transmission entre les deux sites été purement mécanique opérant a courte distance et il transmit à l'opérateur les forces exercés sur la partie terminale. L'apparition des ordinateurs et le progrès en matière de technologie ont permis le développement de systèmes plus en plus performant dans divers fonctions et les munir d'actionneurs électriques et de différents capteurs : capteur de position, de vitesse, de couple.

En présence de retards, il ya un délai entre le moment où l'opérateur réalise une action (par exemple un déplacement du robot maître) et le moment où il observe l'effet de son action (par exemple, le moment où il voit le robot esclave bouger). La stratégie généralement adoptée est la stratégie dite « action attente » (‘’move and wait strategy ‘’) : l'operateur réalise une succession de petits déplacements et attend d'observer le résultat de l'action précédente pour passer à l'action suivante. Il en résulte une augmentation très importante du temps de réalisation d'une tâche. L'autre effet important, mis en évidence est l'apparition d'instabilités dans les systèmes à retour d'effort. Or une bonne gestion de ces forces de contact est cruciale pour certaines tâches de télé opération.

Mon projet est basé sur la présentation d'une architecture de télé opération bilatérale à un degré de liberté.

I.1 Introduction : [17]

L'objectif d'un système de télé opération haptique est de permettre à son utilisateur de réaliser une tâche à distance tout en ressentant les forces mises en jeu à l'endroit de la manipulation. Son fonctionnement est basé sur l'échange d'informations de positions et de forces entre le maître, qui est la télécommande de l'utilisateur, et l'esclave qui est le robot réalisant effectivement la tâche. Ce principe est illustré à la (Figure 1) dont les différents éléments seront décrits dans la suite de ce chapitre.



Figure(I.01) : Illustration de l'architecture générale d'un système de Télé opération

Idéalement, le système de télé opération doit être complètement transparent, en reflétant fidèlement les interactions entre l'esclave et l'environnement sur l'opérateur.
Celui-ci doit pouvoir ressentir les efforts comme s'il réalisait directement la tâche à la place de l'esclave. En pratique, une transparence parfaite est impossible.
Elle sera limitée par la dynamique naturelle de l'interface du maître et celle de l'esclave (inerties, frottements, etc.), par le choix de la méthode de contrôle ainsi que par la présence de délais, de bruits et de la digitalisation. En fonction du nombre et du type de mesures disponibles différentes méthodes de contrôle peuvent être implémentées.
Cette étude se base sur un système à 1 degré de liberté (1 ddl) (linéaire). Malgré la présence en pratique de phénomènes non-linéaires (frottements secs, impacts,...), cette démarche permet de comprendre les principaux enjeux du contrôle.

Dans ce chapitre on donne une description mathématique d'un système de télé opération, avec un formalisme couramment rencontré dans la littérature.
Sur base de cette représentation, des critères de performances seront définis pour quantifier la transparence du système. Ensuite, les principales méthodes de contrôle seront décrites et comparées par rapport à ces critères. Pour finir, une étude sur la stabilité sera présentée pour mettre en évidence le compromis existant entre la recherche d'une meilleure transparence et la stabilité.

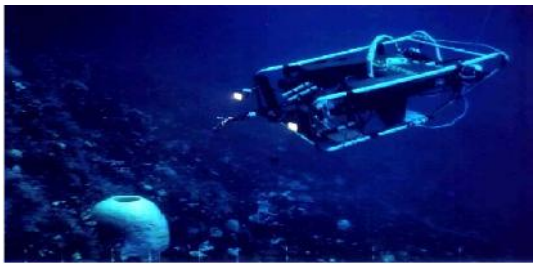
I.2 bref historique : [17]

L'histoire de la télé opération commence avec le manipulateur maître-esclave développé par Ray Goertz à l'Argonne National Laboratory (ANL) en 1948. Ce système conçus pour la manipulation de substance toxique (manipuler le carburant radioactif). Dans ce premier système, le manipulateur maître possédait la même structure mécanique et les mêmes propriétés cinématiques que le manipulateur esclave donc les limitations physiques de l'esclave étaient perçus naturellement par l'utilisateur. En 1954 l'ANL développe la deuxième génération de télémanipulateurs, électriques à retour d'effort. Dans un système à retour d'effort, toute force externe expérimentée par le manipulateur esclave est produite sur le manipulateur maître. Ceci est utilisé pour implanter le contrôle bilatéral : une force appliqué sur l'esclave (respectivement le maître) produira un mouvement du maître (respectivement l'esclave). Les systèmes hydrauliques ont aussi été présents depuis le début de la télé opération. Le premier système de ce type est le (Handyman) de Ralph Mosher développé aux laboratoires de general electric en 1958. Le Handyman, manipulateur à retour d'effort consistait en deux bras hydraulique à 10 degrés de liberté, 2 degrés de liberté pour chaque doigt.

Figure(I.02) : Environnement de télémanipulation au Argonne National Laboratory

Au début des années 60 le champ d'application de la télé opération s'étend à l'exploration spéciale donc à la télémanipulation et au contrôle de véhicule avec des

retards de transmission. En 1970, l'exploration sous-marine devient l'un des principaux champs d'applications des systèmes télé opérés.



Figure(I.03) : ROV de la NASA pendant la réalisation d'une mission en Antarctique

Dans les années 80, les véhicules opérés à distance ROV (Remotely operated vehicles) commencent à être amplement dans les industries des gaz et de petrole.les ROV sont utilisés pour la surveillance oléoducs et pour l'inspection des structures artificielles sous-marines.

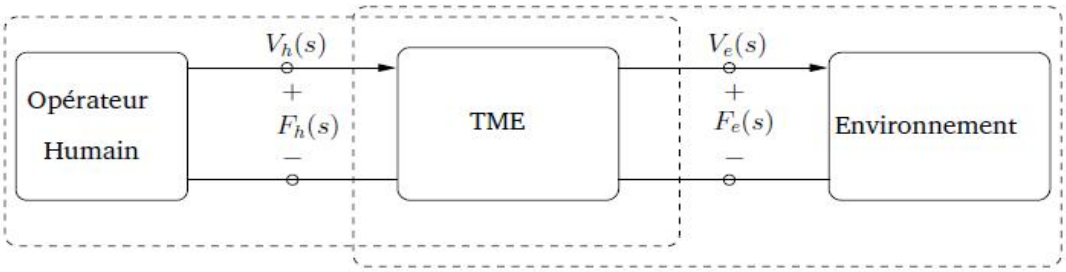
L'apparition des ordinateurs et le progrès en matière de technologie ont permis le développement du systèmes de plus en plus performants dans divers fonctions : le dépannage de satellite, l'exploration de volcans ,le déminage, l'aide aux personnes handicapées (avec, par exemple les fauteuils roulant intelligents…etc.)

I.3 Modélisation du système :[18]

La modélisation d'un TME (Téléopérateur Maitre Esclave) permet de mieux comprendre les interactions entre les différents éléments du système, réaliser des simulations pour mettre en place des stratégies de commande ou encore analyser la stabilité et les performances du TME. Pour cela, deux techniques sont classiquement utilisées dans la littérature :

– la modélisation sous forme de schémas blocs, où chaque élément est modélisé par une fonction de transfert ;

– la modélisation sous forme de quadripôles ou ports d'interaction, particulièrement bien adaptée à l'analyse et la modélisation des systèmes de télé opération avec retour d'efforts, elle permet de représenter de manière assez intuitive les échanges d'énergie et les interactions entre les éléments d'un TME, Ce type d'approche est basé sur une analogie entre modèle électrique et mécanique, où les courants sont remplacés par les vitesses et les tensions par les efforts.



Figure(I.04) : Représentation sous forme quadripôle d'un système de télé opération avec retour d'efforts.

Il est alors possible d'exprimer une relation entrée/sortie du TME sous forme matricielle, les représentations les plus courantes utilisent l'une ou l'autre des matrices d'impédance, d'admittance ou hybride.

La représentation sous forme hybride est certainement la plus utilisée, pour cette représentation les relations entre les efforts et les positions s'écrivent de la manière suivante :

$$\begin{vmatrix} T_h \\ \theta_s \end{vmatrix} = H(s) \begin{vmatrix} \theta_m \\ -T_e \end{vmatrix}$$

Avec la matrice hybride $H(s) = \begin{vmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{vmatrix}$

Les autres représentations matricielles (impédance et admittance) sont données par les expressions suivantes:

- $T(s) = Z(s)\theta(s)$

❖ La relation entre variables indépendantes (θ_m , θ_s) et les variables dépendantes (T_h, T_e) est décrite par une matrice Z qu'on appelle la matrice d'impédance qui généralise l'impédance opérationnelle d'un dipôle.

$$\begin{vmatrix} T_h \\ T_e \end{vmatrix} = \begin{vmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{vmatrix} \begin{vmatrix} \theta_m \\ -\theta_s \end{vmatrix} \text{ ..... Impédance}$$

❖ Si on choisi comme variables indépendantes les couples (T_h, T_e) au port du quadripôle on obtient la représentation :

- $\theta(s)$ = Y(s)T(s)

$$\begin{matrix} \theta_m \\ -\theta_s \end{matrix} = \begin{matrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{matrix} \begin{matrix} \mathrm{T_h} \\ T_e \end{matrix} \quad \text{……Admittance}$$

I.3.1 Modélisation du comportement de l'opérateur humain : [18]

La modélisation de l'être humain en interaction avec un système électromécanique est depuis très longtemps un domaine de recherche très actif. À notre connaissance, les premiers travaux sur le sujet datent de la fin des années 70, avec les modèles de McRuer ou de Kleinman qui sont utilisés dans le but de modéliser le comportement des pilotes de l'aérospatiale [McRuer67, Kleinman70]. Bien que les problématiques en télé robotique soient similaires, les chercheurs du domaine ne se sont intéressés à ces travaux que bien plus tard [Sheridan89, Brooks90]. Tous s'accordent aujourd'hui pour dire que l'être humain est un système extrêmement difficile à modéliser, car il est à la fois un système multi-modèle, adaptatif, ayant la capacité d'apprendre et pouvant ajuster son comportement aux sollicitations extérieures.

Lors d'une manipulation, l'être humain a tendance à fonctionner selon trois modes :

1) Le mode compensatoire qui fonctionne comme une boucle de rétroaction sur la trajectoire désirée et la trajectoire réelle ;
2) Le mode poursuite qui permet d'anticiper la trajectoire dans le cas où l'utilisateur connaît l'environnement avec lequel il interagit ;
3) Le mode de précognition qui s'appuie sur l'expérience et l'expertise de l'être humain.

Ces trois modes de fonctionnement sont généralement indissociables. L'être humain peut être amené à les combiner afin d'atteindre ses objectifs pour une tâche donnée.

I.3.2 Modélisation de l'environnement :[18]

D'une manière générale en télé robotique la perception et l'interaction avec l'environnement posent des problèmes. En effet, les objets manipulés ont des propriétés mécaniques différentes les uns des autres, ce qui rend leur modélisation a priori difficile. Cependant, pour une application médicale, nous savons que l'environnement sera exclusivement constitué d'organes et de tissus vivants mous. Dans ce cas, l'environnement peut être décrit par une impédance mécanique équivalente mettant en relation les vitesses $V_e(t)$, les efforts d'interaction $f_e(t)$ et les efforts internes $f_e\ t\ ^*$. Les efforts

internes représentent les efforts propres à l'environnement, qui ne résulte pas de l'interaction, comme par exemple les efforts dus aux mouvements physiologiques du patient. Lorsque la manipulateur esclave est en interaction avec un tel environnement, la relation dans le domaine de Laplace entre vitesse et efforts s'écrit :

$$Z_e(s) = \frac{f_e(s) - f_e(s)^*}{V_e(s)}$$

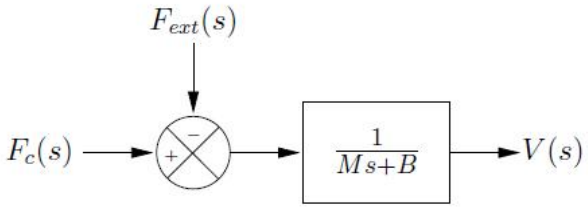
Avec $Z_e(s)$ l'impédance mécanique équivalente de l'environnement.

I.3.3 Modélisation des manipulateurs maître et esclave : [18]

En télémanipulation, les manipulateurs sont classés en deux familles, d'après les caractéristiques structurelles et le type d'actionneurs utilisés:

1) Les manipulateurs de type impédance sont comparables à des sources d'efforts, car leurs actionneurs sont commandés en efforts, lors d'une interaction avec un environnement, ils appliquent un effort en réponse à un déplacement de celui-ci. Généralement ils ont une faible inertie et sont réversibles.

Le schéma bloc d'un tel manipulateur en interaction avec l'environnement est représenté par cette figure :



Figure(I.05) : Manipulateur de type impédance

L'impédance est alors notée Z(s)=Ms+B, avec M et B représentent l'inertie et l'amortissement qui caractérisent le comportement dynamique du manipulateur. La vitesse V(s) du manipulateur résulte alors de l'effort de commande $F_c(s)$ et de l'effort d'interaction avec l'environnement $F_{ext}(s)$, suivant l'expression suivante :

$$V(s) = Z\,(s)^{-1}[F_c(s) - F_{ext}(s)]$$

La vitesse du manipulateur maître est désignée par $V_h(s)$ du fait de l'interaction avec l'utilisateur :

$$V_h(s) = Z_m\ s^{-1}[F_m\ s + F_h\ s\ ]$$

Si le manipulateur esclave est en contact avec un environnement, sa vitesse $V_e(s)$ dépend de l'effort d'interaction $F_e\ s$ et l'effort de commande $F_s\ s$, de la manière suivante :

$$V_e(s) = Z_s\ s^{-1}[F_s\ s - F_e\ s\ ]$$

Les signes dépendent naturellement du type d'interaction considérée : l'interaction entre l'utilisateur et interface maître, ou celle du manipulateur esclave avec l'environnement.

2) Les manipulateurs de type admittance sont comparables à des sources de position, ou de vitesse, car leurs actionneurs sont commandés en position, ou en vitesse, lors d'une interaction avec un environnement, ils appliquent une position, ou une vitesse, à cet environnement en réponse à un contact avec celui-ci. Ces manipulateurs sont généralement non réversibles et présentent une grande raideur.

I.3.4 Modélisation du réseau de communication : [18]

Le réseau de communication permet de transmettre les signaux d'un site à l'autre, ce qui a pour effet d'introduire des temps de retard dans la transmission de données, voire dans certain cas la perte de données. Les problèmes associés aux retards sont un véritable défi pour la communauté automatique, comme en atteste le nombre de publications sur le sujet.

Dans ce mémoire, nous n'intéresserons pas à ce problème, puisque dans notre contexte applicatif la communication entre les deux sites se fait par une carte entièrement dédié à cette utilisation et que la distance séparant le maître et l'esclave n'excède par quelques mètres.

I.4 Les systèmes de télé opération avec retour d'effort :

Le système de télé opération idéal est "transparent". Le robot esclave doit donc dupliquer exactement les déplacements du robot maître. Le robot maître doit restituer le plus fidèlement possible à l'opérateur les efforts exercés par l'esclave sur son environnement.

Dans le cas où les robots sont équipés d'actionneurs électriques, plusieurs solutions techniques sont envisageables pour réaliser ce retour d'effort selon que l'on dispose ou non du capteur d'effort.

Pour illustrer chaque type de couplage de télé opération, nous avons pris un "modèle" simplifié du robot (maître et esclave). Les robots ont un seul degré de liberté, ce sont de simples impédances mécaniques (masse, amortissement, raideur) et les actionneurs sont des générateurs de force parfaits (ce qui revient à considérer que la structure mécanique est réversible).

I.5 Stratégies de commande bilatérale :

Aujourd'hui, l'utilisation des TME s'est diversifiée. L'élaboration d'un cahier des charges précis sur le degré de transparence que l'on souhaite atteindre devient un véritable enjeu. Dans le domaine médical, il est primordial que le système soit stable, quelle que soit son utilisation.

Cependant, on constate que certaines manipulations, comme l'insertion d'aiguille, nécessitent une attention toute particulière en ce qui concerne la qualité de restitution des efforts.

Au fil des années, des problématiques classiques liées à l'utilisation des TME sont apparus. On mentionnera en particulier les problèmes de stabilité dus au contact avec des environnements très rigides, ou encore ceux liés aux retards dans le réseau de communication. [19]

I.5.1 Asservissement symétrique bilatéral :

Chaque robot est asservi sur la position de l'autre robot, l'asservissement bilatéral est réalisé en envoyant comme commande aux actionneurs la différence de position entre les deux robots, chaque robot reçoit une consigne proportionnelle à l'écart de position entre les deux robots.

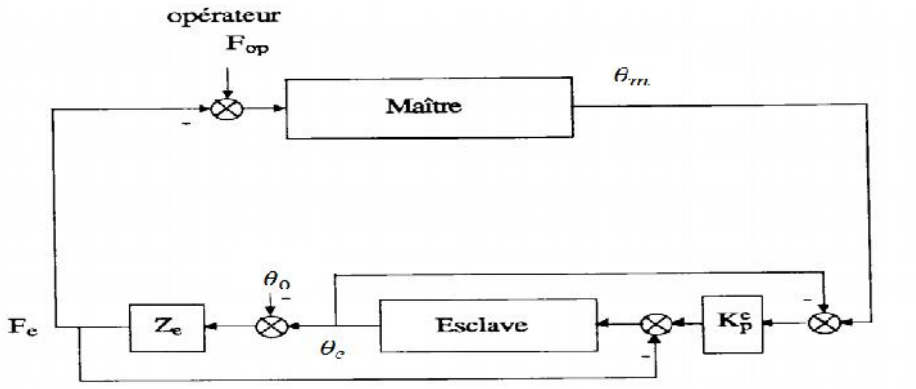
I.5.2 Asservissement Asymétrique bilatéral : [19]

Equipé les robots de capteurs d'efforts, le système de couplage résultant est asymétrique : asservissement de l'esclave sur la position de maître, et asservissement du robot maître sur les forces de l'esclave.

I.5.2.1 Principe :

Le schéma d'asservissement bilatéral asymétrique avec retour d'effort améliore généralement la télé présence (l'opérateur sent les forces exercées sur le robot maître comme si elles étaient exercées directement sur sa propre main).

Le robot esclave est asservi sur la position du maître et le robot maître reçoit comme consigne les forces appliquées par le robot esclave sur son environnement, et mesurées au moyen d'un capteur d'effort (figure7)



Figure(I.06) : Asservissement force-Position

La force mesurée F_e résulte des interactions de l'esclave avec un environnement de raideur Z_e. Si le contact a lieu en un point de coordonneés X_0, la relation liant la force du contact au déplacement est :

$$F_e = Z_e(\theta_e - \theta_0) \quad \text{.....I.5}$$

La figure 7 représente le cas idéal : la force mesurée est la force reéllement appliqué sur l'environnement par le robot esclave. En pratique, On cherchera à mesurer une force qui soit le plus proche possible de cette force réelle.

I.6 Performance et stabilité en télé opération :

Afin d'élaborer une loi de commande adéquate, il est nécessaire de définir les objectifs que l'on désire atteindre avec le système. L'objectif de la télémanipulation bilatérale est de fournir à l'utilisateur une transparence parfaite tout en garantissant la stabilité du système, quelles que soient les conditions d'utilisation. Ceci, traduit la qualité de restitution des efforts et des mouvements entre le maître et l'esclave. [19]

I.6.1 Stabilité : [20]

Pour les systèmes à retour d'effort, il n'ya pas de problème tant que le robot esclave est libre (la force retournée à l'opérateur est nulle), par contre les contacts poseront de sérieux problèmes de stabilité. Lorsque l'on considère un robot en contact avec un environnement, la stabilité de la commande en boucle fermée dépend non seulement de la dynamique du robot, mais aussi de celle de l'environnement.

Pour notre application, la stabilité de la commande peut être obtenue en se basant sur un modèle connu de l'environnement. L'outil privilégié pour garantir la stabilité de tel système est l'analyse de passivité. Plus précisément, on cherche à ce que le port d'interaction (correspond au point d'échange de puissance entre le système et la source d'énergie extérieur) entre le robot et l'environnement soit passif (ne crée pas d'énergie), en d'autres termes, le transfert entre les efforts externes appliqués au système et la vitesse v du système ne crée pas d'énergie. Une fois la passivité de ce port vérifiée, le système interagira via ce port, de façon stable, avec n'importe quel environnement passif. Donc la stabilité est déduite de la passivité.

I.6.2 passivité : [20]

Une définition générale de la passivité, inspirée de la théorie des réseaux, permet d'énoncer qu'un système est passif en un port d'interaction lorsque, pour un état initial quelconque du système et pour une entrée quelconque appliquée au système, la puissance extérieure, fournie au système via ce port, est positive.

Chaque port d'interaction d'entrée T (un effort), et de sortie V ou θ (vitesse ou position), est caractérisé par ses opérateurs immittance Z (impédance) ou Y (admittance) avec θ = Z(s)T et T= Y(s) θ. Ces opérateurs sont des matrices de transfert de fonctions réelles rationnelles en la variable de Laplace s = σ + jω.

Ce port d'interaction est passif si et seulement si la matrice Y(s) est Positive Réelle (PR).

De plus, d'après la Propriété 1 ci-après, la passivité d'un port d'interaction peut aussi être montrée par l'étude de la Positivité Réelle de Z(s).

Propriété 1: Si Y(s) est positive réelle et que Z(s) = Y(s) $^{-1}$ et [In + Y s]$^{-1}$ existent, alors Z(s) est PR. _

Cette propriété est très pratique car elle permet d'avoir le choix de montrer la passivité d'un port d'interaction soit par l'étude de la positivité réelle de son admittance Y(s), soit par l'étude de la positivité réelle de son impédance Z(s).

I.6.3 Transparence :

La notion de transparence est associée aux performances d'un système de télé opération. La transparence parfaite est atteinte lorsque l'utilisateur a l'impression de manipuler directement les objets de l'environnement, sans ressentir ni les effets dynamiques du manipulateur maître, ni ceux du manipulateur esclave. Dans la littérature, plusieurs définitions de la transparence ont été proposées. Parmi les plus cités, la correspondance d'impédance entre celle ressentie par l'utilisateur et celle de l'environnement permet d'exprimer le degré de transparence en termes d'impédance ressentie par l'utilisateur. Si le manipulateur esclave est en contact avec l'environnement et si l'interface maître est maintenue par l'utilisateur tout au long de la manipulation, il est alors possible d'établir une relation mathématique entre les mouvements et les efforts de chaque côté (cotés maitre et esclave).

I.7 couplage par les positions, couplage par les vitesses :

Pour piloter des déplacements, on pouvait envoyer des consignes de position ou de vitesse, ces deux couplages sont strictement équivalents. Mais si le robot esclave rencontre un obstacle, il ya une différence importante qu'il est bon d'avoir à l'esprit.

Prenons notre cas qui est un robot à un degré de liberté repéré par l'axe θ, si le robot esclave rencontre un obstacle à l'abscisse θ_e, l'asservissement commute automatiquement en asservissement de force et impose une force constante positive sur l'obstacle. Pendant ce temps l'operateur à pu commander un déplacement au delà de l'obstacle, qui n'est donc pas réaliser par le robot esclave.

Dans le cas où le robot esclave est asservie sur les positions du robot maître, il faudra que l'opérateur envoie une consigne hors de l'obstacle pour que le robot esclave quitte l'obstacle, c'est-à-dire inférieure à θ_e , par contre Dans le cas où le robot esclave est asservie sur les vitesses du robot maître, le robot esclave quittera l'obstacle dés que le robot maître aura envoyé une consigne de vitesse négative.

Donc dans le 1er cas, le robot esclave est asservie sur la position absolue du robot maître, dans le second cas, il ya une perte de cohérence entre la position du maître et celle de l'esclave.

I.8 influence des retards sur les systèmes avec retour d'efforts :

La présence de retard de transmission entre les deux sites d'un système de télé opération à asservissement bilatéral a pour effet principal de générer un frottement visqueux rapidement très important, cette effet ne peut être éliminée qu'en diminuant les gains des asservissements de position.

Un contact avec l'environnement est alors moins sensible à l'opérateur et la qualité de la téte présence est forcément diminuée.

I.8.1 Qualité de la télé présence :

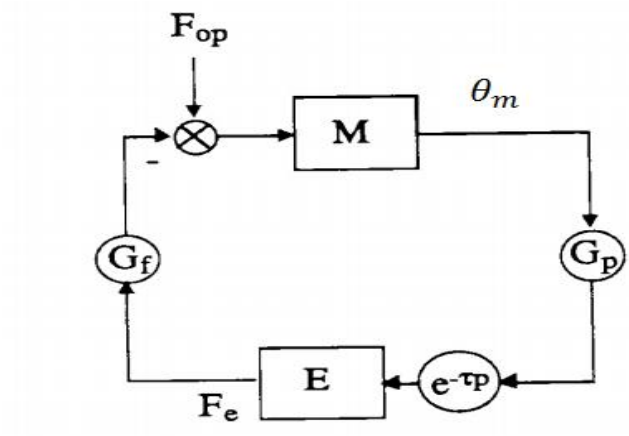
Supposons que le robot esclave soit en contact avec un environnement d'une certaine raideur. Lorsque l'opérateur exerce un effort sur le robot maître, le mouvement résultant est transmis comme consigne au robot esclave.

Comme celui-ci est en contact avec l'environnement l'asservissement de position génère une force (d'autant plus important que l'environnement est raide), cette force est ramenée au robot maître comme une contre réaction.

Le système bouclé maître-esclave de la figure 8 réalise donc un asservissement sur la force par l'opérateur et sur la position de l'environnement. La sensation de l'opérateur d'être réellement en contact avec l'environnement dépendra donc de la bande passante et du gain statique du système complet, on cherchera à augmenter au maximum le gain de retour de force.

I.8.2 Asservissement Asymétrique : [19]

Pour les systèmes à retour d'effort, dont le principe de couplage est représentée par la (figure 8), il n'ya pas de problème tant que le robot esclave est libre (la force retournée à l'opérateur est nulle), par contre les contacts poseront de sérieux problèmes de stabilité.



Figure(I.08) : Couple Maître-esclave à retour d'effort.

Le robot maître M est asservi en force, il reçoit comme consigne la force F_e exercée par l'esclave sur l'environnement à travers un gain G_f. Le robot esclave E est asservi en position, il reçoit comme consigne la position θ_m mesurée du maître à travers un gain G_p. On note M la fonction de transfert de l'ensemble composé par le robot maître et l'opérateur, cette fonction de transfert est caractérisée par un gain statique (raideur $\frac{1}{Z_m}$) et par une dynamique D1(p).la relation liant la position θ_m du robot maître à la force F_m exercée sur celui-ci est :

$$\theta_m = \frac{1}{Z_m} D1(p)\, F_m \qquad \text{avec} \quad F_m = F_{op} - G_f F_e$$

On note E la fonction de transfert de l'ensemble composé par le robot esclave asservi en position et l'environnement. On note Z_e la raideur de l'asservissement de position du robot esclave en contact avec son environnement, D2(p) est la dynamique de E. la relation liant la force F_e mesurée par le capteur d'effort du robot esclave et la position commandée θ_s est :

$$F_e = Z_e D2(p)\, \theta_s \qquad \text{avec} \quad \theta_s = G_p \theta_m$$

Pour simplifier on regroupe dans un terme D(p), la dynamique du couple maitre-esclave en boucle ouverte : D(p)=D1(p)D2(p).la fonction de transfert en boucle ouverte F(p) du couple maître-esclave est donc :

$$F(p)=e^{-Tp} G_f G_p \frac{Z_e}{Z_m} D(p)$$

Où T est le retard de transmission total dans la boucle.

On a représenté le diagramme de Bode du système sans prendre en compte le retard (trait bleu), et le diagramme de Bode du système en prenant compte le retard (trait rouge) sur la figure 7.

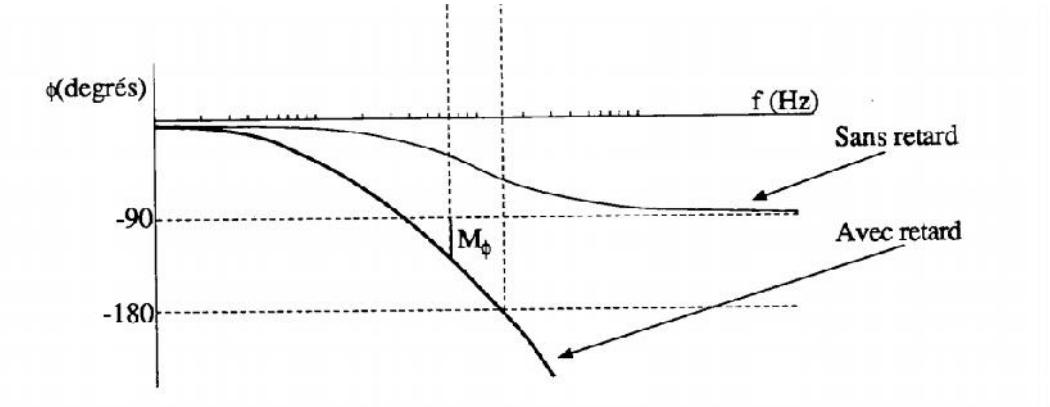


Figure9 : Diagramme de Bode d'un modèle simplifié d'un couple maître-esclave, influence de retard.

Pour que le système soit le plus transparent possible à l'opérateur, il faut que le gain total dans la boucle soit élevé et que la dynamique D(p) soit rapide.
Le retard pur amène un déphasage, il y a donc une diminution de la marge de phase $M_\emptyset$ Le retard pur pose donc forcément une instabilité, on peut jouer sur plusieurs facteurs pour tenter de stabiliser le système malgré le retard :

- Diminuer les gains G_f et G_p.
- Augmenter Z_m.
- Diminuer Z_e.
- Diminuer la bande passante de D(p).

I.9 Structure à deux canaux Force-position :

Contrairement à la structure position-position qui ne requiert pas de capteur d'efforts, la structure force-position nécessite au moins la mesure de l'effort d'interaction entre l'esclave et l'environnement, pour que celui-ci soit restitué à l'utilisateur, par l'intermédiaire du manipulateur maître. Il existe de nombreuses variantes à cette structure [Hannaford89, Lawrence93], selon que les efforts sont mesurées coté esclave seulement ou

des deux coté.la version originale présentée dans [Raju89], n'utilise que les efforts mesurées coté esclave. L'utilisation de la mesure d'effort coté maître permet d'accroître les performances du système, et d'assurer, si nécessaire, une meilleure réversibilité du manipulateur maître.

I.10 Conclusion :

Dans ce chapitre j'ai présenté les concepts de base pour l'étude et la commande bilatérale d'un système de télémanipulation avec retour d'efforts.

L'objectif d'une télémanipulation avec retour d'efforts est de fournir à l'utilisateur un système lui permettant de réaliser une tâche à distance tout en ayant l'impression de la réaliser directement. Pour cela nous avons introduit la notion de transparence liée aux performances générales du système. Bien que la transparence soit un élément essentiel dans ce type de système, il est nécessaire de garantir aussi la stabilité quelle que soit l'utilisation ou l'interaction. Pour cela, nous avons introduit la notion de passivité, qui se réfère à l'énergie dissipée par le système.

II.1 Introduction :

L'abréviation VHDL signifier VHSIC Hardware Description Langage (VHSIC : Very High Speed Integrated Circuit). Ce langage a été écrit durant les années 70, par le département de la défense américaine destiné à modéliser les circuits intégrés complexes.

Au début, ce langage était uniquement destiné à décrire les circuits intégrés déjà conçus et devrait permettre de réaliser des documentations techniques facilement interprétables par certaines personnes.

Aujourd'hui, la finalité de ce langage a bien changée, puisque il est essentiellement utilisé à concevoir et modéliser les circuits, non plus dans un but de documentation, mais de simulation. Car on la étendu en lui rajoutant des extensions pour permettre la conception (synthèse) de circuits logiques programmables (P.L.D. Programmable Logic Device).

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

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

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

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

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

II.2 Historique :[1]

Au début des années 80, le département de la défense américaine (DOD) désire un système de description formelle et standard des circuits dans le cadre de son programme VHSIC, car à cette époque, les fournisseurs du DOD avaient chacun son propre HDL ce qui limitait l'échange des designs. C'est pourquoi, le DOD a décidé de définir un langage de spécification. Il a ainsi mandaté des sociétés pour établir un langage. Parmi ces langages proposés, le DOD a retenu le langage VHDL qui fut ensuite normalisé par IEEE (Institute of Electrical and Electronics Engineers) en vue de satisfaire les objectifs suivants :

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

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

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

En 2001 nouvelle révision (VHDL 2001 ou IEEE 1076 – 2001)

En 2006 nouvelle version (VHDL 2006 ou IEEE 1076 – 2006)

II.3 Définition du langage VHDL :[2 ,3]

Le VHDL est un langage de description matériel destiné à représenter le comportement ainsi que l'architecture d'un système électronique indépendamment d'un fournisseur d'outils.

Ainsi, techniquement, il est incontournable car c'est un langage puissant, moderne et qui permet une excellente lisibilité, une haute modularité et une meilleure productivité des descriptions. Il permet de mettre en œuvre les nouvelles méthodes de conception.

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

II.4 La structure d'une description VHDL :[2]

La description VHDL est composée de 2 parties qui sont indissociables :

- L'entité (ENTITY)
- L'architecture (ARCHITECTURE).

II.4.1 Déclaration des bibliothèques : [4 , 5]

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

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

```
Library ieee.std_logique_1164.all ;
Use ieee.std.numeric_std.all;
Use ieee.std_logique_unsigned.all;
```

II.4.2 Déclaration de l'entité: [4 ,6]

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

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

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

La syntaxe générale de l'entité :

```
Entity Nom_de_l'entité is
        Port (nom_entrée_1 : in type_du_signal
        Nom_entrée_2: in type_du_signal;
        ….
        Nom sortie_1 : out type_du_signal ;
        Nom_E_S_1 : inout type_du_signal
    ) ;
End Nom_de_1_entite ;
```

Remarque :

Il ne faut jamais maitre un point virgule après la dernière définition de signal de l'instruction port mais il faut la maitre après la fermeture de la parenthèse de cette instruction.

II.4.2.1 Le signal d'entré/sortie :

Un signal est défini par son :

- Nom:
- Mode: in : pour un signal en entrée.

Out : pour un signal en sortie.

Inout : ; peur un signal entrée sortie

Buffer: pour un signal en sortie mais utilisé en comme entré dans la description.

- Type :

Std_logique : pour un signal.

Std_logic_vector(x down to x0): pour un bus qui est compose de plusieurs signaux, avec x correspond au MSB et x0 correspond au LSB.

Remarque :

Les signaux std_logic peuvent prendre différentes valeur qui sont :

- '1' ou 'H' : pour le niveau haut.
- '0' ou 'L' : pour le niveau bas.
- 'Z' : pour l'état de haut impédance.
- '_' : quelconque, c.à.d. n'importe quel valeur.

II.4.3 Les architectures :[2 ,1]

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

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

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

En VHDL la boîte noire est nommé entité (entity) car l'entité doit toujours être associée avec au moins une description de son contenu, de son implémentation qui est l'architecture. Une architecture est un ensemble de processus qui s'exécutent en parallèle.

Une architecture fait toujours référence à une entité. Elle est définie par un nom, que l'on pourra choisir pour expliciter la façon dont en code.

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

Ensuite, viens le mot clé «begin». A sa suite, on trouvera le code qui décrit le fonctionnement de 1'architecture.

La description d'une architecture peut prendre trois formes :

- Comportementale
- Structurelle
- Mixte

La description quelle soit structurelle, comportementale ou mixte se fait de la manière suivante :

```
Architecture description of nom_entité is
                    {Partie déclarative}
Begin
                    {Partie descriptive}
End description:
```

II.4.3.1 Description comportementale : [2 ,1]

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

ARCHITECTURE comportementale of circuit is

- Partie déclarative.

BEGIN

- Partie descriptive.

END comportementale

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

a. Sous forme de flow de données(DATAFALOW):

Dans ce type de description comportementale DATAFALOW, on modélise le circuit par un ensemble d'équations logiques et arithmétiques, car elle consiste à décrire chaque

sortie par une équation en fonction des entrées.

Example:

```
ARCHITECTURE XOR of circuit is
BEGIN
SI <= IN 1 XOR IN 2;
32 <= IN1 XÔR ÎN3;
END XOR;
```

b. Sous forme d'instruction séquentielle:

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

La structure alternative :

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

La structure d'aiguillage :

```
CASE <identification> IS
WHEN choix=séq_stat;
END CASE;
```

Remarque:

Séq_stat désigne une ou plusieurs déclarations séquentielles.

II.4.3.2 Description structurelle : [2 ,1]

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

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

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

- **Déclaration des composants utilisés:**

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

Elle se fait de la manière suivante :

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

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

Chaque connexion est définie de la manière suivante :

```
Camp 1 : composant port map (….) ;
```

Où :

Comp1 : représente l'affectation du mapping.

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

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

II.4.3.3 Description Mixte : [2 ,1]

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

Cette dernière est spécifiée par le mécanisme de package.

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

II.5 Les instructions concurrentes et séquentielles :[5 ,6]

II.5.1 Les instructions concurrentes :

Programme concurrent: le programme dans lequel les instructions s'exécutent de manière simultanée. Il n'y a aucun ordre d'exécution de ces événements. VHDL est un programme à exécution concurrente, de telle sorte que pour une description VHDL toutes les instructions sont évaluées et affectent les signaux de sortie en même temps. L'ordre dans lequel elles sont écrites n'a aucune importance. En effet la description génère des

structures électroniques, c'est la grande différence entre une description VHDL et un langage informatique classique.

Dans un système à microprocesseur, les instructions sont exécutées les une après les autres.

Avec VHDL il faut essayer de penser a la structure qui va être générée par le synthétiseur pour écrire une bonne description cela n'est pas toujours évident.

II.5.1.1 L'affectation simple :

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

Exemple avec des portes logiques : S1 <= E2 and El ;

II.5.1.2 L'affectation conditionnelle :

Cette instruction modifie l'état d'un signal suivant le résultat d'une condition logique entre un ou des signaux, valeurs constantes.

SIGNAL <= expression when condition
[Else expression when condition}
[Else expression];

Remarque :

L'instruction [else expression] n'est pas obligatoire mais elle est fortement conseillée, elle permet de définir la valeur du SIGNAL dans le cas où la Condition n'est pas remplie.

On peut mettre en cascade cette instruction.

II.5.1.3 L'affectation sélective :

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

With SIGNAL_DE_SELECTIDN select
SIGNAL <= expression when valeur_de_seleetion,
[Expression when valeur_de_selection,]
[Expression when others];

Remarque:

L'instruction [expression when others] n'est pas obligatoire mais fortement conseillée, elle permet de définir la valeur du SIGNAL dans le cas où la condition n'est pas remplie.

Example N°1:

```
-- Multiplexeur 4 vers 1
With SEL select
        S2 <= El when “00",
E2 when "01",
E3 when "10”,
EA when "11",
‘ 0 ‘ when others;
```

Remarque:

When others est nécessaire car il faut toujours définir les autres cas du signal de sélection pour prendre en compte toutes les valeurs possibles de celui—ci.

En conclusion, les descriptions précédentes donnent le même schéma, ce qui est rassurant. L'étude des deux instructions montre toute la puissance du
langage VHDL pour décrire un circuit électronique, en effet si on avait été obligé d'écrire les équations avec des opérateurs de base pour chaque sortie, on aurait en les instructions suivantes :

S2 <= (El and not (SEL (l)) and not (SEL (0))) or (E2 and not SEL (1) and (SEL (0)) or (E3 and SEL (1) and not (SEL (0))) or (E4 and SEL (1) and SEL (0));

L'équation logique ci-dessus donne aussi le même schéma, mais elle est peu Compréhensible, c'est pourquoi on préfère des descriptions de plus haut niveau en utilisant les instructions VHDL évoluées.

II.5.1.4 Les opérateurs :

a) Opérateur de concaténation : &

Cet opérateur permet de joindre des signaux entre eux.

Exemple :

```
-- Soit A et B de type 3 bits et S1 de type 8 bits
-- A= "001" et B ="110"
```

```
S1 <=A&B&"0l":
-- S1 prendra la valeur suivante après cette affectation
-- Sl="00111001"
```

b) Opérateurs logique :

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

Exemples :

```
S1 <= A sll 2 ; --S1 = A décalé de 2 bits à gauche
S2 <= A rol 3 ; -- S2 = A avec une rotation de 3 bits à gauche
S3<= not (R) ; --S3=R
```

c) Opérateurs arithmétiques :

Opérateur	**VHDL**
ADDITION	+
SOUSTRACTION	-
MULTIPLICATION	*
DIVISION	/

Remarque N°1 :

Pour pouvoir utiliser les opérateurs ci-dessus il faut rajouter les bibliothèques suivantes au début du fichier VHDL :

```
Use ieee.numeric_std.all ;
```

```
Use ieee.std_logic_arith.all ;
```

Exemples:

```
S1<= A -3 ; --S1 =A -3
-- On soustrait 3 à la valeur de l'entrée / signal A
S1 <= S1 + 1 ; -- On incrémente de 1 le signal S1
```

Remarque N°2 :

Attention l'utilisation de ces opérateurs avec des signaux comportant un nombre de bits important peut générer de grandes structures électroniques.

Exemples :

```
S1 <= A * B ; -- S1 = A multiplié par B : A et B sont codés sur 4 bits
S2 <= A / B ; -- S2 = A divisé par B : A et B sont codés sur 4 bits
```

d) Opérateurs relationnels :

Ils permettent de modifier l'état d'un signal ou de signaux suivant le résultat d'un test ou d'une condition. En logique combinatoire ils sont souvent utilisés avec les instructions :

- when … else…
- with … Select…

Voir ci-dessous :

Opérateur	**VHDL**
Egal	=
Non égal	/=
Inférieur	<
Inférieur ou égal	<=
Supérieur	>
Supérieur ou égal	>=

II.5.2 Les instructions séquentielles:

Programme séquentiel : programme se déroulant de manière séquentielle. Toutes les opérations ont lieu les unes après les autres de manière ordrée (comme dans un programme informatique par exemple, ADA, C, C++. Pascal,...)

Système séquentiel : système numérique dans lequel les valeurs des sorties sont déterminées par des informations précédemment mémorisées dans des bascules, et par les valeurs “actuelles” (aux temps de propagation près) des entrées. Les informations à mémoriser sont enregistrées dans des bascules en synchronisme avec un signal d’horloge. Un événement à une entrée ne se répercute pas forcément de façon “immédiate” (aux temps de propagation près) sur les sorties. Il n’influencera les informations mémorisées que lors du prochain “coup” d’horloge.

Les instructions séquentielles doivent être utilisées uniquement dans une zone de description séquentielle. Nous utiliserons principalement ces instructions à l’intérieur de l’instruction process. Ces instructions peuvent aussi être utilisées dans des procédures et des fonctions.

II.5.2.1 L’affectation simple:

La syntaxe générique d‘une affectation simple est la suivante:

```
Signal_l <= Signal_2 fonction_logique Signal_3 ;
```

Remarque:

L’affectation ne modifie pas la valeur actuel du signal. Celle—ci modifie les valeurs futures. Le temps n’évolue pas lors de l’évaluation d’un process.

L’affectation sera donc effective uniquement lors de l’endormissement du process.

L’instruction when … else n‘est pas utilisable à l‘intérieur d’un process. C‘est une instruction concurrente. L’instruction séquentielle correspondante est l‘instruction conditionnelle (if then else).

II.5.2.2 L'instruction conditionnelle:

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

```
If Condition_Booléenne_l then
--Zone pour instructions séquentielles
elsif Condition_Booléenne_2 then
--Zone pour instructions séquentielles
elsifCondition_Booléenne_3 then
…
```

```
else
--Zone pour instructions séquentielles
end if ;
```

Nous reprenons la description du détecteur de priorité (3 entrées et une sortie sur 2 bits) pour démontrer l'utilisation de l'instruction conditionnelle.

```
Architecture Comport of Detect_Priorite is
Begin
Process (Entrecl,ËntreeZ, Entree3)
Begin
Priorite <= "00"; -- Valeur par defaut
if(Entree3 = '1') then
Priorite <= "11";
elsif (Entree2 = '1') then
Priorite <= "10";
Elsif (Entree1= '1') then
Priorite <= "01";
--else pas nécessaire, déjà valeur par défaut
--Priorite <= "00";
end if;
end process ;
end Comport;
```

II.5.2.3 L'instruction de choix:

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

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

Il est possible de définir plusieurs valeurs de la manière suivante:

when Val_3 | Val_4] Val_5 => --Zone pour instructions séquentielles

Si le type utilisé pour l'expression est ordonné (exemple: Integer, Natu-ral..), il est

possible de définir des plages:

```
when 0 to 7 => --Zone pour instructions séquentielles
ou
when 31 downto 16 => --Zone pour instructions séquentielles
```

Il est important de noter que le type Std_Logic_Vector n‘est pas un ensemble de valeurs ordonnées. C‘est un type discret. Il est donc impossible d’utiliser les plages to et downto.

Nous allons montrer l‘utilisation de l’instruction case avec la description d‘un démultiplexeur 1 à 4. Ce-circuit dispose d‘une entrée de sélection Sel de 2 bits, une entrée à commuter Val_In et de 4 sorties.

```
Architecture Comport of DMUX1_4 is
    begin
    process (Sel, Val_in)
    begin

        Y0 <= '0'; --Valeur par défaut
        Y1 <= '0'; --Valeur par défaut
        Y2 <= '0'; --Valeur par défaut
        Y3 <= '0'; --Valeur par défaut
    Case Sel is
        When “00” => Y0 <= Val_In;
        When “01” => Y1 <= Val_In;
        When “10” => Y2 <= Val_In;
        When “11” => Y3 <= Val_In;
        When others => Y0 <= ‘X’;--pour simulation
                       Y1 <= ‘X’;--pour simulation
                       Y2 <= ‘X’;--pour simulation
                       Y3 <= ‘X’;--pour simulation
    end case ;
    end process ;
    end Comport:
```

Remarque:

Le cas « others » n‘est jamais utilisé pour les combinaisons logiques de Sel.

Mais le type « Std_Logique » comprend 9 états ('X', 'U' , 'H', etc.). Le vecteur Sel de 2 bits comprend donc 81 combinaisons et pas seulement 4! Il est indispensable de prévoir ces combinaisons dans la description. Lors de la simulation si Sel est différent d'une des 4 combinaisons logiques ("00", "01", "10","11"), les sorties sont affectées avec l'état 'X' pour indiquer qu'il y'a un problème dans le fonctionnement du démultiplexeur.

II.6 Définition de process : [4]

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

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

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

Exemple en utilisant le process :

```
Entity MUX is
        Port ( A, B, SEL : in std_logic;
        OP : out std_logic
        );
End MUX;
Architecture TEST of MUX is
        Begin
        MUXPROCESS: process (A, B, SEL)
        Begin
If(SEL =—- '1') then
        OP <=A;
Else
        OP <=B;
End if;
    End process MUXPROCESS;
End TEST;
```

II.7 Les fonctions et procédures : [3 ,6]

II.7.1 Rôle, principe et fonctionnement :

Le langage VHDL permet l'utilisation et la création de fonctions ou de procédures que l'on peut appeler à partir d'une architecture. Le rôle de ces fonctions ou procédures est de permettre, au sein d'une description, la création d'outils dédiés à certaines tâches pour un type déterminé de signaux (voir chapitre suivant concernant les types de signaux). Cette notion d'objets est typique aux langages évolués de programmation.

Une fonction reçoit donc des paramètres d'entrées et renvoie un paramètre de sortie.

Dans l'exemple précédent, la fonction reçoit trois paramètres A, B et C et retourne un paramètre S. Le mot clé RETURN permet d'associer au paramètre de sortie une valeur. Une fonction peut donc contenir plusieurs RETURN,

II.7.2 La déclaration des fonctions et procédures :

Dans les exemples qui précèdent, les fonctions ou procédures ont été insérées dans des architectures. Or, dans ce cas précis, les fonctions ou procédures en question ne sont accessibles que dans l'architecture dans laquelle elles ont été décrites. Pour les rendre accessibles à partir de plusieurs architectures, il est nécessaire de les déclarer au sein de packages et de déclarer en amont de l'architecture vouloir utiliser ce package. En reprenant l'exemple de la création d'une fonction NONET et en insérant cette fonction au sein d'un package, voici ce que l'on obtient :

II.8 Les attributs : [7 ,2]

Les attributs sont des propriétés ou des caractéristiques associées à un objet ou à un type. Ils se divisent en deux grandes catégories ; les attributs prédéfinis et les attributs définis par l'utilisateur.

La désignation d'un attribut fait intervenir un préfixe et un suffixe séparés par une apostrophe.

La syntaxe de déclaration de ces attributs est comme suit :

Nom_var' nom_attribut ;

II.9 Paquetage : [7 ,2]

Pour ce langage, il existe une unité de conception dont le rôle est de permettre le regroupement de données, de variables, de fonctions et de procédés. Cette unité est une sorte de bibliothèque d'objets (constantes, variables, fonctions ou procédures ..... etc.)

que l'on pourra appeler à partir de plusieurs descriptions. Cette unité est particulièrement intéressante lorsqu'on utilise au sein d'un projet contenant plusieurs entités de ressources identiques.

Un paquetage est constitué de deux parties principales :

- La première est la partie de déclaration ; elle contient principalement des déclarations de types, de constantes, de composants et de sous- programmes destinés à être utilisés par d'autres unités de conception.

Package identificateur ls

Déclaration de types, de fonctions, de composants, d'attributs.

End Package ;

- Le corps d'un paquetage contient le corps des sous-programmes déclarés dans la partie visible et des déclarations locales qui n'ont pas vocation à être visibles de l'extérieur.

Package body identificateur is

Corps des sous programme déclarés.

End Package body;

II.10 Les différences avec un langage de programmation : [1]

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

Avec ce langage, un système est structuré en composants qui fonctionne en parallèle et en permanence; Par contre, un autre langage de programmation est structuré en sous—programme qui a une durée d'exécution limité dans le temps avec un début en une fin. Il

existe aussi une différence qui concerne le support de l'information: car dans un système matériel les composants sont interconnectés avec des signaux qui sont des connexions permanentes et concurrentes, des sorties de canaux par lesquels transitent les informations la notion de signal est la base de la description matérielle des structures de donnés qui sont le support de l'information en transit dans le système, car un signal est affecté en permanence. par contre pour les autres langages de programmation les variables supportent l'information et elles sont affectées de manière ponctuelle dans le temps.

II.11 Les avantages du langage VHDL : [7]

Le langage VHDL offre deux avantages majeurs en plus de la simplicité lors de la conception:

- **Simulation:**

Le but de la modélisation, c'est la simulation. Il existe deux points importants pour la simulation :

La fidélité: UN modèle doit être aussi précis que possible dans son champ d'application.

L'efficacité: le modèle doit pouvoir être simulé le plus rapidement possible et doit être portable, réutilisable et facile à maintenir.

Pour simuler un modèle, il faut disposer du modèle, bien sûr, mais aussi de stimuler, c'est-à-dire de la description des signaux d'entrées du modèle au cours du temps. Ces stimuler sont appelés les TESTSBENCHES en Anglais.

Les langages fonctionnels de description de matériel permettent de simplifier la conception en analysant automatiquement les résultats en cours de simulation.

Un environnement de simulation complet comprend : un générateur de stimuli, un modèle de l'objet à simuler et un vérificateur automatique des résultats. Ces trois composants sont modélisés à l'aide du même langage.

- **Synthèse :**

Les langages fonctionnels de description matérielle servent aussi à concevoir. Il ne s'agit plus de modéliser en vue de la simulation, mais de décrire les objets qui seront véritablement fabriqués. Si les considérations de vitesse d'exécution en simulation existent toujours (la description sera simulée avant d'alimenter le synthétiseur, afin de vérifier que la fonction décrite est bien la fonction désirée) elles ne sont plus prioritaires. Ce qui compte le plus, c'est l'efficacité du code au sens du synthétiseur. En effet, pour que ce dernier produise la description structurelle la plus économique passible (et donc la surface

de silicium la plus petite possible).

II.12 Environnement ISE : [7]

L'environnement ISE est un logiciel de programmation de la famille XILINX. Il est défini comme étant un environnement de développement de systèmes numériques ayant pour objectif une implantation matérielle sur FPGA de la famille XILINX.

ISE intègre différents outils permettant de passer à travers tout le flot de conception d'un Système numérique. En effet il dispose :

- D'éditeur de textes, de schémas et de diagramme d'états
- D'un compilateur VHDL
- D'un outil de simulation
- D'outils pour la gestion des contraintes temporelles.
- D'outils pour la synthèse
- D'outils pour la vérification
- D'outils pour l'implantation sur FPGA

Les étapes d'implémentation d'un circuit numérique à l'aide de cet environnement sont présentées dans l'annexe.

II.13 Conclusion :

Dans ce deuxième chapitre on a pu faire une présentation détaillée du langage VHDL, ainsi qu'une petite comparaison avec les langages informatiques, et ces avantages, et pour enfin conclure par un aperçu sur le logiciel de développement ISE qui est l'outil pour l'implantation sur FPGA.

III.1 Introduction

Les FPGA se situent entre les réseaux logiques programmables et les circuits logiques près diffusés.

1- **Les réseaux logiques programmables** : sont des composants qui ne nécessitent aucune étape technologique supplémentaire pour être personnalisés, ce sont des circuits standards, programmables par l'utilisateur grâce aux différents outils de développement.

2- **Les prés diffusés :** sont des circuits intégrés basés sur l'utilisation des réseaux de cellules dont les blocs sont préalablement diffusés. Il faut créer les connexions entre ces blocs.

Les FPGA combinent donc à la fois la **souplesse** de la programmation des réseaux logiques programmables et **les performances** des circuits prés diffuses. En d'autres termes le FPGA permet d'avoir une architecture conçue sur mesure à haute densité dans un circuit électronique, et avec la possibilité de modifier cette architecture quand de nouvelles applications apparaissent.

Le langage HDL comme le VHDL ou le Verilog sont utilisés pour décrire les fonctionnalités qui seront implémentées sur le composant. La description matérielle est traduite dans un fichier de configuration pour le FPGA cible [8].

La technologie FPGA actuellement disponible permet de construire des systèmes qui sont configurables ou reconfigurables. L'idée commune à tous les circuits configurables est de proposer, non pas une fonction figée dont le champ d'application est le plus large possible, mais une structure adaptable. La reconfiguration consiste à modifier, en cours de fonctionnement, l'architecture matérielle. Cela permet de concevoir des circuits intelligents qui peuvent s'auto-adapter à leur environnement.

Les circuits FPGA sont programmables ou reprogrammables sur les cartes implantées par l'utilisateur. Cette reconfiguration est une propriété nécessaire face aux systèmes à charges et contraintes variables. Les dernières technologies qui sont utilisées ont permis la satisfaction des besoins forts en densité. Selon l'architecture des FPGA, chaque point d'interconnexion peut être réalisé selon deux technologies qui définissent deux classes différentes des FPGA à savoir **La technologie de programmation SRAM e**t la technologie de programmation **à base d'inti-fusibl.** La technologie de programmation SRAM permet d'avoir une reconfiguration rapide des FPGA. Les points de connexions sont des ensembles des transistors commandés. Quant la technologie de programmation **à base d'inti-fusible**, les points d'interconnexions sont de type ROM. c'est-à-dire la modification du point est irréversible [9 ,10]

L'avantage des circuits programmables FPGA est de pouvoir être reprogrammable, Contrairement aux circuits intégrés de type ASIC. Ce qui rend cette solution modulable et donne la possibilité de modifier le programme générique de base afin de le rendre spécifique au circuit utilisé. Ces circuits programmables permettent aussi d'obtenir les meilleurs performances car la technologie FPGA utilise du parallélisme matériel, qui offre une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP). BDTI, une importante société d'analyse et de « *benchmarking* », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application [11,12].

Actuellement, on trouve sur le marché des circuits FPGA de faible, moyenne et haute densité produits grâce aux deux principaux producteurs de circuits logiques programmables :

Xilinx et Altera. Sur le même marché, on trouve plusieurs autres producteurs de circuits FPGA, on peut dont : Actel, Abound Logic, Achronix, Atmel, Cypress, Lattice Semiconductor, etc. L'apparition nombreuses des FPGA sur le matché mondial ont révolutionné certains domaines de contrôle numérique et de plus en plus utilisés pour intégrer des architectures numériques complexe. Ils sont devenus les plus populaires en matière d'implantation et de prototypage des circuits numérique après leur apparition sur le marché en 1984. La clé maitresse de leurs réussites est l'espace de programmation de ces derniers. Leurs utilisations actuelles couvrent les deux domaines : civil et militaire. Parmi ces applications nous citons [9] :

Informatique : périphérique spécialisés

Machinerie industrielle : contrôle des machines

Télécommunication : traitement d'images et filtrage **Instrumentation :** équipement médical, prototypage **Transport :** contrôle des avions et métrons **Aérospatiale :** les satellites

Militaire : radar, communication protégée, la détection de la surveillance Et autre **….**

III.2 Définition des FPGA

Un FPGA (Field-Programmable Gate Array) est un dispositif semi-conducteur contenant des composants à logique programmable (Blocs logiques) et des interconnexions programmables (routage programmable). Les blocs logiques peuvent exécuter n'importe quelle fonctionnalité depuis les fonctions logiques de base (comme : AND, OR, XOR, NOT) jusqu'aux fonctions complexes combinatoires. Les FPGAs comprennent aussi des éléments de mémoire (bascules, blocs complets de mémoire). Quelque soit la fonction logique, elle peut être réalisée par l'interconnexion des blocs logiques. La principale caractéristique des FPGAs est la reconfiguration qui peut être même effectuée même en cours de fonctionnement.On parlera alors de reconfiguration dynamique[13].

III.3 Architecture interne d'un FPGA [14]

Les circuits FPGA sont structurés sous forme de matrices d'éléments logiques de base, constitués de portes logiques qui présentent physiquement sur le circuit. Les circuits FPGA du fabricant Xilinx utilisent deux types de cellules de base, les cellules d'entrées/sorties appelées IOB et les cellules logiques appelées CLB. Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions configurable. Donc les trois blocs principaux sont les blocs logiques configurables, les blocs d'entrées/sorties et les ressources de communications. Il existe aussi des circuits FPGA qui, contiennent des microprocesseurs. La **figure (III.01)** nous montre l'architecture interne des cartes FPGA.

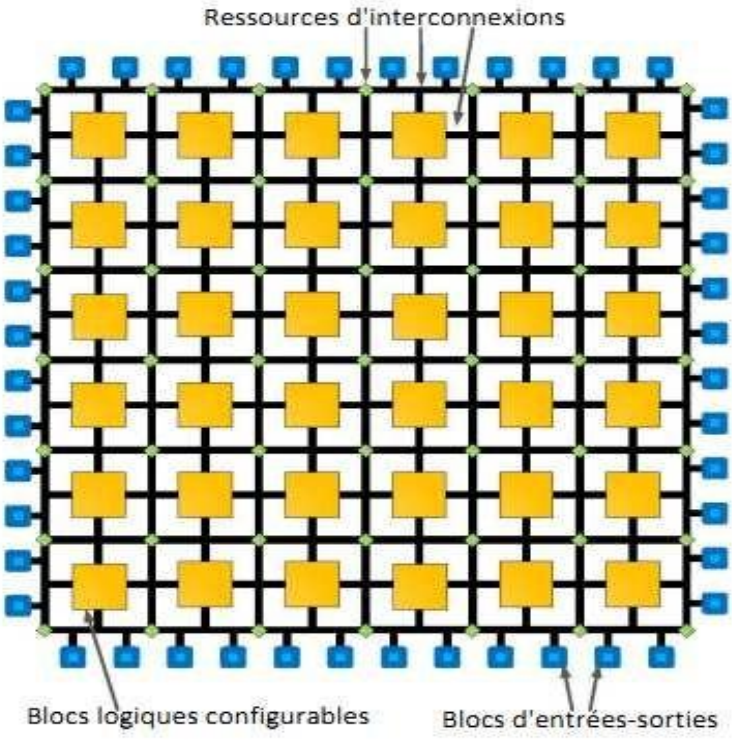


Figure (III.01) : Architecture interne d'un PFGA [14]

III.3.1 Les blocs logiques configurables (CLB)

Les blocs logiques configurables sont les principaux éléments d'un FPGA. Ils peuvent avoir un ou plusieurs générateurs de fonctions réalisées avec des tables de correspondance (LUT - look-up tables) qui peuvent mettre en œuvre une logique arbitraire en fonction de leur configuration. Autour d'une LUT, il y a une logique d'interconnexion qui permet les liaisons à destination et vers la LUT.

Elle est mise en œuvre à l'aide de portes logiques et de multiplexeurs. Pendant le processus de configuration d'un FPGA, les mémoires des LUT sont écrites pour y implémenter une fonction, et la logique qui l'entoure est configurée pour router correctement les signaux afin de construire un système complexe. Il existe différents types de CLB en fonction du FPGA utilisé. Pour Xilinx le bloc logique configurable est appelé slice. L'architecture
du slice (avec des modifications mineures) est utilisée dans toutes les familles FPGA Virtex. La **figure(III.02)** nous Présente l'architecture interne d'un CLB de vertex 5 qui contient quatre Slices :

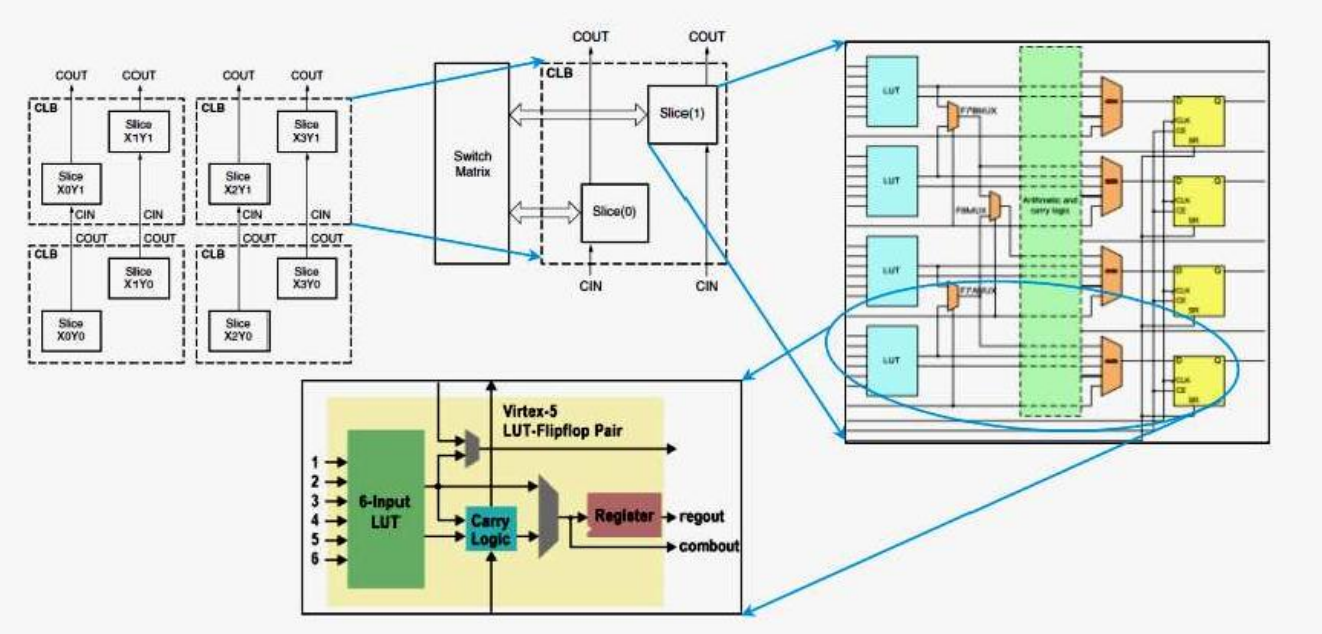


Figure (III.02). Architectures interne d'un CLBs dans les FPGAs Virtex5

III.3.2 Blocs d'entrée-sortie (IOB)

Les blocs d'entrée-sortie permettent l'interconnexion de la logique interne aux ports d'entrées et de sorties du FPGA. Les IOB ont leur propre mémoire de configuration, elle stocke les standards de tension et la direction des ports. Ces blocs sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé (haute impédance).

III.3.3 Ressources d'interconnexions

Les ressources d'interconnexion au sein d'un FPGA, permettent la connexion arbitraire des CLB et des IOB. Les connexions internes dans les circuits FPGA sont composées de segments

métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, à l'aide des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Pour parvenir à cet objectif, Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons comme le montre la **Figure (III.03).**

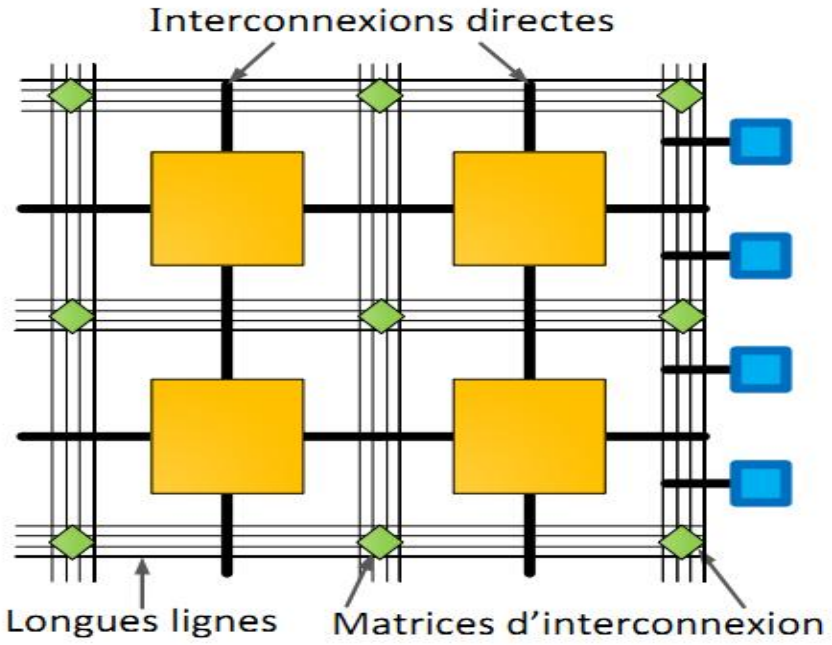


Figure (III.03): Structure générale du routage

- **Interconnexions directs**

Ces interconnexions permettent l'établissement des liaisons entre les CLB et les IOB. Il est possible aussi de connecter directement certaines entrées d'un CLB aux sorties d'un autre.

- **Longues lignes**

Ce sont de longs segments métallisés parcourant toute la longueur et la largeur du FPGA, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion

- **Les matrices d'interconnexions**

Ce sont des aiguilleurs situés à chaque intersection. Leur rôle est de raccorder les longues lignes entre elles selon diverses configurations. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre CLB sur le FPGA, selon plusieurs possibilités comme le montre la figure suivante

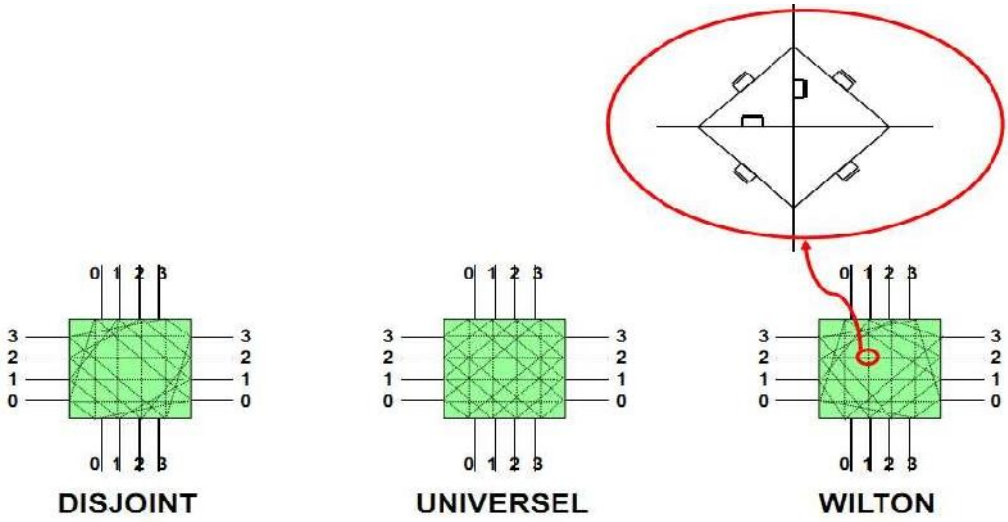


Figure (III.04): La matrice de commutation [15]

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

III.4 FPGA de la famille XLINX

Xilinx est une entreprise américaine de semi-conducteurs créé en 1984. Inventeur du FPGA avec un premier produit en 1985, Xilinx fait partie des plus grandes entreprises spécialisées dans le développement et la commercialisation de composants logiques programmables, et des services associés tels que les logiciels de CAO électroniques.

Xilinx fabrique une large gamme de FPGA pour diverses applications. En effet, l'offre commerciale de Xilinx est découpée en plusieurs gammes : (FPGA hautes performances : gamme Virtex, FPGA pour la fabrication en grande série : gamme Spartan. Les plus onéreux sont les FPGA Virtex (Virtex II/pro, Virtex4, Virtex5, Virtex6, Virtex7). En 2010, Xilinx a lancé les FPGA-Virtex6, en technologie 40nm, avec 3 catégories différentes (LX, LXT et SXT). Même si les Virtex6 ne disposent pas de processeur PowerPC405, la dernière catégorie est optimisée, spécialement, pour les applications lourdes en traitement numérique du signal. Ensuite,

en 2012, Xilinx a lancé les FPGA-Virtex7 (avec une seule catégorieXC7VxxxT) fabriqués en technologie 28nm. Afin de satisfaire à terme les applications très exigeantes en performances de calcul et en bande passante tout en limitant la puissance consommée, Xilinx va proposer dans le courant du second semestre 2012 des circuits qui couplent plusieurs FPGA Virtex7 au sein d'un même boîtier via une technologie d'interconnexion 3D. Selon Xilinx, le boîtier Virtex7 LX2000T en technologie 28 nm sera le premier FPGA multi-puce au monde et affichera une densité de portes logiques 3,5 fois supérieure au FPGA 40 nm le plus puissant actuellement disponible.

Le tableau (II.02) organise cette famille (virtex) selon l'année de fabrication ainsi que la technologie utilisée et nombre de cellules de chaque une d'elles.

Famille	**Année**	**Technologie**	**Max cellules**
Virtex 4	2004	90 nm	200 000 LUT 4
Virtex 5	2007	65 nm	330 000 LUT 5
Virtex 6	2010	40 nm	760 000 LUT 6
Virtex 7	2012	28 nm	2 000 000 LUT 7

Tableau (III.01): la famille vertex de Xilinx [15]

Les composants Virtex5 sont disponibles, à bon prix, avec plusieurs catégories (LX, LXT, SXT, TXT et FXT). Dans notre travail nous utilisons la carte FPGA de la famille Vitex5 et de catégorie LX avec la plateforme ML501.

III.5 Présentation et description des éléments de FPGA ML 501

Une plateforme d'évaluation FPGA est un circuit imprimé contenant un circuit FPGA, un processeur (éventuellement), des périphériques d'E/S permettant l'interaction avec l'environnement externe. Ces plateformes permettent non seulement de tester de façon rapide et efficace des prototypes de test mais peuvent aussi être utilisées comme solution finale après validation. Pour cela, elles sont équipées de mémoire ROM leurs permettant de garder de façon définitive la solution finale.

Nous utilisons la plateforme d'évaluation XC5VLX50-1FFG676. C'est à dire le cœur de la plateforme ML501 est un circuit FPGA de type Virtex5 XC5VLX50-1FFG676, dont les caractéristiques sont données dans le **tableau (III.01).** La carte ML501 est une plateforme riche en composants pour l'évaluation et le développement des multiples applications. Ces composants permettent un accès facile et pratique aux ressources disponibles dans le circuit FPGA [80]. Les ports de la vidéo, de l'audio et les ports de communication ainsi que les ressources mémoires généreuses donnent la fonctionnalité et la flexibilité à cette plateforme et la rendent une plateforme de développement FPGA typique. La plateforme ML501 dispose donc de :
[13, 22,23]

- **Mémoires** : Xilinx Platform Flash XCF32P ; 9Mb ZBT synchronous SRAM ; 32MB Intel P30 Strata Flash ; 2MB SPI Flash ; System ACE™ Compact Flash configuration controller avec un connecteur Compact Flash ; IIC-EEPROM de 8Kbits.

- **Connectivité** : Port JTAG utilisé avec un câble parallèle (III) ; câble parallèle (IV) ou bien plateforme de téléchargement USB ; Ethernet PHY 10/100/1000 tri-speed ; Codec Stéréo AC97 avec connecteurs line-in/microphone, line-out/headphone, et audio difital SPDIF ; Piezo Speaker ; Port PS/2 pour souri/clavier ; des E/S d'extension ; Port série type RS-232 ; Connecteur vidéo (DVI/VGA).

- **Composants divers** : Générateur de système d'horloge programmable ; CPLD type XC95144XL de Xilinx ; support d'oscillateur d'horloge (3.3V) avec un oscillateur à 100MHz ; contrôleur de température et de tension ; boutons-poussoirs à usage général ; afficheur LCD de 16 caractères sur 2 lignes ; entrée d'alimentation ; LED d'indication (d'alimentation, d'initialisation et d'activation) ainsi que plusieurs switchers et ports d'E/S.

La figure (III.05) nous montre le placement des ces composants et autres dans cette carte.

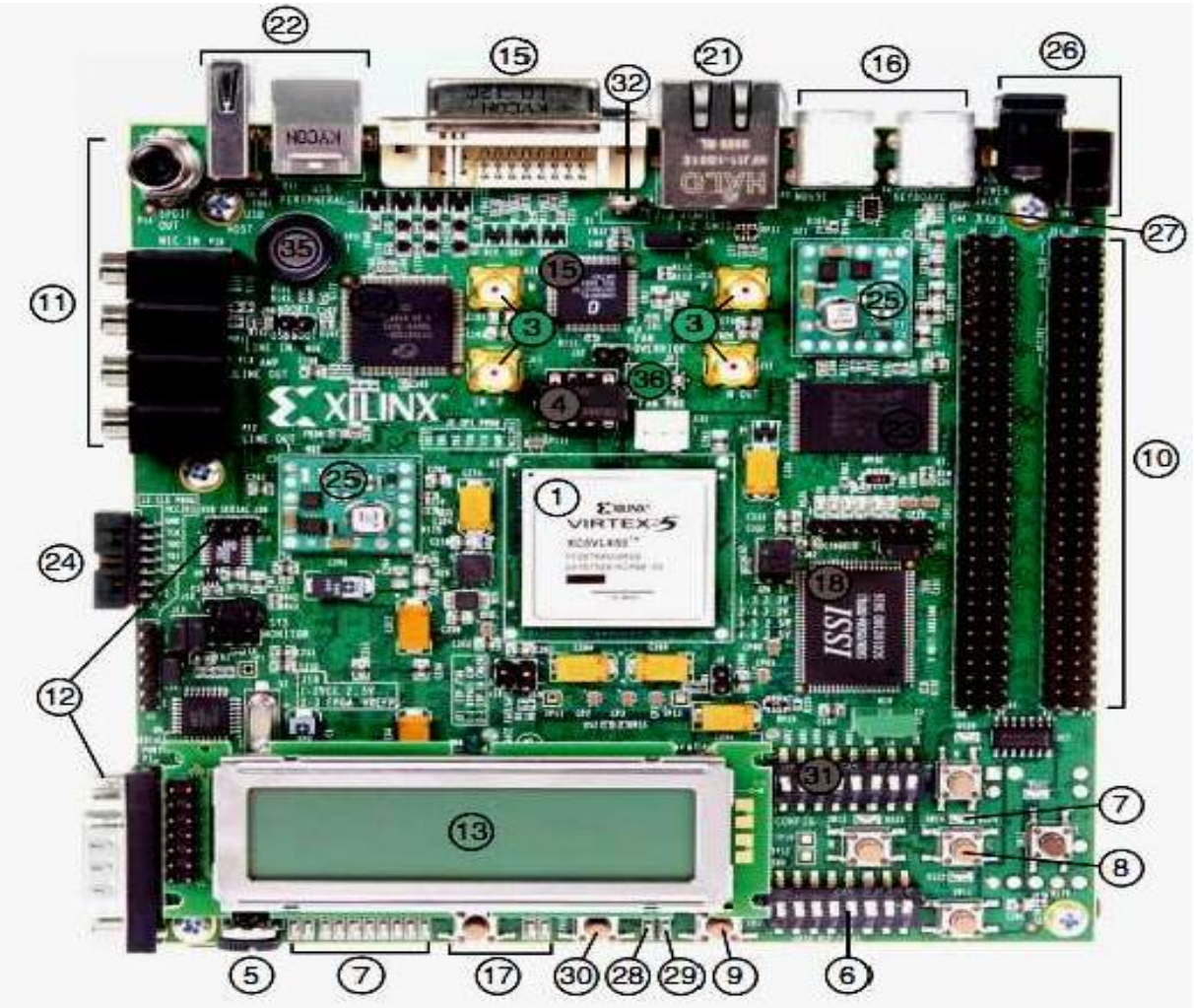


Figure (III.05). La carte Virtex5-ML501 de Xilinx [22].

Dans notre cas nous programmons notre carte FPGA à partir d'un ordinateur en utilisant un câble JTAG avec l'application appropriée IMPACT montré en **figure (III.06**).

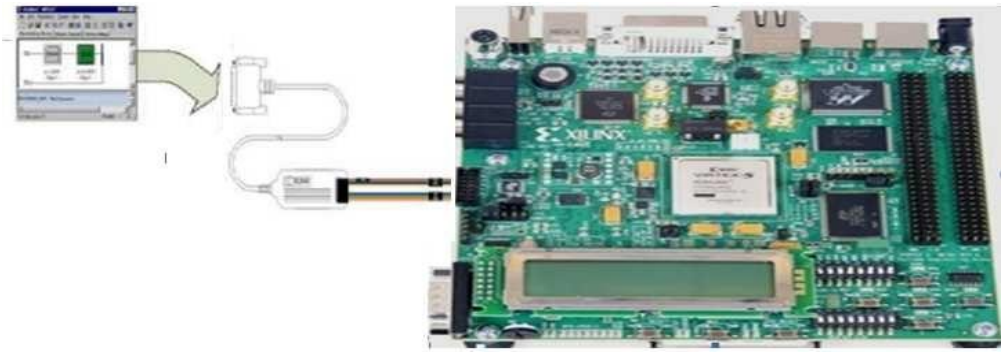


Figure (III.06). Programmation d'un FPGA avec un câble JTAG en utilisa l'applicationIMPACT**.[21]**

Conclusion

Ce chapitre est dédié à la présentation des circuits programmable FPGA (Field programmable Gate Array). Nous avons présenté les d'FPGA de la famille XILINX, et plus particulièrement la famille Vertex 5. Nous avons présenté l'architecture externe de la carte FPGA Vertex5 du cœur ML501. En plus dans ce chapitre noua avons présenté l'architecture interne des circuits programmables FPGA pour pouvoir les configurer.

IV.1 Introduction:

D'une manière générale, un système de télé-opération se décompose en deux parties qui coopèrent : l'une est dite **opérative (esclave)** l'autre est dite **commande (maître).**

Pour réaliser ses systèmes il existe plusieurs interfaces permettent de répondre aux critères

comme les arduino , les microcontrôleurs, les FPGA…….etc

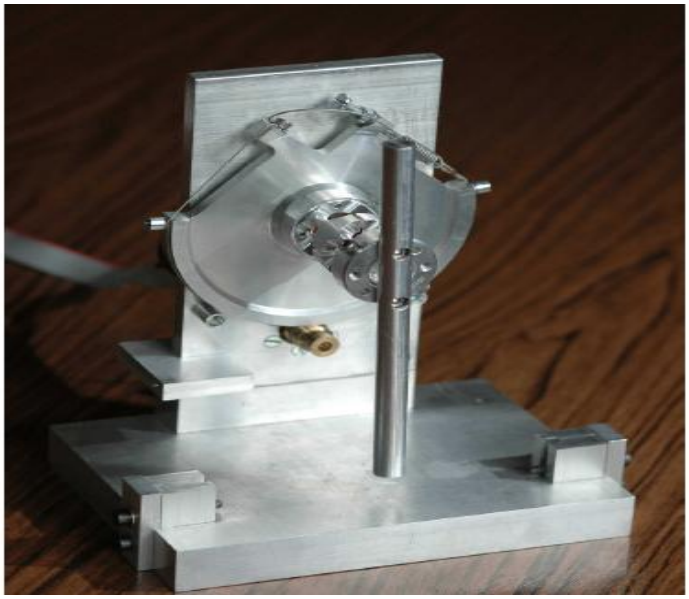
Mais les FPGA repondent mieux aux critères exigés par les systèmes de télé-opération.

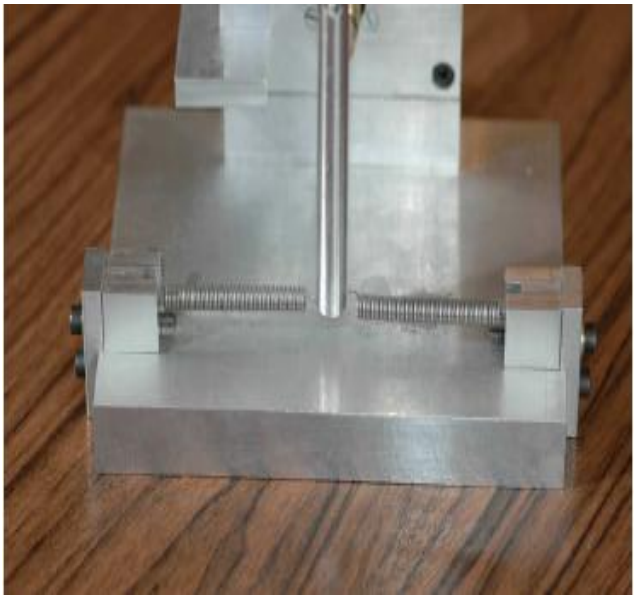
IV.2 description du système :

Notre système est constituer de deux partie (deux moteur), l'une maitre et l'autre partie c'est l'esclave et d'une interface de communication ou de commande la carte FPGA.

Ce travaille consiste a élaborer une commande force-position pour cette télé-opération pour réaliser une tache à distance.

la figure(IV.1) montre l'image du maitre et de l'esclave :

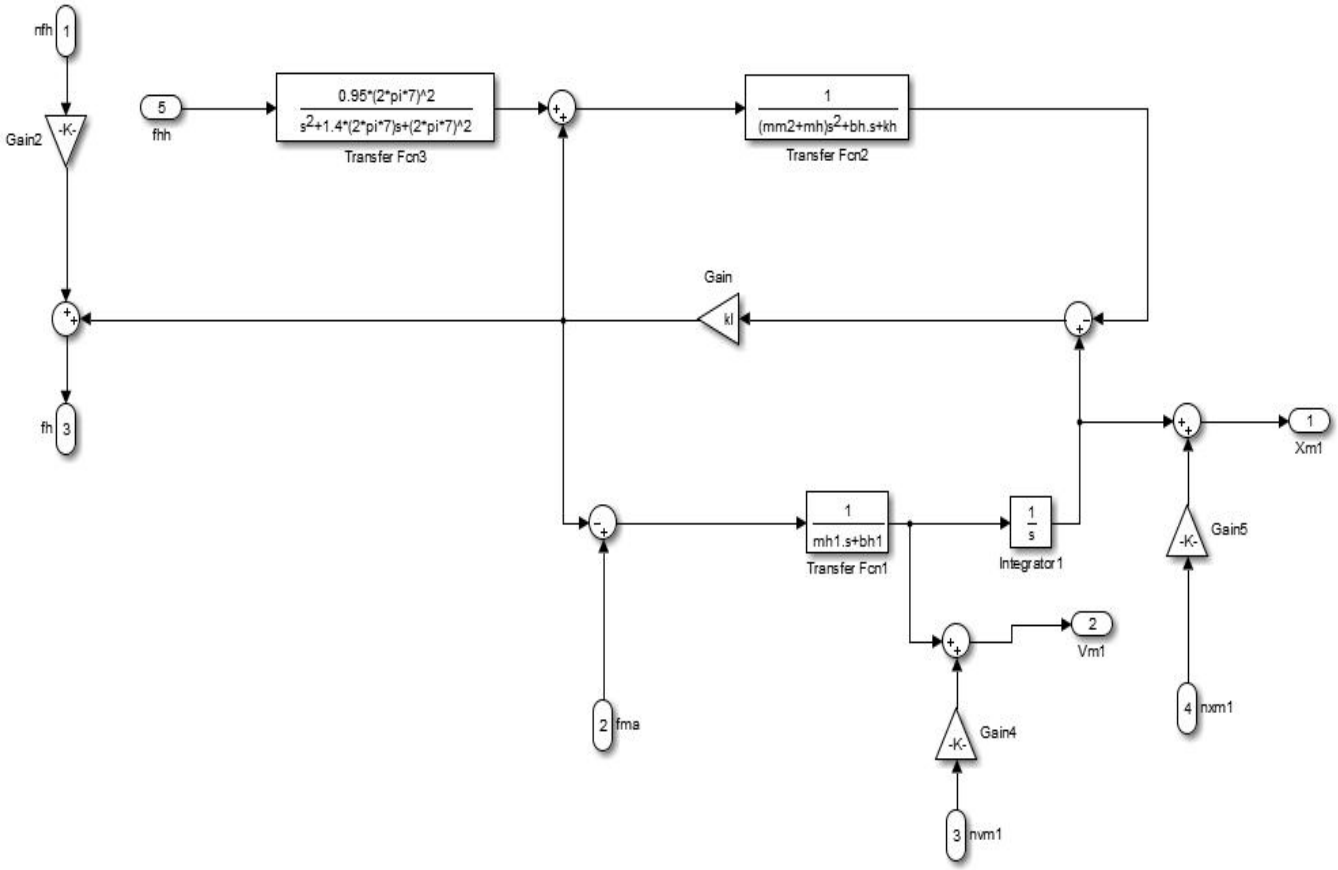




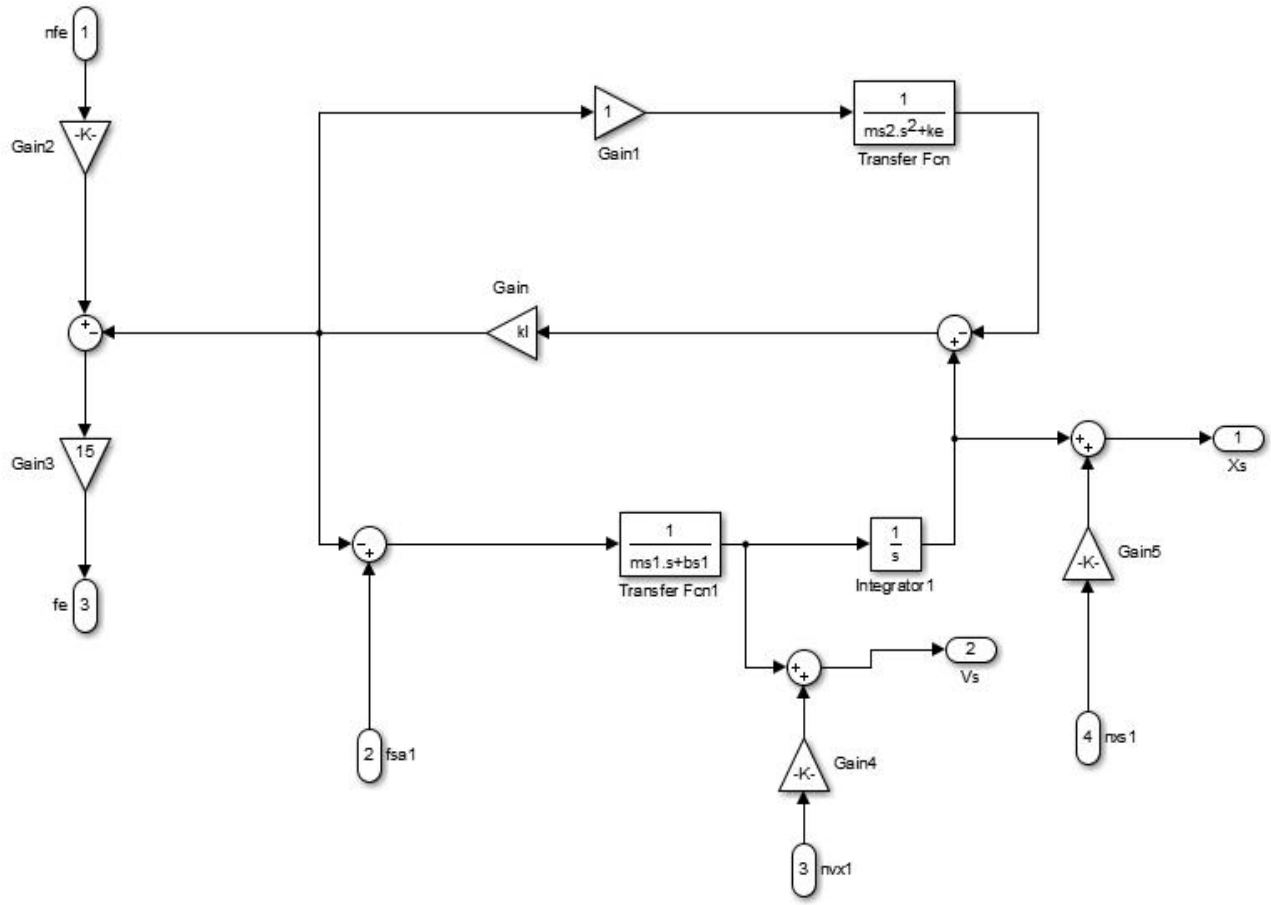
Figure(IV.1) : image du système de télé opération haptique à 1 ddl .

IV.3 modélisation du système :

Pour modéliser notre système il suffit de représenter les deux blocs, maitre et esclave et on a aussi l'operateur humain et l'environnement de notre esclave.



Figure(IV.2) : bloc du système maitre.



Figure(IV.3) : bloc système esclave

Avec les données suivantes :

ke=0.9025;wke=1;kh=5.37;wkh=0.15;bh=0.0465;wbh=0.15;

mh=0.0013;mh1=0.145;bh1=5.15;

mm1=4.25e-4;ms1=4.25e-4;bm1=6.95e-4;bs1=6.95e-4;mm2=9.3e-5;ms2=9.3e-5;kl=125;

wuh=0.0015;wue=0.0015;wxm=7e-4;wxs=7e-4;wfrics=0.04;wfricm=0.04;

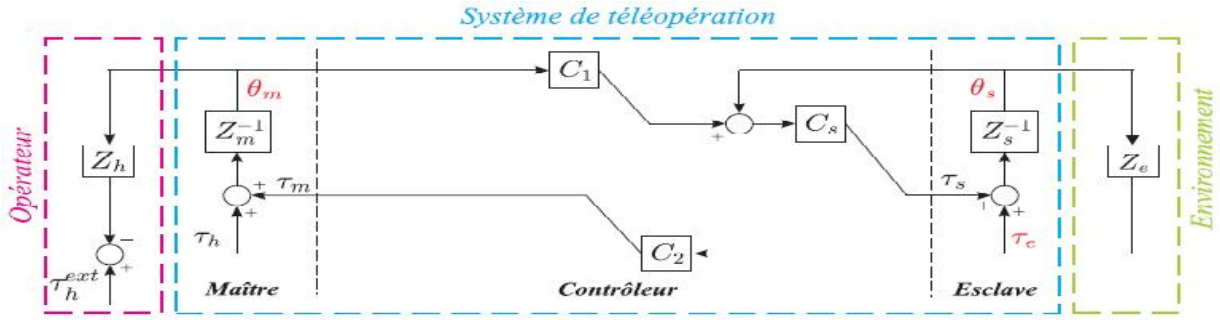
En relions ses deux systèmes avec deux régulateurs classique (PID) qui sont notre commande, on réalise notre architecture de télé-opération.

IV.4 élaboration de la commande :

La méthode de contrôle Force-Position, également appelée contrôle par impédance ou en anglais Direct Force Feedback, est la méthode la plus naturelle à imaginer pour un retour de force. En effet, elle est basée sur la mesure de l'interaction entre l'esclave et l'environnement par un capteur de force, dans lesquelles la force d'interaction est calculée

par la simulation et renvoyée comme consigne vers le maître.

Nous avons le schéma de contrôle ci-dessous :



Figure(IV.4): Contrôle Force-Position (FP)

La position de l'utilisateur θ_m est mesurée par l'encodeur du maître et envoyée comme consigne vers l'esclave, à travers le canal de communication C_1. La différence avec la mesure de l'encodeur du robot θ_s est alors injectée dans le contrôleur de position C_s pour calculer la commande motrice T_s. En même temps, l'interaction entre l'esclave et son environnement T_e, mesurée par le capteur de force, est renvoyée comme consigne vers le maître, à travers le canal de communicationC_2, pour générer le retour d'effort T_m.

Les commandes des moteurs sont donc définies par :

$$\left.\begin{matrix} \mathsf{T}_m = C_2\, T_e \\ \mathsf{T}_s = C_s(C_1\theta_m - \theta_s) \end{matrix}\right\}$$

Avec $C_s = (k_d^s s + k_p^s)$ la fonction de transfert du contrôleur de position de type proportionnel/dérivé .les blocs C_1 et C_2 représentent respectivement le transfert de position du maître vers l'esclave et le transfert de force de l'esclave vers le maître.ils modélisent les éventuels gains d'amplification (position) et $\boldsymbol{\beta}$(force),et les délais dans les communications $\Delta 1$ et $\Delta 2$. La transformé de Laplace d'un délai d'une durée Δ est donnée par $e^{-s\Delta}$, ce qui permet d'écrire :

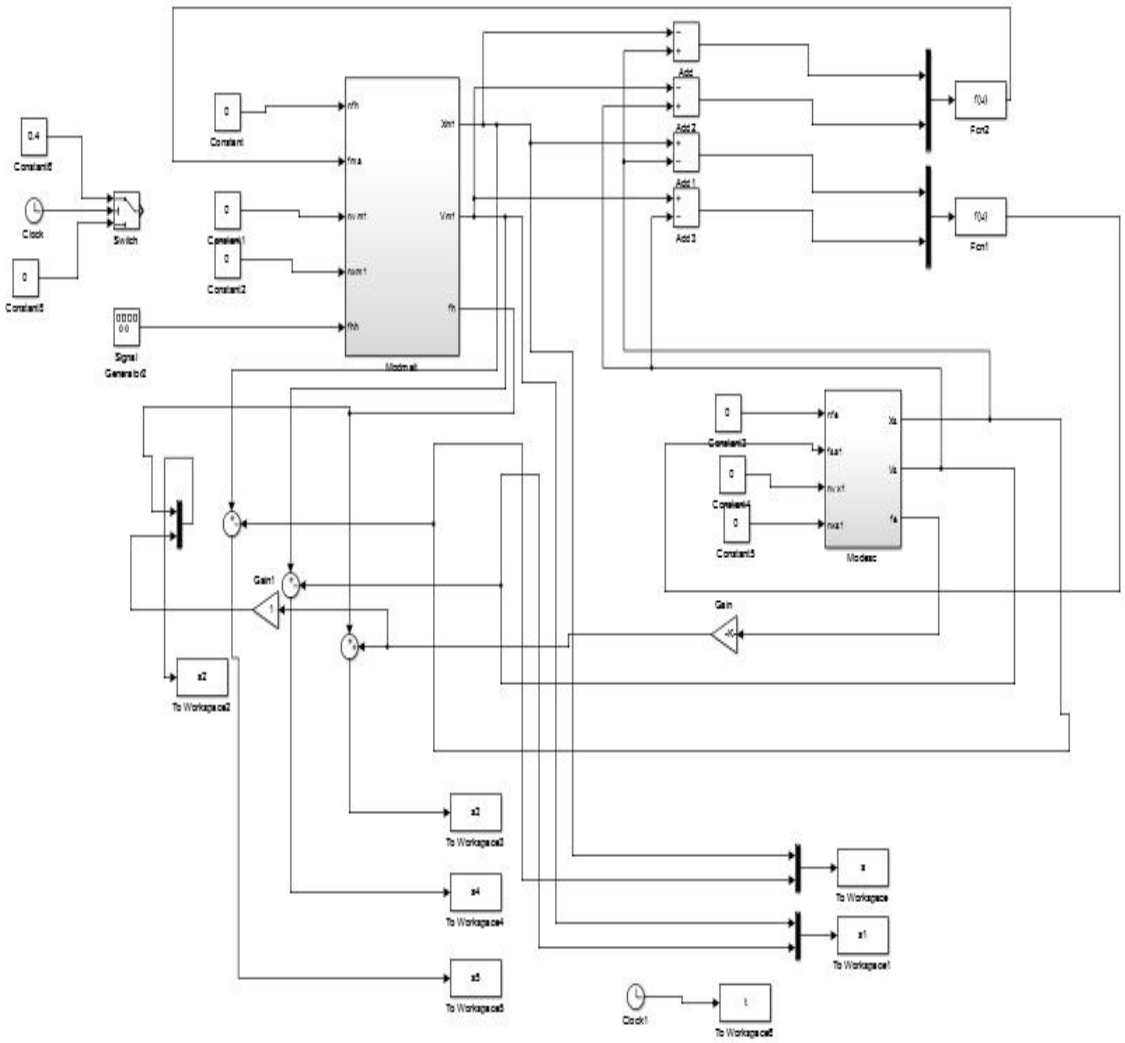
$$C1 = \alpha e^{-s\Delta 1} \qquad\qquad C_2 = \beta e^{-s\Delta 2}$$

IV.5 résultat de simulation avec SIMULINK

IV.5.1 simulation de l'architecture sur matlab SIMULINK :

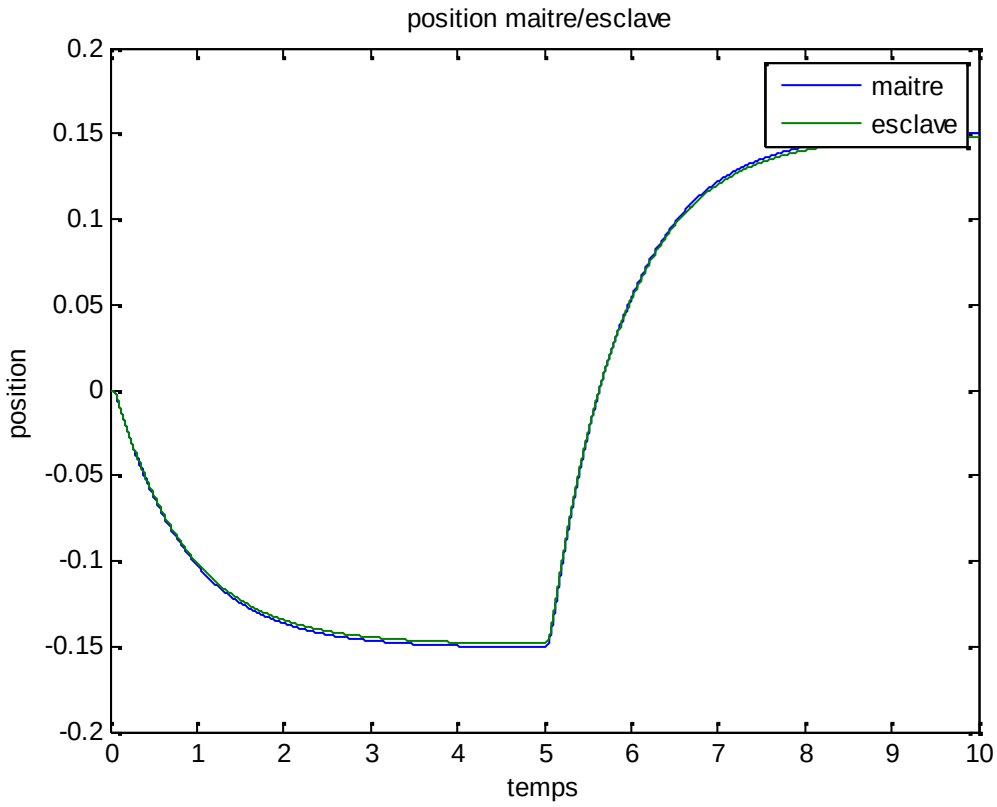
La figure(IV.5) représente le schéma de simulation de notre architecture de télé-opération .

Avec des régulateurs PID comme commande.

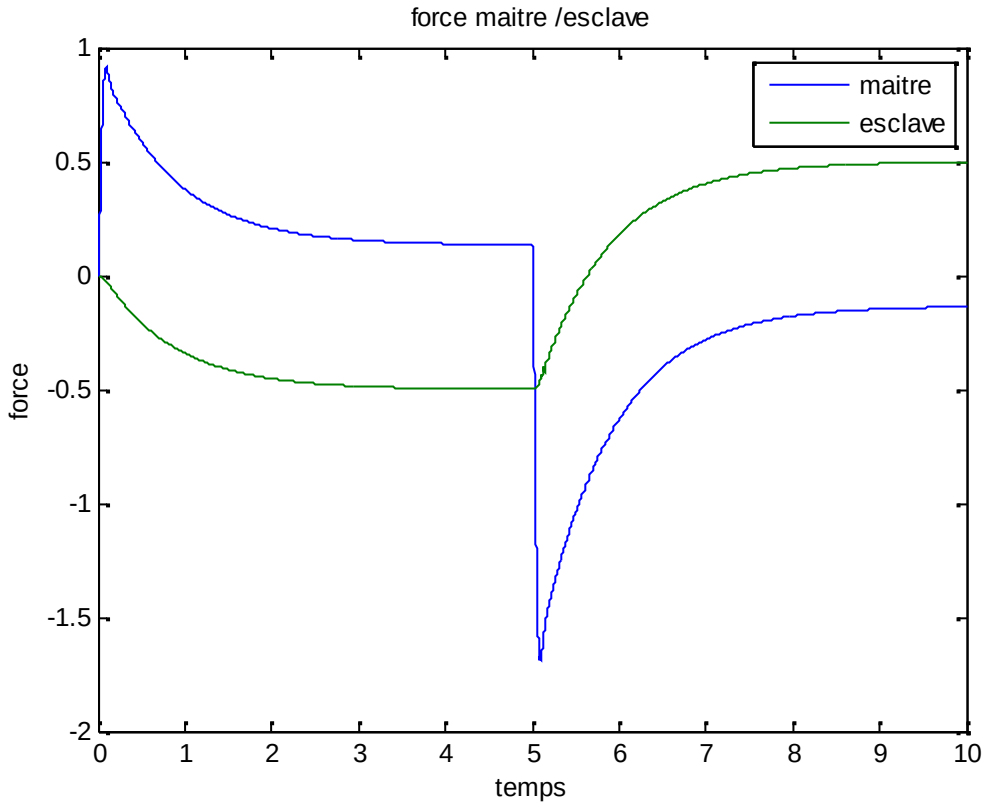


Figure(IV.5) : schéma de simulation maitre/esclave

Après la simulation on obtient les graphes (graph de position et graph de force) dans les figures suivants :



Figure(IV.6) : position maitre/esclave4



Figure(IV.7) : force maitre/esclave

- **Interprétations des résultats :**

D'après les graphes, notre commande avec les régulateurs PID est presque parfaite en position car elle nous offre une erreur de position qui tend vers 0 entre la position du maitre et de sont esclave.

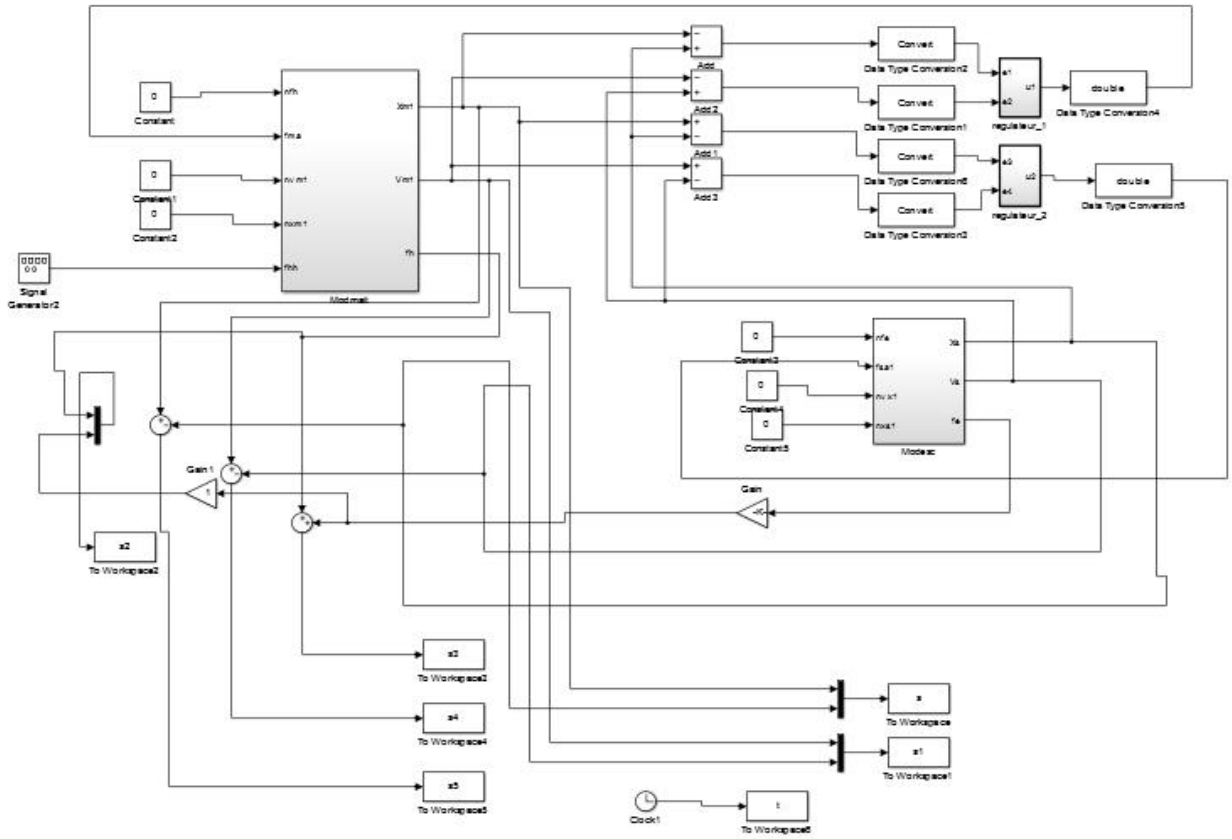
Par contre en force on voie une erreur de force lisible. Mais le maitre suit toujours la trajectoire de l'esclave en force.

IV.6 Conversion du matlab au VHDL :

Pour convertir un schéma du matlab SIMULINK au VHDL il suffit de numériser tout ce qui est calculateur et dans notre cas on va numériser les deux régulateurs .

IV.6.1 numérisation des régulateurs :

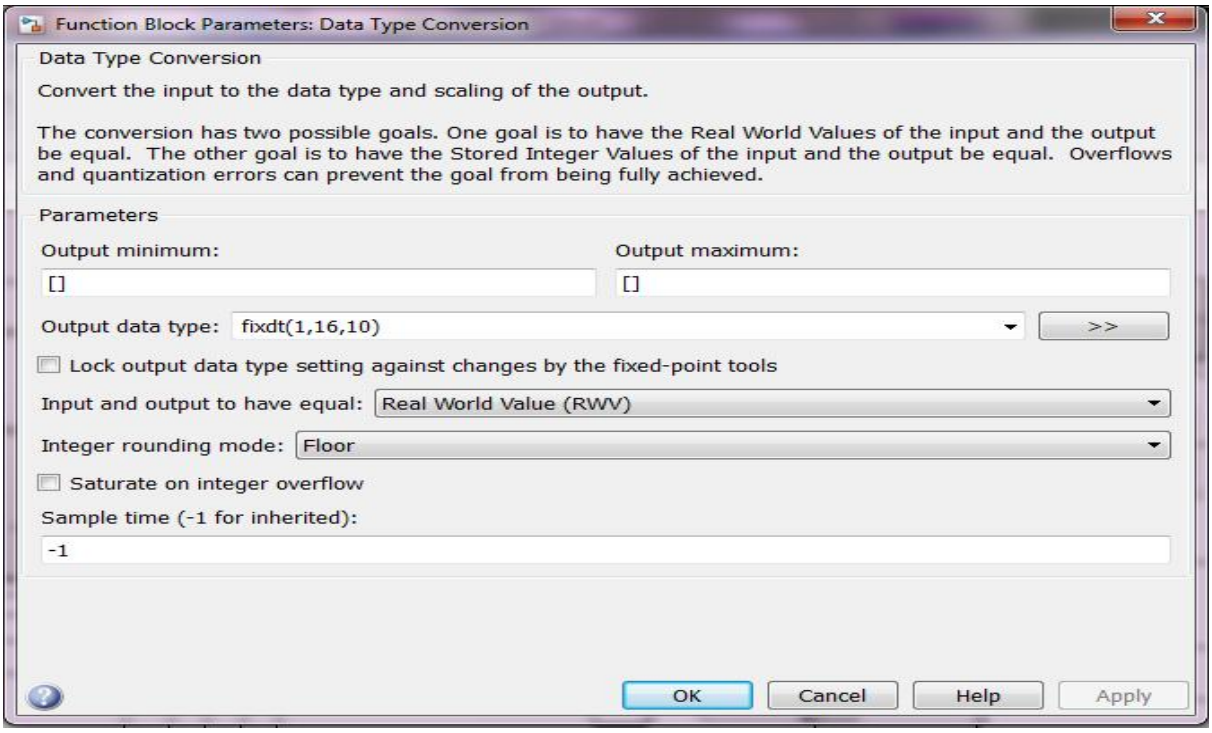
En insère des convertisseurs analogique numérique (CAN) aux entrées des régulateurs et des convertisseurs numérique vers l'analogique aux sorties des régulateurs on obtient le schéma de simulation suivant :



Figure(IV.8) : schéma de simulation maitre/esclave avec convertisseurs

Puis en configure les convertisseurs d'entrées que leurs entrées est en réel et leurs sorties en binaire,

Et en configure aussi ses convertisseurs de la sortie des régulateurs que leurs entrées est en binaire et leurs sorties en réel. Tout cela en cliquons deux fois sur chacun des convertisseurs.



Figure(IV.9) : fenêtre de réglage des convertisseurs

Enfin, pour que les régulateurs fonctionne on numérique on doit les configurer selon les étapes suivantes :

- première étape : en click sur le bouton droite de la souris sur chacun des régulateurs la fenêtre

Suivante vas-nous accueillir :

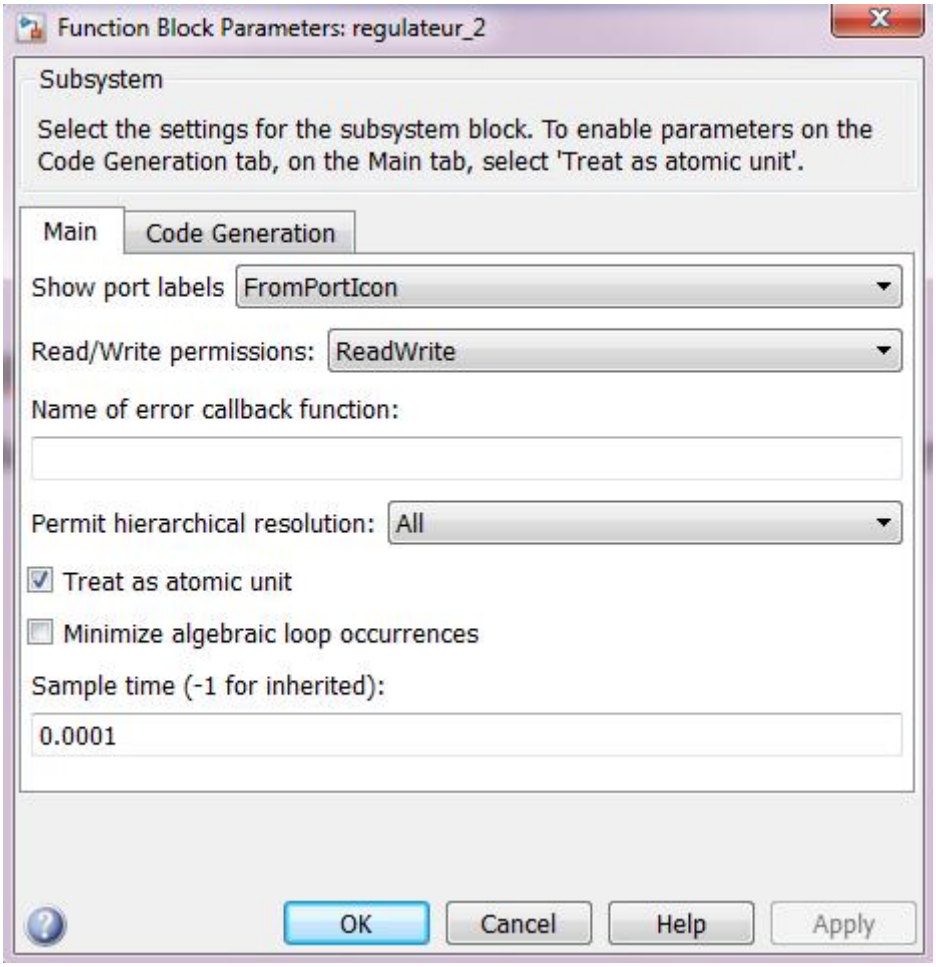
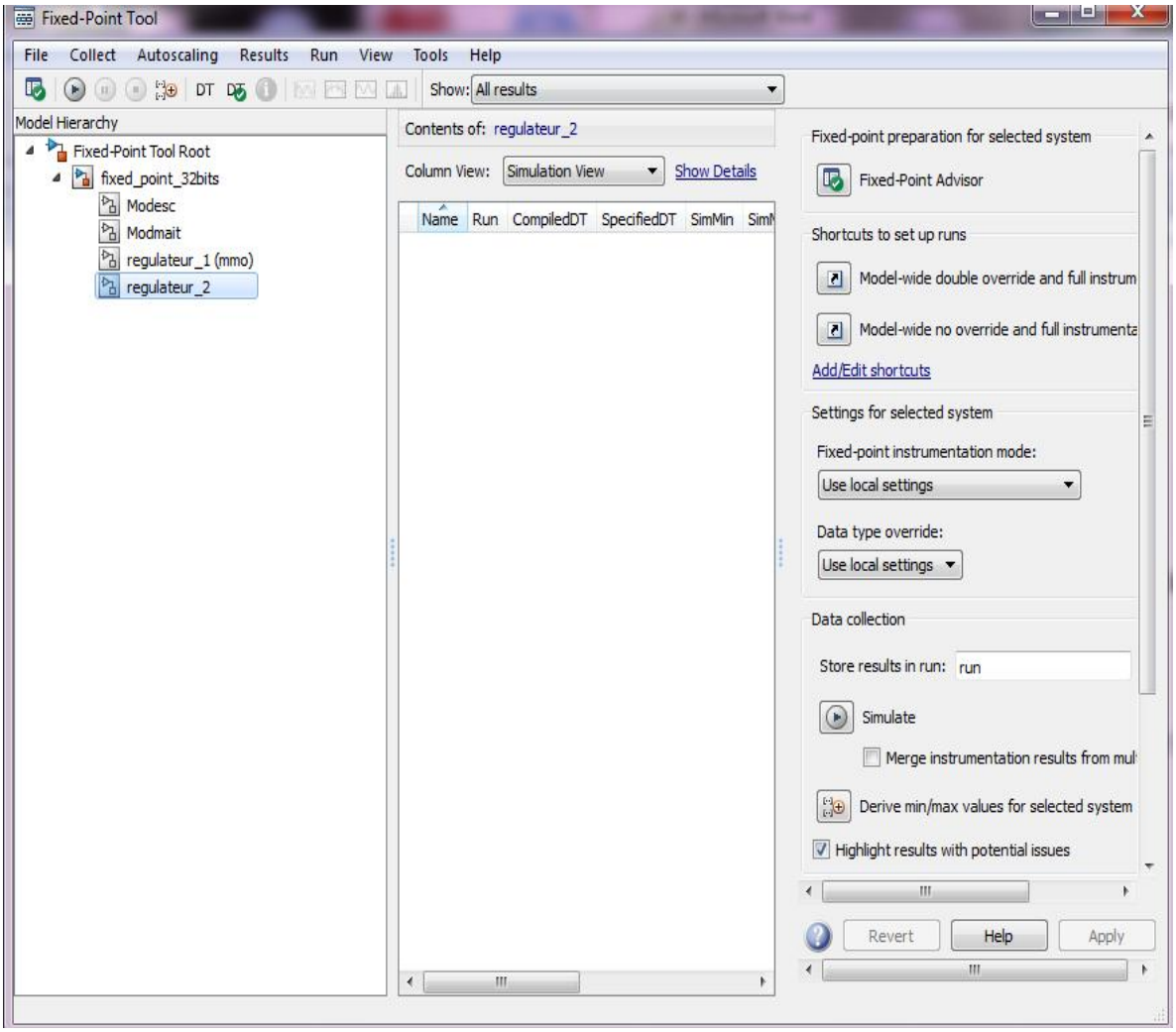


Figure (IV.10) : fenêtre de réglage de la période d'échantillonnage

En vas choisir la période la plus petite possible comme dans cette fenêtre (0.0001).

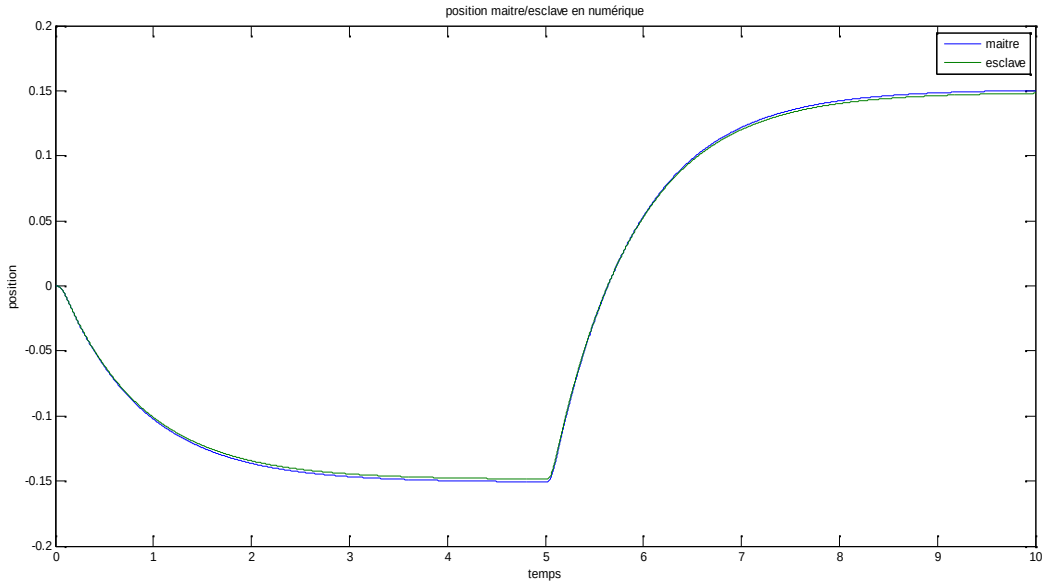
-deuxième étape : en click toujours sur le bouton droite de la souris sur le régulateur et on va choisir

« Fixed-poit tool… » en auras

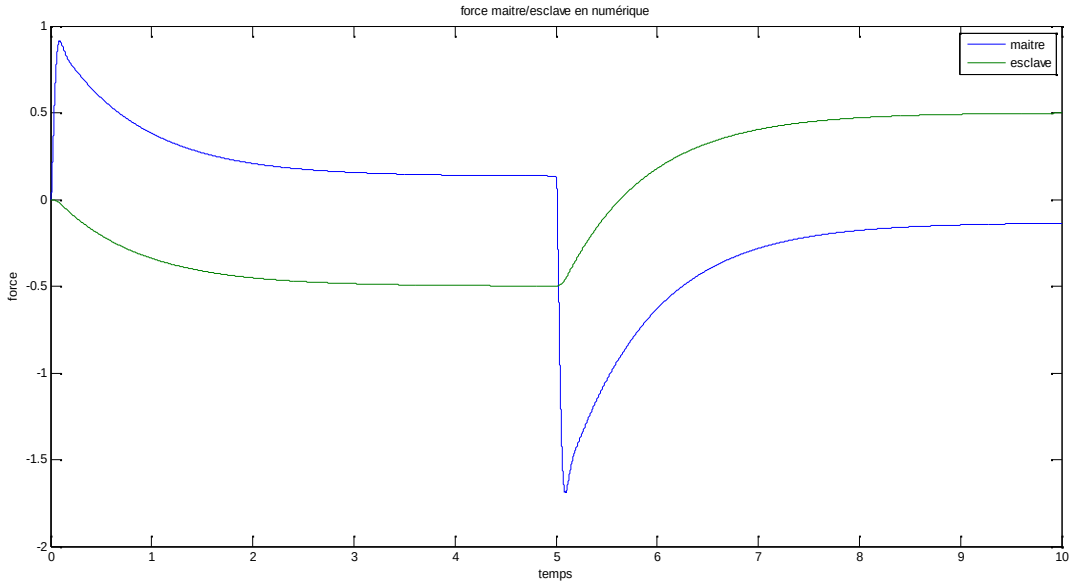


Figure(IV.11) : fenêtre de calcule des points du signal discrets

En simule tout les étapes qui se trouve sur cette fenêtre et en auras notre système de commande de notre architecture de télé-opérations avec des régulateurs qui fonctionne en numérique comme la montre la **figure(IV.8).** Puis en simule avec ce nouveau schéma (avec régulateur numérique) si comme ci on a simulé sur le VHDL et on aura les résultats suivant :



Figure(IV.12) : position maitre / esclave avec régulateurs numérique



Figure(IV.13) : force maitre/esclave avec régulateur numérique.

- **interprétation des résultats :**

A partir des graphes on a les mêmes résultats qu'avec les graphes de force et de position en analogique donc on a fait le bon choix des paramètres de la numérisation des régulateurs.

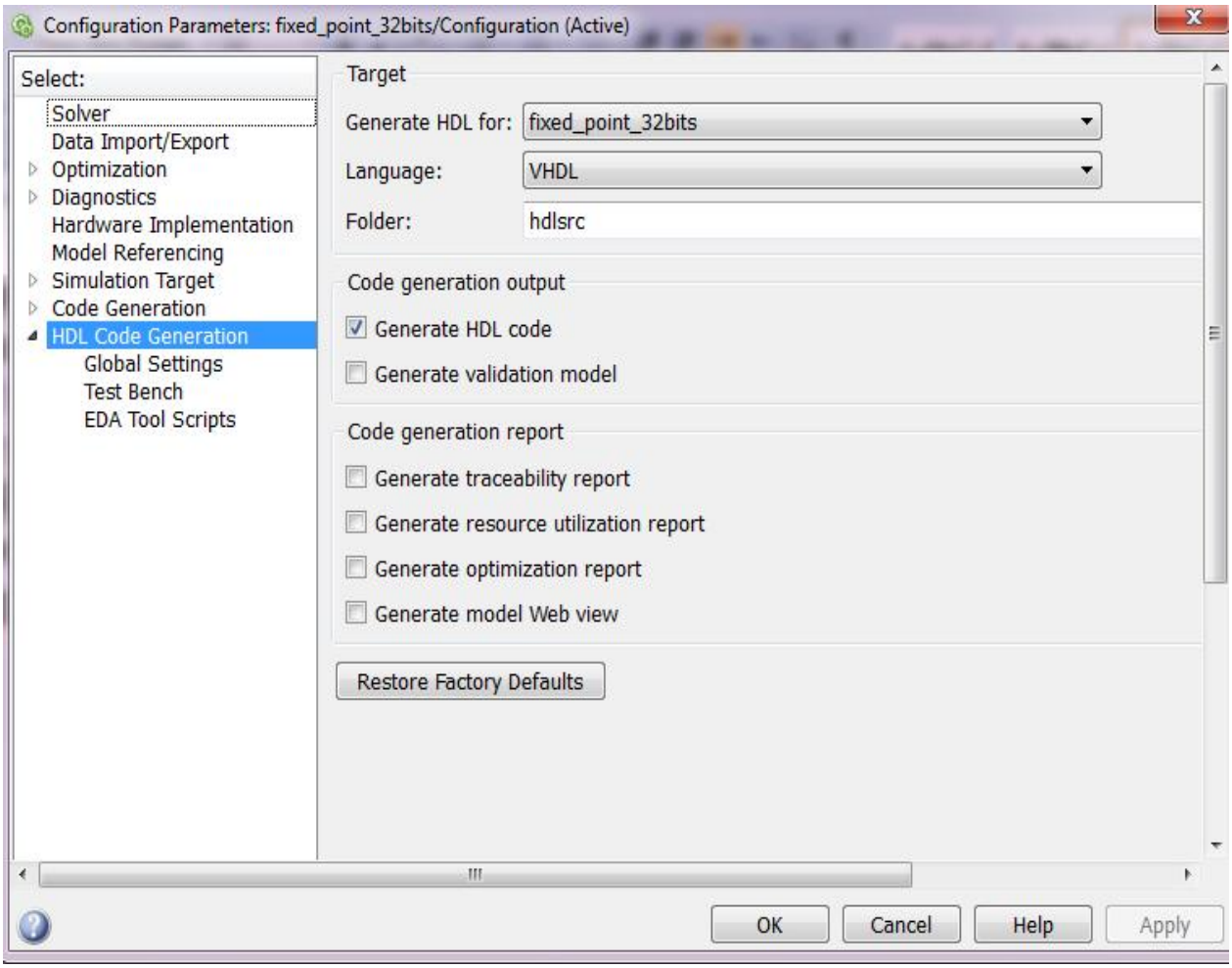
IV.6.2 génération du code VHDL :

Pour générer le code VHDL et sera l'essentielle de notre travaille pour l'implémenter sur notre carte de commande (FPGA) en doit dabord exécuter la commande suivante :

« c:\xilinx\14.2\ISE_DS\ISE\bin\nt\ise.exe »

Sur matlab commande pour que la matlab reconnaisse notre logiciel de programmation VHDL qui est l'ISE de la société xilinx et consulte sa bibliothèque VHDL.

Et puis en click sur le bouton droite de la souris en choisisse « HDL code »=> « HDL code propreties… » Et on aura :



Figure(IV.14) : fenêtre pour générer le code VHDL.

Et on vas choisir l'emplacement de notre fichier et le type de notre code et en click sur « apply » au finale on vas avoir notre programme VHDL et il seras prêt pour être charger sur FPGA (vertex5 ML501) en suivant toute les étapes d'implémentation sur ISE .

IV.7) Conclusion

Dans le présent chapitre, nous avons réalisé les étapes de programmation de circuit de commande de nos deux moteur (télé-opération : maitre/esclave) pour réaliser une tache a distance et ressentir les force appliquer sur l'esclave dans le maitre, et l'esclave vas avoir la même position que son maitre.

Conclusion générale :

Un télémanipulateur maître-esclave (TME) est un dispositif électromécanique qui permet à l'opérateur humain de réaliser une tache à distance toute en manipulant l'interface maître. Les mouvements qu'il réalise avec cette interface sont transmis vers le manipulateur esclave via un réseau de communication, de l'autre coté, le manipulateur esclave reproduit les mouvements de manière à accomplir la tache.

On parle de télé opération bilatéral si les interactions entre le manipulateur esclave et l'environnement sont restituées à l'utilisateur par l'intermédiaire de l'interface maître.

Dans le cas contraire, si les interactions ne sont pas renvoyées à l'utilisateur on parle de télé opération unilatéral.

Et dans notre projet on a réalisé l'étude d'une architecture bilatérale.

En perspectives, notre travail peut être amélioré par :

- Implémentation du code généré sur une carte FPGA de type Vertex5 (ML501)
- Réalisation expérimentale de l'architecture générale (les moteurs maitre-esclave, les cartes de puissance,….. etc.).
- intégration d'une interface de contrôle des paramètres de l'architecture (position et vitesse) temporairement.

bibliographie :

[1] :NAAK HOURIA , HASSID HASSINA ET MOUSOUNI FADHILA ;mémoire sur l'implémentation de variateur de vitesse a base d'un circuit FPGA pour obtenir le diplôme d'ingénieur d'état en électronique a UMMTO ;2007.

[2] :Mr CHIBAH AREZKI ;mémoire sur CONCEPTION D'UN CONTROLEUR D'ETAGE DE PUISSANCE PAR FPGA pour obtenir le diplôme du magister en électrotechnique option : entrainement électrique a UMMTO ;2011.

[3] :D. HOUZET, l. BARRANDON ; CONCEPTION DE CIRCUIT EN VHDL ET VHDL-AMS ;2006.

[4] : ROMAIN BERNY ; INTRODUCTION AU LANGAGE VHDL POUR LA SYNTHESE ;2005.

[5] : philipe lecardonnel & philipe LETENNEUR ;introduction a la synthese logique VHDL ;2001.

[6] :JACQUES WEBER , MAURICE MEAUDRE ; le langage VHDL cours et exercices ;2001.

[7] : NACHEF TOUFIK ; mémoire sur implimentation d'une instruction sur un FPGA pour obtenir le diplôme magister en electronique à UMMTO ; 2011.

[8] :NAOUAR.M.W « commande numérique à base de composants FPGA d'une machine synchrone » thèse en Génie électrique à l'université de CERGY PONTOISE. 2007.

[9] : AIT OUALI. Z« applications des FPGA à la commande d'un moteur asynchrone » mémoire de magister à l'université MMTO.

[10] : Eduardo Sanchez EPFL « Circuits reconfigurables: Les FPGAs »

[11]: Wikipedia FPGA, http://en.wikipedia.org/wiki/FPGA)

[12] : « FPGA EPRM » Conservatoire National des Arts et Métiers FIP-CPI 2013–2014

[13]: Arnaud Lagger, "Self Reconfiguration plateform for cryptographic application", Master theisis, 2006, Swiss federal institute of technology lausanne.

[14] : MARQUES. N « Méthodologie et architecture adaptative pour le placement efficace des taches matérielles de tailles variables sur des partitions reconfigurables » THESE à l'université de Lorraine 2012

[15] : Jean-Luc Danger « Les FPGA Principes innovants et tendances » 8 Mars 2012

[16] : Aura Nancy Rodriguez architecture générale pour la télé robotique : thèse de doctorat de l'université Paul Sabatier de Toulouse . janvier 2003

[17] :Pierre LETIER bras exosquelette haptique : conception et contrôle(laboratoire des structure active service des constructions mécaniques et robotique) juillet 2010

[18] : Laurent Barbé , thèse de l'école doctorale mathématique , sciences de l'information et de l'ingénieur télé opération avec retour d'efforts pour les interventions percutanées .

[19] : Yves BRIERE , Télé opération en présence de retard : thèse de doctorat de l'école nationale supérieure de l'aéronautique et de l'espace .

[20] : Nabil Zemiti , commande en effort des systemes robotique pour la chirurgie mini-invasive : université paris 6 spécialité mécanique-robotique 2005.

[21]: (Xilinx, Virtex-II complete datasheet", Product Specification, no. 31, Mars 2005Available from: http://www.xilinx.com/partinfo/ds031.pdf

[22]: Xilinx Inc., 'ML501 Evaluation Platform - User Guide', UG226 (v1.4),Août2009.

[23]: www.XILINX.com