الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

وزارة التعليم العالي و البحث العلمي
المدرسة الوطنية المتعددة التقنيات

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

Département d'Electronique

Projet de fin d'études

Pour l'obtention du diplôme

D'Ingénieur d'état en Electronique

THÈME

ETUDE ET MISE EN ŒUVRE DE LA CARTE XS40 CONCEPTION ET IMPLEMENTATION D'UN CODEUR/DECODEUR SUR FPGA XC4005XL

Etudié par

M^elle LANI Fatiha
Mr DIB Hakim

Proposé et dirigé par

Mr. R. SADOUN

Promotion Juin 2004

Département d'Electronique
Ecole Nationale Polytechnique, 10, AV. Hassen Badi, El-Harrach, Algérie

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

Je remercie en premier lieu Dieu, le tout puissant qui m'a donné le courage et la patience pour mener à bien mes études.

Je dédie ce travail

A mes très chers parents, qui m'ont appris à progresser, et dont le sacrifice, l'amour, la patience, le suivi, et compréhension sont l'essence de ma réussite.

A mon très cher oncle Mohammed qui m'a toujours soutenu.

A mes très chers sœurs, Souad et Fatma Zohra, et mon frère Mohammed pour leur compréhension.

A mes grand parents.

A la mémoire de mon regretté oncle Hocine.

A tous les amis de mon père messieurs Salhi, Moukhtari, Mohktari et leurs familles pour leur soutien et leurs amitié.

Aux personnes qui m'ont offert des stages, Mme Berkane de l'ENSI; a Mr. Benhamouda, Mr. Fatmi, Mr. Boulahouache de NAFTAL

A tous les enseignants qui ont contribués à ma formation.

A tous mes collègues du groupe électronique en particulier Gacem, Sami et Adel, qui me considère comme leur sœur.

A tous mes amis Mimi, Amina, Samiha, Nabila, Samira, Lamia ...

LANI

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

REMERCIEMENTS

Le travail, présenté dans ce mémoire de fin d'études, a été effectué au sein du laboratoire d'Implémentation, de l'Ecole Nationale Polytechnique d'Alger (ENP), sous la direction de Monsieur M. R.SADOUN, que nous remercions pour son encadrement, son aide, ses conseils précieux et pour avoir mis à notre disposition sa documentation et le matériel du laboratoire.

On remercie également les membres du jury, qui nous ont fait l'honneur de participer à l'examen de ce travail.

On remercie toute personne, qui nous a apporté son aide, durant toutes nos années d'études.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

ملخص

هذا العمل يهدف إلى دراسة و إنجاز بطاقة التجسيد XS40 للدارات المنطقية المبرمجة للسلسلة XC4000. هذه البطاقة تسمح ببرمجة الدارة FPGA بملف البرمجة الناتج عن أداة تطوير الدارات المبرمجة مباشرة من جهاز الكمبيوتر, مع إمكانية برمجة هذه الدارة انطلاقا من ذاكرة.
التجسيد يتم عن طريق إنجاز المرمز و المرمز العكسي (codeur/décodeur) على الدارة FPGA حسب خوارزمية HAMMING, المنسوب إلى نظام تبادل المعطيات Maître/esclave. نموذج هذا النظام, يمكن تكييفه لأنواع أخرى من الخوارزميات.

الكلمات المفتاحية : FPGA، البطاقة XS40 المبرمجة لـ: FPGA ، منهجية التصميم للدارات المبرمجة، ترميز HAMMING.

Résumé

Ce travail consiste en l'étude et en la mise en œuvre d'une carte d'implémentation XS40 des circuits FPGA de la série XC4000. Cette carte permet la configuration de l'FPGA avec des données issues d'un outil de développement de circuits programmables directement d'un PC, avec une possibilité de configurer le circuit à partir d'une mémoire.

La mise en œuvre a été concrétisée à travers l'implémentation sur le composant FPGA d'un codeur/décodeur suivant un algorithme de codage canal de Hamming, associé à un système d'échange de données maître/esclave. Le modèle du système ainsi obtenu peut-être adapté à d'autres types de traitement

Mots clés: FPGA; La carte XS40; méthodologie de conception, codage de HAMMING.

Abstract

This work deals with the study and the progamming of the XS40 - implementation board made with the XC4000 series FPGA circuits. This board allows the FPGA configuration based on data from a programmable circuit developement tool directly from a personal computer (PC), with a possible configuration of the FPGA – circuit from a memory.

The programming of the XS40 – board was achieved by the implementation of a encoder/decoder on the FPGA component according to a Hamming channel encoding. The encoder/decoder is associated with a master/slave data exchange. The resulting model may be suited for other data processing applications.

Key words: FPGA; XS40 – board lite; design flow, HAMMING encoding.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

SOMMAIRE

CHAPITRE I

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE
Ecole Nationale Polytechnique

CHAPITRE II

CHAPITRE III

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

CHAPITRE IV

PARTIE I :METHODOLOGIE DE CONCEPTION

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE المكتبة
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

LISTE DES FIGURES ET TABLEAUX

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

PARTIE I:

PARTIE II:

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

NOMENCLATURE

ABEL	(Advanced Boolean Expression Lanaguage): langage de programmation des circuits de faible densité d'intégration.
ALTERA	fabricant de circuits programmables.
ASIC	(Application Specific Integrated Circuit): circuit non programmable configuré lors de sa fabrication pour une application spécifique.
BA	(BUS d'Adresse).
BITSTREAM	flux de données
CAO	(Conception Assistée par Ordinateur)
CLB	(Configurable Logic Bloc): appellation de XILINX pour les blocs configurables dans un FPGA représentant la composante calculatoire.
CPLD	(Complex Programmable Logic Device): circuit intégrant plusieurs PLD sur une même pastille.
DSP	(Digital Signal Processor): processeur dédié au traitement du signal.
EEPROM	(Electrical Erasable Programmable Read Only Memory): mémoire ROM programmable et effaçable électriquement.
FIFO	(First_in First_out buffers).
Flash EEPROM	EEPROM utilisant 2 transistors par point mémoire. utilisé pour les connexions dans les CPLD.
FPGA	(Field Programmable Logic Array): réseau programmable à haute densité d'intégration.
FPLD	(Field Programmable Logic Device) : terme générique pour les CPLD et FPGA.
FSM	(Finite State Machine): Machine d'états finis.
HDL	(Hardware Description Language): langage permettant la description des circuits.
JTAG	(JOIN TEST ACTION GROUP).
IOB	(Input Output Bloc): bloc d'entrée/sortie permettant l'interface de l'FPGA avec les modules extérieurs.
IRQ	(Interruption Request): demande d'interruption
ISP	(In Situ Programmable): caractérise un circuit reprogrammable sur l'application.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE
Ecole Nationale Polytechnique

LAB	(Logic Array Bloc): Un CPLD est constitué d'une matrice de LABs constituées de plusieurs PLD
LUT	(Look Up Table): nom donné aux cellules mémoire réalisant les fonctions combinatoires dans les CPLD et FPGA.
MUX	abréviation pour multiplexeur
NCD	(Native Circuit Description): format de netlist qui donne le représentation physique de la conception mappé.
OLMC	(Output Logic Macro Cell): Cellule de base dans les PLDs et les CPLDs permettant de réaliser une fonction combinatoire que séquentielle.
OTP	(One Time Programming): Cellules programmables une seule fois
PAL	(Programmable Array Logic): ancien nom des PLD.
PIA	(Programmable Interconnect Array): matrice d'interconnexion dans les CPLD
PLA	(Programmable Logic Array): réseau à matrice ET et OU permettant la réalisation de fonctions combinatoires.
PLD	(Programmable Logic Device) : circuit logique programmable intégrant une matrice ET programmable, une matrice OU fixe et plusieurs cellules de sortie.
PROM	(Programmable Read Only Memory): mémoire ROM programmable.
RISC	microprocesseur à jeu d'instructions réduit
RTL	(Register Transfer Level): La description porte sur les états des registres et les transferts d'états. On distingue trois grandes parties: la partie chemin de données qui représente les calculs, la partie contrôle, et la partie mémoire.
SRAM	(Static Random Acess Memory): technologie utilisée pour les connexions dans les CPLD et FPGA.
VERILOG	langage de synthèse des circuits numériques
VHDL	(VHSIC Hardware Description Language): un langage destiné au début à la documentation, il permet la synthèse et la modélisation des circuits numérique.
VHSIC	(Very High Speed Integrated Circuit)
UAL	(Unité Arithmétique et Logique)
UV_EPROM	(UV_Erasable PROM): PROM effaçable par UV.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

XILINX	fabricant de circuits programmables
XSLOAD	utilitaire de chargement des données de configuration.
XSPORT	utilitaire de test de la carte XS40.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

INTRODUCTION GENERALE

L'apparition des composants programmables hautement flexibles a révolutionné le monde de l'électronique. Aujourd'hui, l'utilisation de ces composants s'étend depuis le monde de l'analogique, seul terrain encore gardé par les composants prédiffusés, jusqu'aux cœurs de microprocesseurs pour des applications à la fois câblées et programmées.

C'est dans ce contexte qu'est apparue l'application des langages de description matériel afin de permettre un recensement des fonctionnalités à réaliser et laisser au logiciel porteur le soin de résoudre les équations et autres « tables de vérité ». La puissance descriptive de ces langages en fait l'outil incontournable de toute conception électronique d'aujourd'hui avec en plus la flexibilité du choix de la cible technologique. Changer la version d'un ensemble numérique n'impose alors que la reprogrammation du composant sans même l'extraire de son environnement matériel. Les gains en terme de vitesse et de coût réalisés sont importants que ce soit dans le milieu de l'électronique professionnelle que celui de l'amateur.

L'objet de notre travail s'inscrit dans l'utilisation de ces composants. Il consiste en l'étude et en la mise en œuvre d'une carte à base d'un circuit programmable FPGA (Série XC4005XL). Comme application à la mise en œuvre de cette carte, il nous a été proposé l'implémentation d'un codeur ou décodeur pouvant être intégré dans une architecture maître/esclave.

Nous avons structuré la rédaction de notre mémoire en quatre chapitres principaux recouvrant toutes les étapes nécessaires pour passer de la cible technologique jusqu'à l'implémentation de l'application.

Le chapitre I fait un rappel sur les différents circuits numériques dits programmables (FPGA, CPLD, PLD,...), leurs architectures, leurs technologies d'interconnexion, leur classification ainsi que les avantages et les inconvénients que présentent chacun d'eux.

Le chapitre II sera consacré à l'étude d'un cas particulier de circuits programmables en l'occurrence le FPGA XC4005XL associé à la carte qu'on va étudier. On présentera en premier lieu la partie de son architecture dédiée à l'implémentation. En second lieu, nous aborderons les mécanismes dédiés cette fois ci à la gestion de ses ressources internes dont la communication avec l'extérieur.

Le chapitre III traitera en détail la carte XS40 destinée à la programmation du XC4005XL.

Le chapitre IV sera consacré à la conception de l'application cible et sa mise en œuvre sur la carte XS40. Il sera décomposé en deux parties. La première explore la méthodologie de

1. INTRODUCTION

L'un des éléments clés qu'on a utilisé durant notre travail est le FPGA (Field Programmable Gate Array). Il représente le cœur de la carte d'implémentation objet de notre étude. Ce type de circuit fait partie d'une famille dite famille des Circuits Programmables numériques (PLD). Toute conception qui en découle présentera de nombreux avantages par rapport à une conception analogique : grande insensibilité aux parasites et aux dérives des caractéristiques des composants, meilleure modularité et (re)configurabilité, facilité de stockage de l'information ...etc.

La figure ci-après tente une classification possible des circuits numériques.

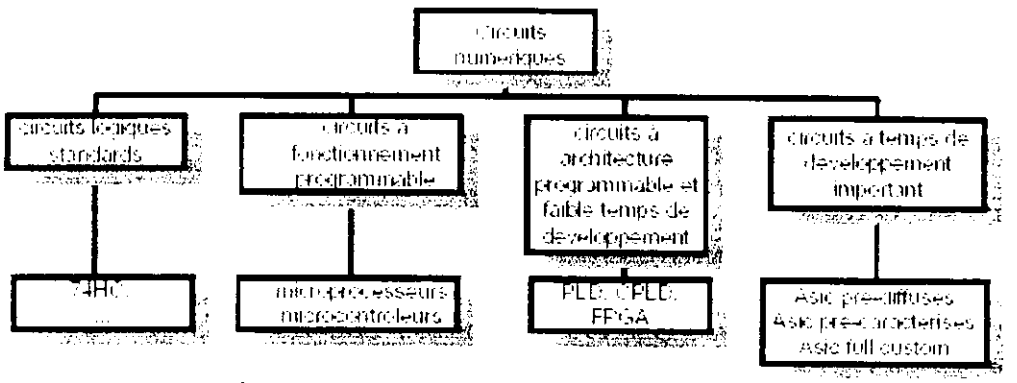


Figure I.1 : Classification des circuits numériques

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

La plupart de ces circuits sont maintenant programmés à partir d'un simple ordinateur directement connecté à son environnement. En cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement.

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

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

Nous nous intéresserons surtout aux circuits à architecture programmable à faible temps de développement. Le principe de base de ces circuits consiste à realiser des connexions logiques programmables entre des structures présentant des fonctions de bases. Le premier problème va donc être d'établir ou non suivant l'algorithme à implémenter, un contact electrique entre deux points. Avant de passer à la présentation de ces circuits et à leur programmation, nous nous intéressons d'abord à leurs technologies d'interconnexion.

2. CODAGE D'UNE FONCTION LOGIQUE [11]

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

Pour coder une fonction combinatoire, trois solutions sont classiquement utilisées:

2.1. SOMMES DE PRODUITS, PRODUITS DE SOMME ET MATRICE

N'importe quelle fonction peut être codée par une somme de produits, par un produit de sommes ou un mélange des deux. La figure ci-après illustre cette technique de codage d'une fonction combinatoire. On peut immédiatement en déduire une structure de circuits, appelé matrice PLA (Programmable Logic Array) qu'on va présenter dans le § 4.1.

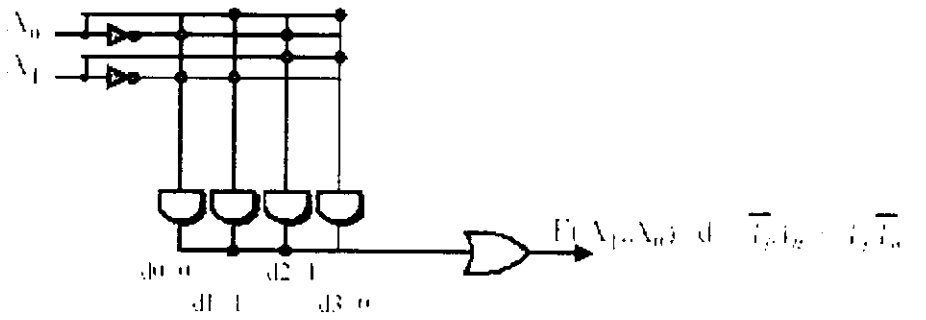


Figure I.2:Exemple [12] de programmation d'une fonction avec la technique somme produits

2.2. MEMOIRES

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

Ce principe utilisé pour les mémoires l'est aussi pour les CPLD (Complex Programmable Logic Device) et surtout pour les FPGA (Field Programmable Gate Array) sous le nom de LUT (Look Up Table).

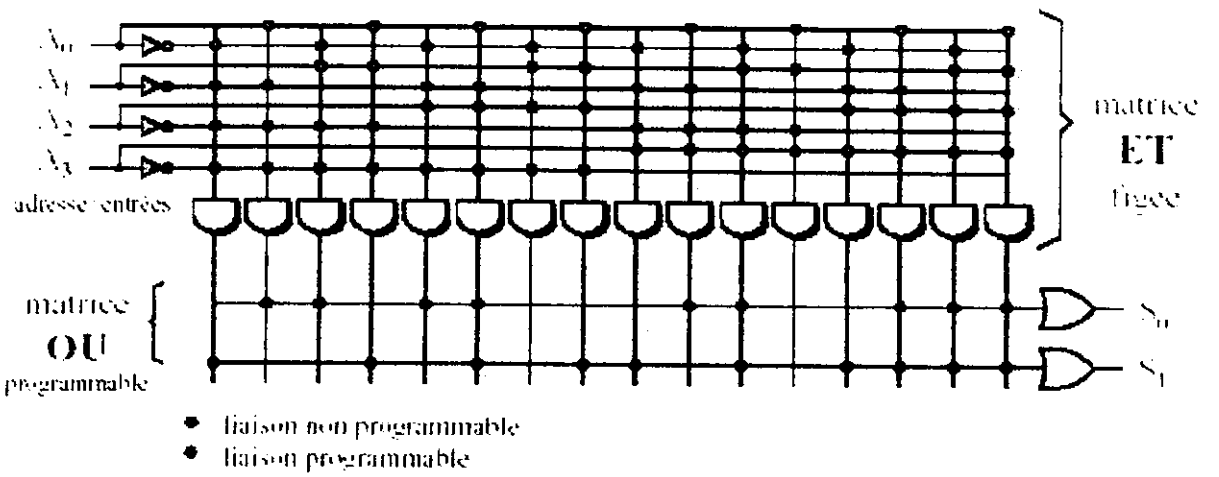


Figure I.3: Architecture d'une mémoire [12]

2.3. MULTIPLEXEUR

Le multiplexeur permet également de coder une fonction combinatoire, comme le montre la figure suivante :

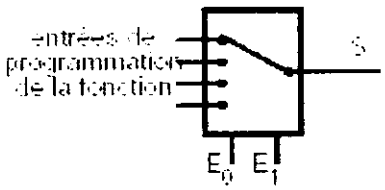


Figure I.4: Multiplexeur

A chaque valeur des entrées E0 et E1 est associé un niveau logique défini dans la table de vérité pour la sortie S. Ce niveau logique est imposé sur l'entrée de programmation correspondante. Ce principe est utilisé dans les FPGA.

3. TECHNOLOGIE D'INTERCONNEXIONS [4], [11], [13]

Comme nous venons de le voir, l'un des éléments clé des circuits étudié est la connexion programmable. Le choix d'une technologie par rapport à une autre dépendra essentiellement des contraintes imposées par l'utilisateur à savoir :

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

Plusieurs technologies sont décrites dans cette partie. Nous avons scindé leur description en deux sections : la première est relative aux dispositifs programmables une seule fois OTP (One Time Programming) utilisant des fusibles ou des anti-fusibles tandis que la seconde est relative aux dispositifs programmables plusieurs fois utilisant différents types de mémoire.

3.1. CONNEXIONS PROGRAMMABLE UNE SEULE FOIS (OTP)

3.1.1. CELLULES A FUSIBLE

Ce sont les premières à avoir été utilisées et elles ont aujourd'hui disparu au profit de technologies plus performantes. Leur principe(figure1.5)consistait à détruire un fusible conducteur par passage d'un courant fourni par une tension supérieure à l'alimentation (12-25 V).

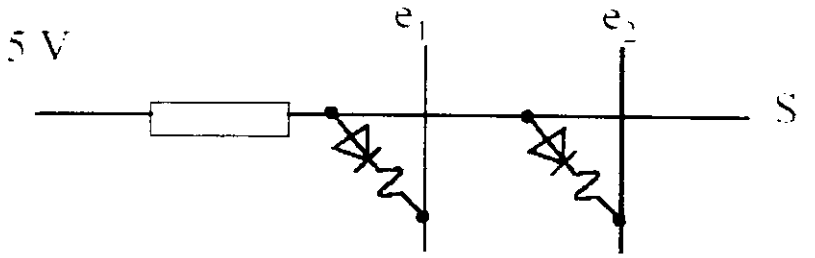


Figure I.5:Utilisation des fusibles

3.1.2. CELLULES ANTIFUSIBLES

En opposition aux cellules fusibles, l'interrupteur dans ce type de cellule (l'état de la connexion entre deux points) est par défaut ouvert, la programmation va donc établir la connexion .Le principe de programmation consiste à faire diffuser du silicium conducteur dans la couche diélectrique

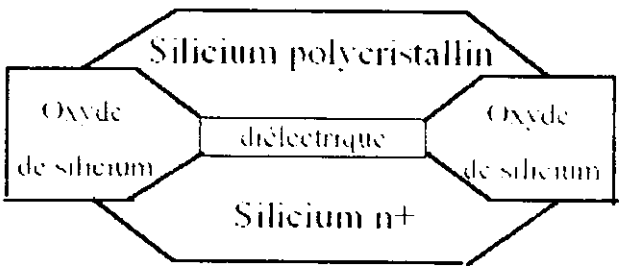


Figure I.6:Structure d'une cellule antifusible

3.2. CELLULES REPROGRAMMABLES

3.2.1. CELLULE A TRANSISTOR MOS A GRILLE FLOTTANTE ET EPROM (ERASABLE PROGRAMMABLE READ ONLY MEMORY)

L'apparition du transistor MOS à grille flottante a permis de rendre le composant bloqué ou passant sans application permanente d'une tension de commande. Le principe de programmation consiste à piéger des électrons dans la grille flottante qui s'opposent à la conduction dans le canal; le transistor est alors équivalent à un interrupteur ouvert. Lorsque le transistor n'est pas programmé, la grille flottante ne contient aucun électron, le canal est conducteur et le transistor est équivalent à un interrupteur fermé.

L'extraction éventuelle des électrons piégés permet le retour à l'état initial. Plusieurs technologies EPROM sont en concurrence :

- **UV- EPROM**

Les connexions sont réinitialisables par une exposition à un rayonnement ultraviolet durant environ 20 minutes, une fenêtre étant prévue sur le composant. Cet effacement est reproductible plus d'un millier de fois. L'inconvénient de ce type de cellule est que l'effacement n'est pas sélectif et ne permet pas la programmation sur site d'utilisation du système. Il est en effet nécessaire de retirer le composant de son support pour l'effacer.

- **EEPROM (Electrically EPROM)**

L'effacement et la programmation se font cette fois électriquement avec une tension de 12 V et peuvent se faire de manière sélective (la reprogrammation de tout le composant n'est pas nécessaire).

Une cellule demande toutefois 3 transistors pour sa réalisation (voir figure 1.7), ce qui conduit à une surface importante (75 à 100 µm2 en CMOS 0,6 µm) et réduit la densité d'intégration possible. D'autre part le nombre de cycles de programmation est limité à un nombre de 100 (en CMOS 0,6 µm) à 10 000 (en CMOS 0,8 µm) à cause de la dégradation des isolants. La programmation ou l'effacement d'une cellule dure quelques millisecondes.

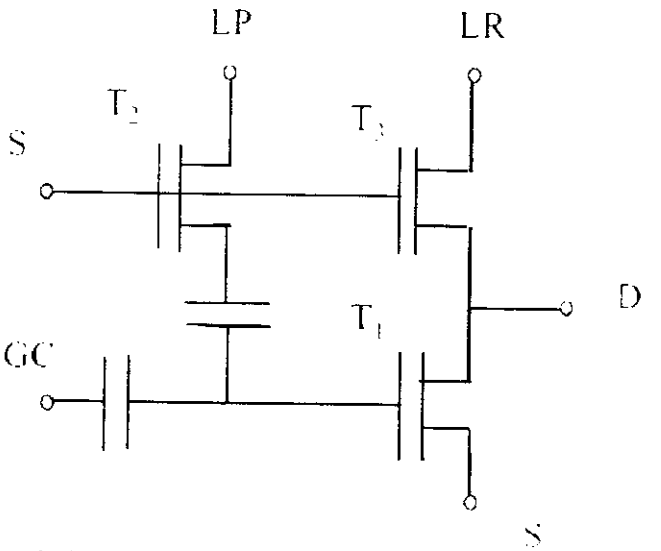
S: l'effacement ou la programmation ne peut se faire que si un 5Vest appliqué à cette ligne.

GC: (grille de contrôle) une tension positive sur GC permet la programmation de la connexion (migrer des électrons à la grille flottante). Inversement pour l'effacement, une tension négative permet l'extraction des électrons de la grille flottante.

LP: ligne de programmation.

T1: (transistor à grille flottante) représente un interrupteur ouvert, si la tension est positive sur GC ou fermé si c'est le contraire.

T2: transistor de programmation et d'effacement



LR : ligne de lecture de l'état de T1 à travers le transistor T3.

Figure I.7: Schéma électrique équivalent à une cellule mémoire EEPROM

- **Flash EPROM**

L'utilisation de deux transistors par cellule uniquement (3 pour l'EEPROM) et une structure verticale permettent une densité d'intégration importante (25 µm2 par cellules en CMOS 0,6 µm) trois à quatre fois plus importante que l'EEPROM, elle reste 10 fois moins que la technologie à antifusible.

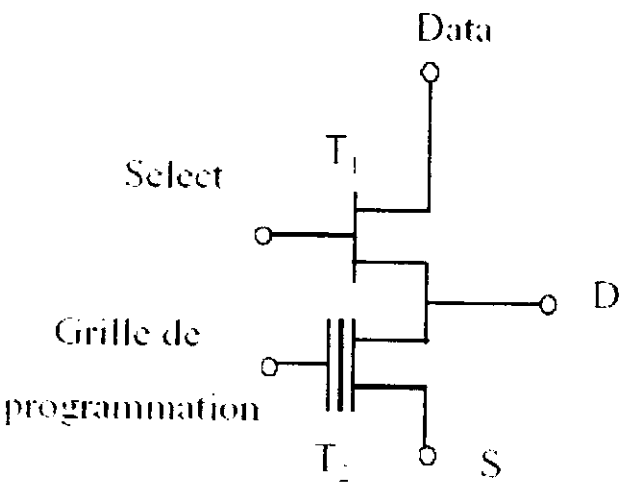


Figure I.8: Schéma électrique équivalent à une cellule mémoire Flash EPROM

Le nombre de cycles d'écriture est également plus grand que pour l'EEPROM car l'épaisseur de l'isolant est plus importante. Par contre, la simplicité de la cellule élémentaire n'autorise pas une reprogrammation sélective (éventuellement par secteur), ce qui n'est pas gênant pour le type de circuits qui nous intéressent. La tension de programmation et d'effacement est de 12 V, avec un temps de programmation de quelques dizaines de µs pour un temps d'effacement de quelques millisecondes.

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

Figure I.9: Programmation des circuits programmables

3.2.2. CELLULES SRAM (STATIC RANDOM ACESS MEMORY) A TRANSISTORS MOS CLASSIQUE

La SRAM est une mémoire dans laquelle on peut lire ou écrire autant de fois que l'on veut et sans restriction. C'est un dispositif programmable utilisé pour sa stabilité par rapport aux tensions de fluctuations et la rapidité d'effacement tel que en cours d'exploitation on peut changer complètement la fonction d'un système en rechargeant la mémoire .

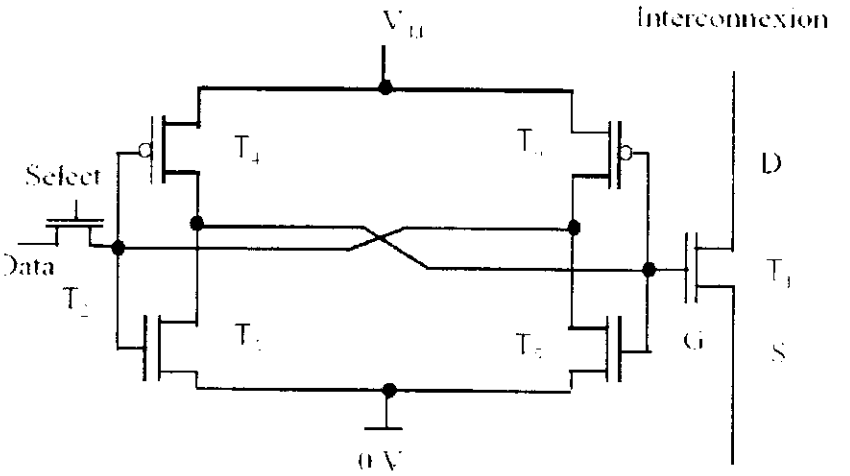


Figure I.10: Schéma électrique équivalent à une cellule SRAM

T1: transistor MOS, se comporte comme un interrupteur.

T3, T4, T5, T6:cellule réalisant le fonctionnement d'un bistable permettant la mémorisation.

T2: permet l'écriture et la lecture de la configuration de T1.

4. ARCHITECTURES UTILISEES

4.1. PLA (PROGRAMMABLE LOGIC ARAY) [4]

Les PLAs sont des circuits dont les deux matrices (côté **ET** et côté **OU**) sont programmables, de tels circuits peuvent être utilisés pour réaliser dans un espace réduit (taille d'un seul circuit intégré) un fonctionnement équivalent à un ensemble complexe de portes logiques. De plus, le temps de développement d'un système basé sur de tels circuits est nettement réduit, par rapport à la durée nécessaire pour concevoir un système basé sur des portes logiques :

- L'implantation des composants est facilitée, car les composants sont moins nombreux.
- Les passages de pistes sur les circuits imprimés sont plus faciles à concevoir et à réaliser, car il est facile de router un bus que des pistes individuelles.
- La maintenance est facilitée, car il est plus aisé de reprogrammer un PLA que de reconcevoir une implantation de circuits.

La figure I.11 représente une matrice PLA à 4 entrées et 4 sorties, Chacune des 4 entrées et son complémentaire arrive sur une des 2^4=16 portes ET à 2x4=8 entrées. L principe de

programmation de ce type de circuit (programmation des matrices ET et OU) est le même que celui exposé dans §2.1 et §2.2.

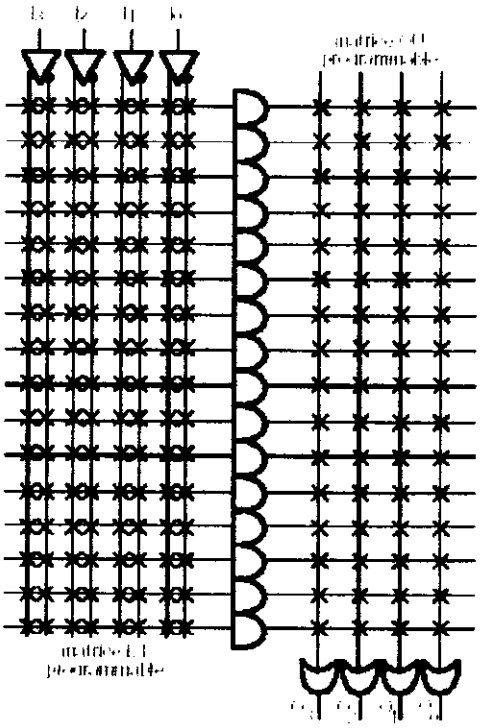


Figure I.11 : Structure d'un PAL

4.2. PLD (PROGRAMMABLE LOGIC DEVICE) [6]

Les PLD simples peuvent être représentés par le schéma bloc suivant :

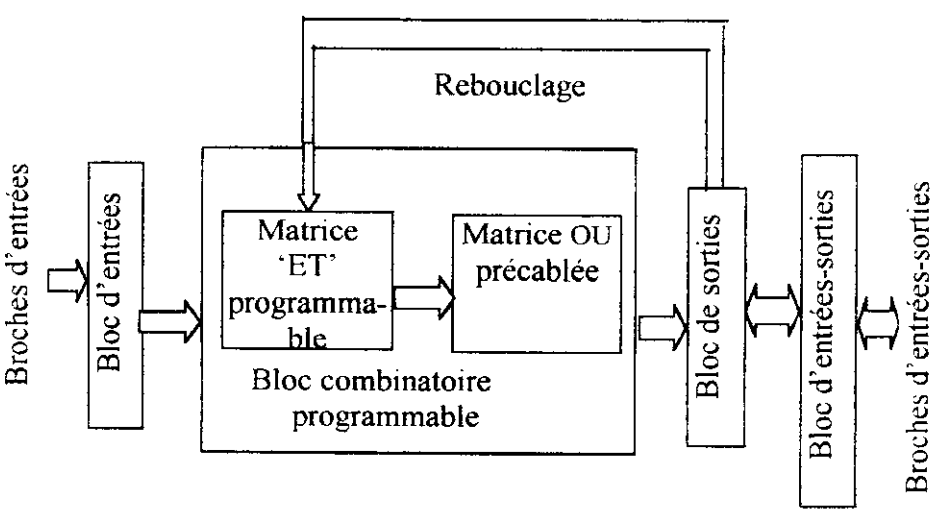


Figure I. 12 : Structure des PLD simple

Ils comportent les blocs suivants :

4.2.1. BLOC D'ENTREE

Le bloc d'entrée permet de fournir au bloc combinatoire l'état de chaque entrée et de son complément.

4.2.2. BLOC COMBINATOIRE

Ce bloc repose sur le principe suivant : toute fonction logique peut s'exprimer sous la forme d'une somme (matrice OU) de terme produit (matrice ET) avec des variables complémentées ou non.

4.2.3. BLOC DE SORTIES

On l'appelle aussi macro-cellule que l'on nomme OLMC, abréviation anglaise de Output Logic Macro Cell. Les différentes configurations de l ' OLMC et le bloc d'entrée/sortie associé donnent aux PLD leur souplesse d'exploitation. Les configurations sont choisies par le positionnement d'un certain nombre de fusibles, appelés fusibles de configuration. Une macro cellule est constituée de :

- Une porte OU exclusif, une bascule D ;
- Des multiplexeurs, qui permettent de définir les différentes configurations et un dispositif de rebouclage sur la matrice ET,
- Des fusibles de configurations.

Les éléments cités ci-dessus conduisent au schéma de principe d'une macro-cellule donné par la figure I.13. La macro-cellule permet de réaliser toute fonction logique écrite sous forme de somme de termes produit provenant de la matrice ET à l'aide de la matrice OU qui réalise la somme d'un nombre d'entrée fixe.

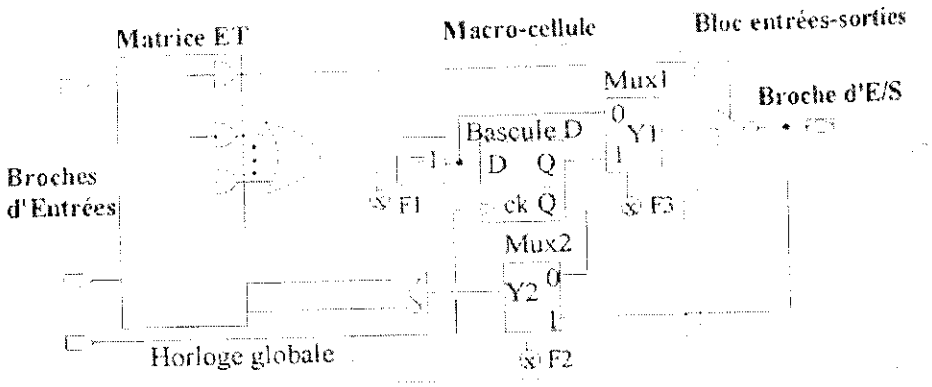


Figure I. 13 : Structure d'une macro-cellule

Cette fonction ou son complément peut :

- apparaître directement sur la sortie du multiplexeur, MUX1. On dit que la macro-cellule produit une sortie combinatoire (fusible F3 présent ➔ niveau bas)
- être appliquée à l'entrée de la bascule dont la sortie est aiguillée sur celle du multiplexeur, MUX1 .on dit que la macro-cellule produit une sortie séquentielle ou par registre (F3 absent ➔ niveau haut)

4.2.4. BLOC D'ENTREES/SORTIES

Ce bloc associé aux branches d'entrée/sortie constitue un élément de souplesse qui permet d'adapter les broches du circuit aux spécificités d'un système.

Les broches d'entrée/sortie sont bidirectionnelles à l'aide des 3 modes de fonctionnement du bloc d'E/S :

Sortie: Cette configuration est obtenue quand la porte 3-états présente un état basse impédance. Dans ce cas, les broches d'entrée/sortie permettent de commander des dispositifs externes. Dans ce cas, on peut avoir soit une sortie combinatoire ou une sortie séquentielle.

Entrée : La porte 3-états présente un état haute impédance (F2 doit être absent).Les broche d'entrée/sortie sont considérées alors comme des entrées du PLD.

Entrée/sortie : La porte 3-états est commandée par un terme produit .Lorsque ce terme vaut 1, le bloc se comporte en sortie, lorsque le terme vaut 0, le bloc se comporte en entrée.

4.3. CPLD (COMPLEX PROGRAMMABLE LOGIC DEVICE) [10]

La nécessité de placer de plus en plus de fonctions dans un même circuit a conduit tout naturellement à intégrer plusieurs PLD (blocs logiques) sur une même pastille, reliés entre eux par une matrice centrale.

Sur la figure suivante chaque bloc LAB (Logic Array Block) de 16 macro-cellules est l'équivalent d'un PLD à 16 OLMC. Ils sont reliés entre eux par une matrice d'interconnexion (PIA pour Programmable Interconnect Array).

Un seul point de connexion relie entre eux les blocs logiques. Les temps de propagation d'un bloc à l'autre sont donc constants et prédictibles.
La phase de placement des différentes fonctions au sein des macro-cellules n'est donc pas critique sur un CPLD, l'outil de synthèse regroupant au maximum les entrées/sorties utilisant des ressources communes.

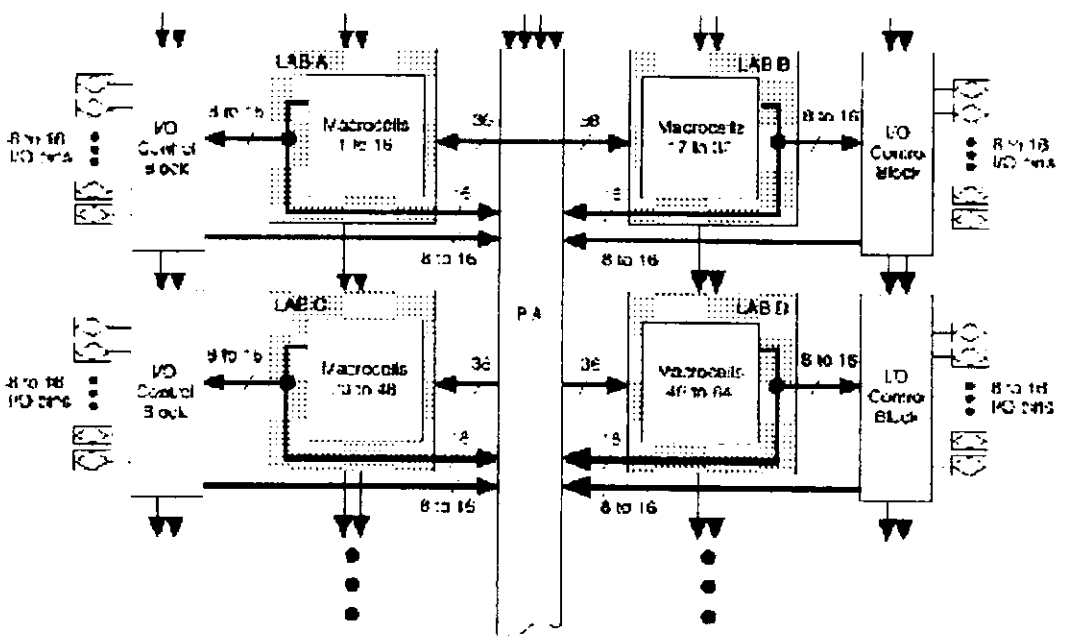


Figure I. 14 : Exemple de structure de CPLD

L'établissement des liaisons (routage) entre les différentes macro-cellules est encore moins critique, un seul point de connexion -cause de retard- relie les LAB entre eux. Le temps de propagation des signaux est parfaitement prédictible avant que le routage ne soit fait. Ce dernier n'influence donc pas les performances du circuit programmé.

Dans l'outil de synthèse, la partie s'occupant du placement et du routage est appelée le "fitter" (to fit : placer, garnir).La technologie de connexion utilisée est généralement l'EEPROM (proche de celle des PLD) ou EEPROM flash.

4.4. FPGA (FIELD PROGRAMMABLE GATE ARRAY)

4.4.1. GENERALITES

FPGA se traduit en français par circuits prédiffusés programmables. Contrairement aux circuits prédiffusés conventionnels, les circuits prédiffusés programmables ne demandent pas de fabrication spéciale en usine, ni de systèmes de développement coûteux. Inventés par la société Xilinx, le FPGA, dans la famille des ASICs, se situe entre les réseaux logiques programmables et les prédiffusés C'est donc un composant standard combinant la densité et les performances d'un prédiffusé avec la souplesse due à la reprogrammation des PLD. Cette configuration évite le passage chez le fondeur et tous les inconvénients qui en découlent.

Schématiquement, le FPGA est constitué de deux couches [9]:

- La couche configurable composée d'un périmètre de plots d'entrées/sorties, de cellules logiques et d'un réseau d'interconnexions.
- La couche mémoire SRAM qui permet de mémoriser la configuration du circuit. Cette mémoire contient la personnalisation des éléments de la couche configurable. La configuration est stockée dans une mémoire externe non volatile. A chaque mise sous tension, un dispositif permet de charger la mémoire SRAM en quelques millisecondes. Une seule mémoire externe suffit au chargement de plusieurs composants par un système de chaînage.

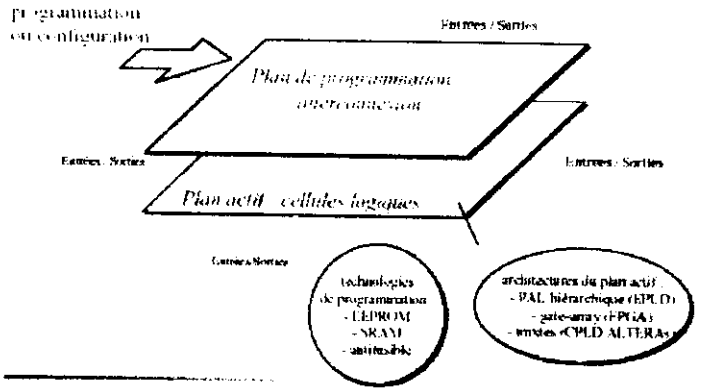


Figure I.15: Structure d'un circuit programmable (FPGA)

4.4.2. ARCHITECTURE INTERNE D'UN FPGA [11]

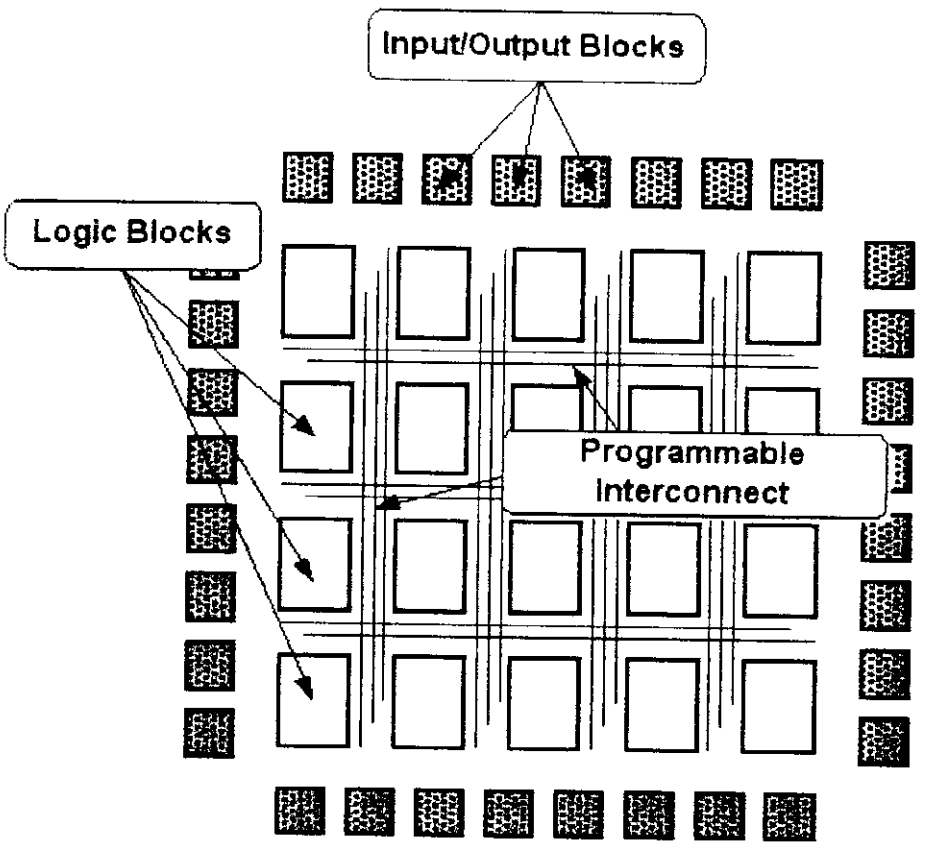


Figure I.16: Architecture d'un circuit programmable (FPGA)

Les blocs logiques sont plus nombreux et plus simple que pour les CPLD, mais cette fois les interconnexions entre les blocs logiques ne sont pas centralisées.

Comme le montre la figure ci-dessus, un FPGA est composé d'une matrice de blocs logiques, chacun d'eux est constitué de plusieurs macro cellules, ces blocs forment la composante calculatoire du FPGA permettant de réaliser n'importe quelle fonction combinatoire ou séquentielle. Plusieurs appellations sont utilisées suivant les différents fabricants, la plus répondue est celle de XILINX « CLB signifiant Configurable Logic Bloc ».

Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées/sorties dont le rôle est de gérer les entrées/sorties réalisant l'interface avec les modules extérieurs appelé IOB (XILINX), ces cellules sont totalement dissociées des structures logiques et sont plus ou moins complexes. Toutes possèdent des buffers trois états, le contrôle du taux de montée est associé généralement à chaque buffer de sortie .Pour certains composants, ces cellules proposent une bascule de maintien de signal de sortie et une bascule de synchronisation du signal d'entrée. La programmation du circuit FPGA consistera à interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux.

4.4.3. ROUTAGE DANS UN FPGA

La physionomie de routage [4] peut être vue comme une multitude de segments métalliques continus pouvant être reliés entre eux ou connectés en entrée ou en sortie des blocs logiques. Il existe donc plusieurs chemin possibles pour relier deux blocs logiques (responsable du temps de propagation) .Chaque interconnexion réalisée crée, sur la ligne formée de plusieurs segments, une résistance série supplémentaire non négligeable. Associée à la capacité du système d'interconnexion, à celle générée par les connexions aux modules, elle engendre un délai limitant la vitesse de fonctionnement. On doit noter que la distance entre deux blocs logiques n'est en aucune manière responsable du délai ; seul le nombre d'interconnexions et donc des résistances mises en série est responsable de la dégradation des délais.

La figure I.17 illustre très schématiquement deux liaisons au sein d'un FPGA [11], la liaison entre le bloc BD et le bloc BH utilise des segments courts. Les points d'interconnexions sont nombreux et le délai est conséquent. Par contre la liaison entre les blocs BA et BL utilise des lignes longues. Malgré l'éloignement spatial entre ces deux modules ; seul un point d'interconnexion est utilisé et le délai est optimisé.

Dans un CPLD, quel que soit le positionnement relatif entre deux éléments (broche d'entrée, macro-cellule) à relier, les ressources de couplage utilisées seront les mêmes. Par conséquent, le même modèle équivalent électrique est appliqué à chaque liaison et donc les délais engendrés seront aussi identiques. C'est en ce point que réside la différence entre CPLD et FPGA. Les performances d'un FPGA seront donc beaucoup plus difficiles à évaluer.

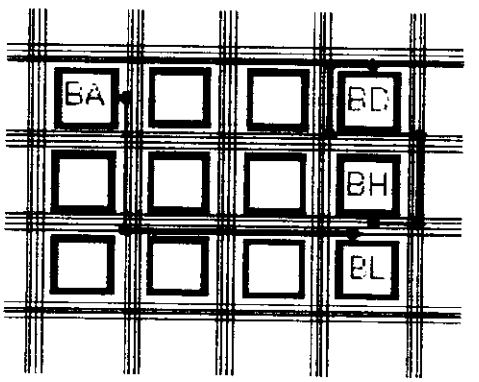


Figure I. 17 : Routage dans un FPGA

4.4.4. TYPE DE FPGA

On distingue deux types de FPGA :

- **FPGA de type SRAM**

Introduits à la fin des années 1980 ,les FPGA SRAM offrent des spécificités particulières (figure I.18), Le contenu des mémoires SRAM est automatiquement téléchargé à la mise sous tension, à partir d'une mémoire externe PROM proposée par le fabricant du FPGA. cette PROM est préalablement programmée avec les données de configuration . Il est aussi possible d'envisager plusieurs configurations qui correspondraient par exemple à des modes de fonctionnements différents d'un système [4].

Un autre mode de configuration [1] peut se faire en utilisant plusieurs mémoires externes permettant d'adapter au mieux l'architecture du circuit.
Le FPGA peut être considéré comme un esclave. Un système intelligent (microprocesseur) peut aller y écrire les données de configuration préalablement insérées dans une zone mémoire du système ou stockées sur disque dur.

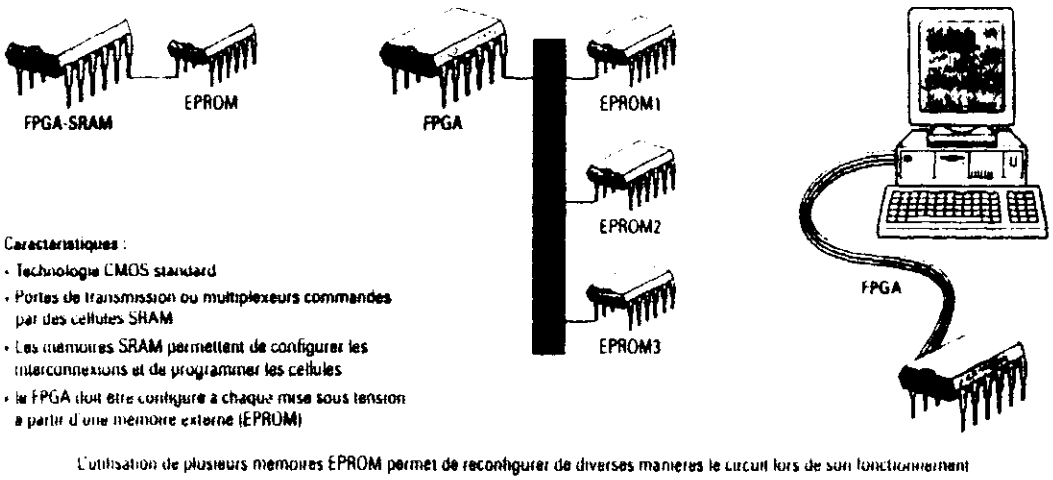


Figure I.18: Principe des circuits FPGA-SRAM

L'utilisation d'une technologie CMOS standard permet d'optimiser les coûts de fabrication et de prendre en compte les évolutions de ces procédés.

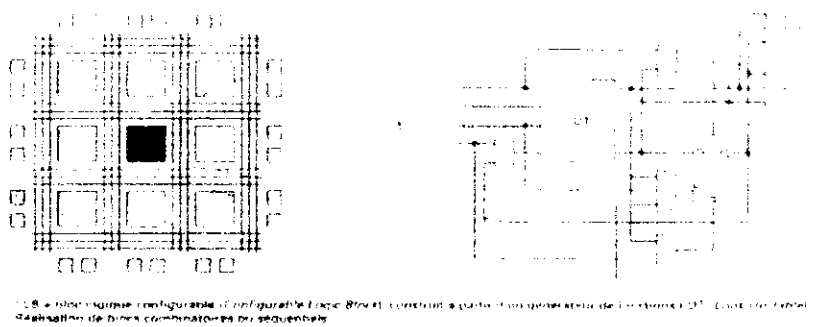


Figure I.19: Principe de l'architecture d'un FPGA-SRAM

- **<u>FPGA de type antifuse [4]</u>**

Les FPGA de type antifuse sont généralement beaucoup moins complexe que ceux des SRAM; la différence principale réside dans le fait que ce type de cellule est OTP (One Time

Programmable). La fonction est physiquement figée. Il s'agit en l'occurrence de multiplexeurs, les connexions effectuées en entrée permettent de déterminer la fonction. On notera que contrairement aux cellules SRAM, le nombre de sorties est réduit. Pour une fonction donnée le nombre de blocs logiques nécessaires est supérieur au nombre de blocs nécessaire dans un FPGA de type SRAM. Conséquence directe de ce constat, le nombre d'interconnexions est lui aussi largement supérieur. La compensation réside dans le fait qu'un antifusible est moins volumineux qu'une cellule SRAM. Le nombre de canaux de routage peut être augmenté. Les performances d'une connexion de type antifuse (de l'ordre de la centaine d'Ohm comparé à quelques centaines d'Ohm pour une connexion SRAM), autorisent aussi l'utilisation successive de beaucoup de modules sans trop affecter les temps de transfert.

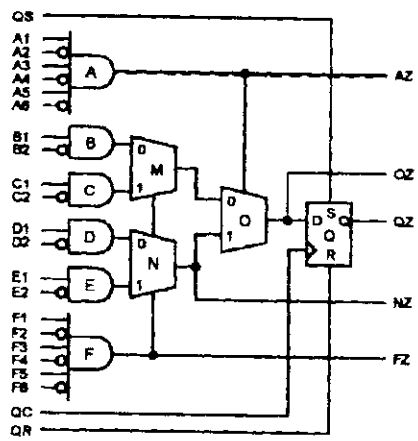


Figure I.20 : Bloc logique à base de multiplexeur

- **Comparaison entre les technologies Antifuse et SRAM [4]**

Les différentes solutions en matière de systèmes de connexion, retenues par les différents constructeurs ont une incidence directe sur la topologie, les caractéristiques et les performances des FPGAs. Cette incidence est essentiellement liée à la dimension même du système d'interconnexion. Pour une technologie donnée (exemple 0,6 microns), il est possible d'établir un comparatif entre les technologies SRAM et antifuse. Le tableau ci-dessous en fait la synthèse.

	SRAM	Antifuse
Taille physique du système d'interconnexion	+	-
Taille physique du bloc logique	+	-
Quantité de blocs logiques	-	+
Capacité fonctionnelle d'un bloc logique	+	-
Quantité de canaux de routage	-	+
Efficacité de la technologie (vitesse)	-	+

Tableau I. 1 : Comparaison entre technologie SRAM et ANTIFUSE

La complexité d'une fonction réalisable sur un bloc logique de type SRAM permet de réduire le nombre de liaisons nécessaires dans un composant. Le manque d'efficacité en terme de vitesse d'une connexion de type SRAM n'est donc pas un critère trop pénalisant. Dans une telle technologie, la priorité est plutôt donnée aux blocs logiques qu'aux ressources d'interconnexion.

Les composants de type antifuse exploitent simultanément le fait que la technologie est peu encombrante et que la résistance engendrée par une connexion de type antifuse est relativement faible. Le nombre d'interconnexions n'est donc pas trop pénalisant pour les délai .Il en découle l'utilisation possible d'un grand nombre de blocs logiques de taille réduite .

5. CONCLUSION

Dans ce chapitre, nous avons présenté les circuits programmables et les différentes architectures proposées. Nous nous sommes intéressés en particulier aux « FPGA ». Ces derniers permettent actuellement l'intégration de plus d'un million de portes logiques équivalentes sur le même circuit.

Les possibilités offertes par les circuits programmables de type FPGA SRAM permettent par ailleurs la mise en œuvre le concept de prototypage (ou maquette) pour la vérification fonctionnelle.

Pour mettre en évidence les particularités des circuits programmables de type FPGA, on va aborder dans le chapitre II l'étude d'un circuit FPGA de la série XC4000 de XILINX.

Le choix de cette série est imposé du fait que la carte de programmation qu'on va étudier n'est destinée qu'aux FPGA de la série XC4000.

1. INTRODUCTION

Dans ce chapitre on abordera l'étude d'une famille de circuits programmables FPGA XC4000. On s'y intéressera aux différents blocs constitutifs du circuit pour comprendre le processus de communication, de programmation et de test du circuit afin de faciliter l'étape d'implémentation d'algorithmes sur la carte XS40.

Introduite dés 1985, la gamme XILINX est sans doute la plus répandue. La gamme XC4000 s'étend du XC4003 (3000 portes) au XC4062XL (62000 portes). La configuration du composant porte sur deux points :

- la définition des interconnexions définies par des cellules SRAM.
- les fonctions booléennes définies dans des mémoires de type SRAM.

La mémorisation de la configuration est donc volatile. Ces circuits sont programmés après chaque mise sous tension. Les données de programmation sont issues soit directement d'une ROM (sous forme série ou parallèle), soit d'un système intelligent (processeur).

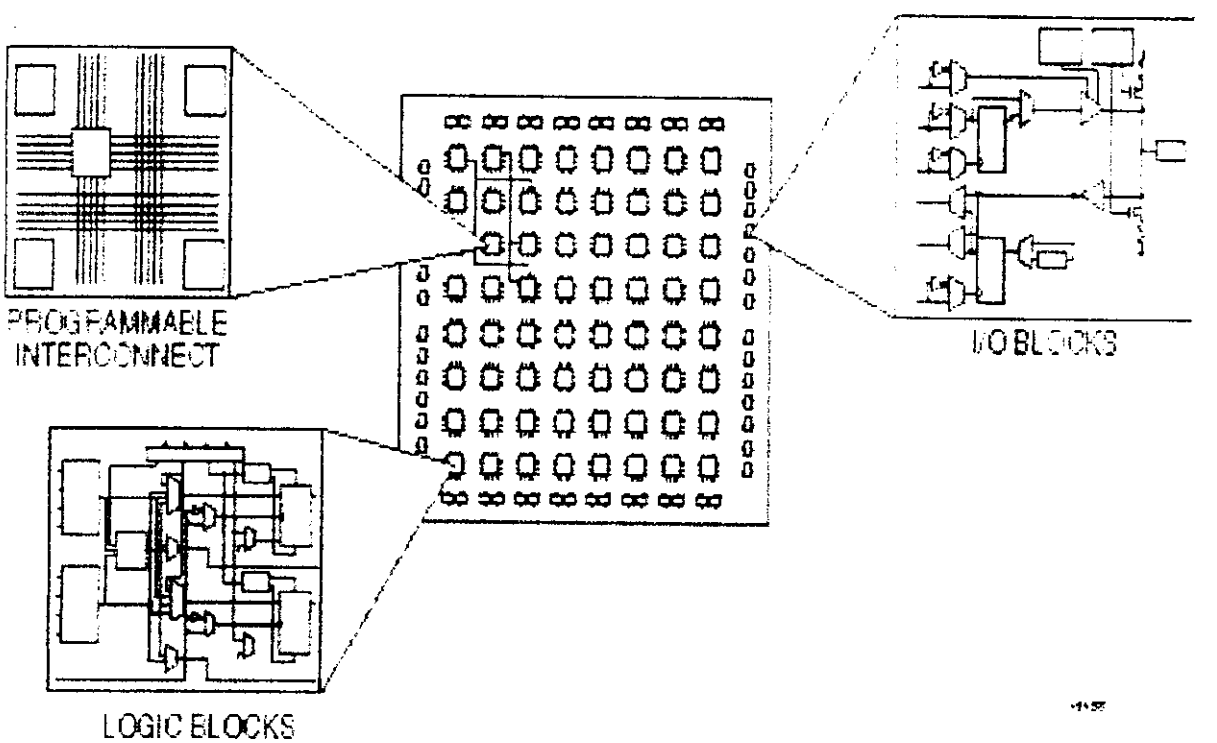


Figure II.1 : Structure générale

2. BLOCS DE BASES

Les composants de la famille XC4000 de XILINX sont constitués des blocs de base suivant :

- éléments configurables : ces éléments peuvent être des CLB ou des IOB.
- trois autres types de circuits importants qui sont :
  - des buffers trois états ;
  - des décodeurs ;
  - un oscillateur interne ;
- ressources d'interconnexion.
- automate dédié à la communication, test et programmation.

Le fonctionnement de chacun des blocs ainsi définis est déterminé pendant l'étape de configuration autrement dit pendant la programmation des cellules SRAM, qui définissent soit des fonctions logiques soit des interconnexions souhaitées.

3. LES CLB (CONFIGURABLE LOGIC BLOC) [7]

Les blocs logiques configurables sont les éléments déterminants des performances du FPGA. Chaque bloc est composé d'un bloc de logique combinatoire composé de deux générateurs de fonctions à quatre entrées et d'un bloc de mémorisation synchronisation

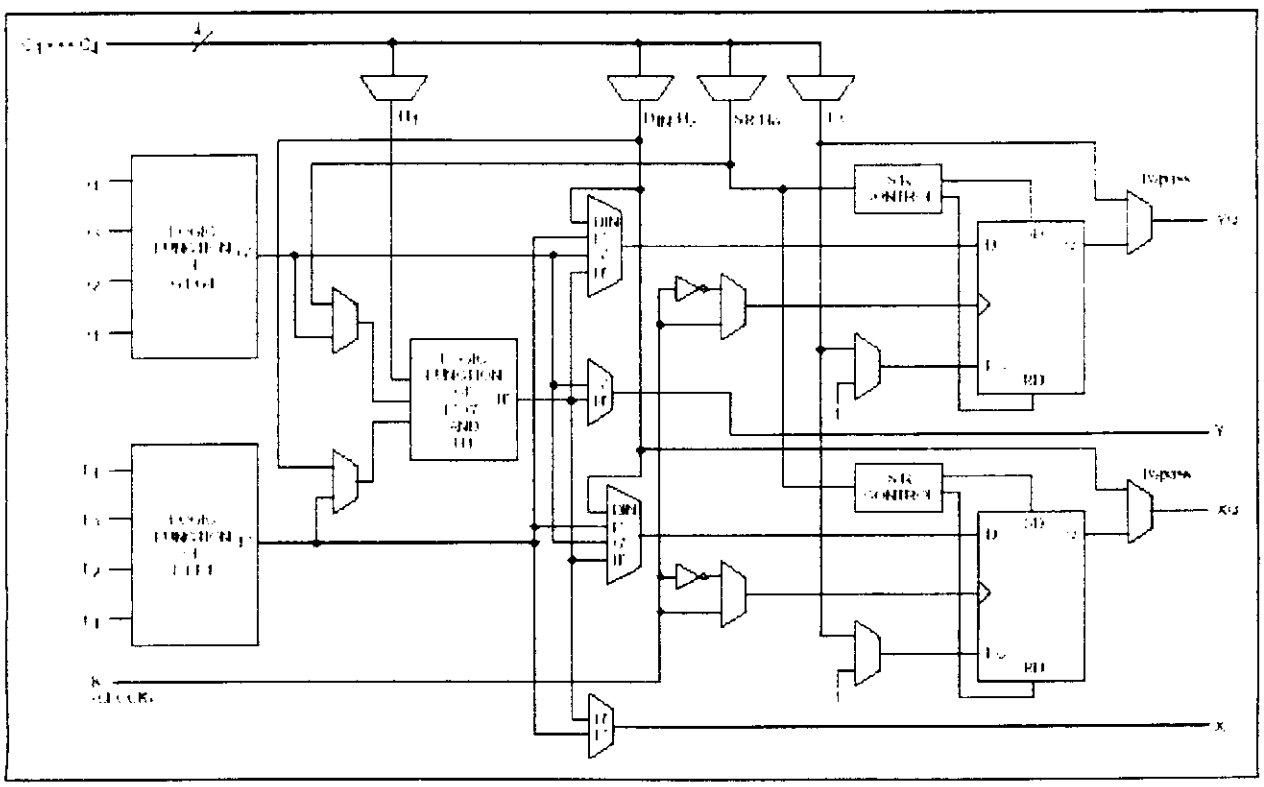


Figure II.2 : Cellules logiques (CLB)

composé de deux bascules D. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB. La figure II .2 nous montre le schéma d'un CLB.

Voyons d'abord le bloc logique combinatoire qui possède deux générateurs de fonctions F' et G' à quatre entrées indépendantes (F1...F4, G1...G4), lesquelles offrent aux concepteurs une flexibilité de développement importante car la majorité des fonctions aléatoires à concevoir n'excède pas quatre variables. Les deux fonctions sont générées à partir d'une table de vérité câblée inscrite dans une zone mémoire, rendant ainsi les délais de propagation pour chaque générateur de fonction indépendants de celle à réaliser. Une troisième fonction H' est réalisée à partir des sorties F' et G' et d'une troisième variable d'entrée H1 sortant d'un bloc composé de quatre signaux de contrôle H1, DIN, S/R, Ec. Les signaux des générateurs de fonction peuvent sortir du CLB, soit par la sortie X, pour les fonctions F' et H', soit Y pour les fonctions G' et H'. Ainsi un CLB peut être utilisé pour réaliser deux fonctions indépendantes à quatre entrées indépendantes ou une seule fonction à cinq variables ou deux fonctions, une à quatre variables et une autre à cinq variables.

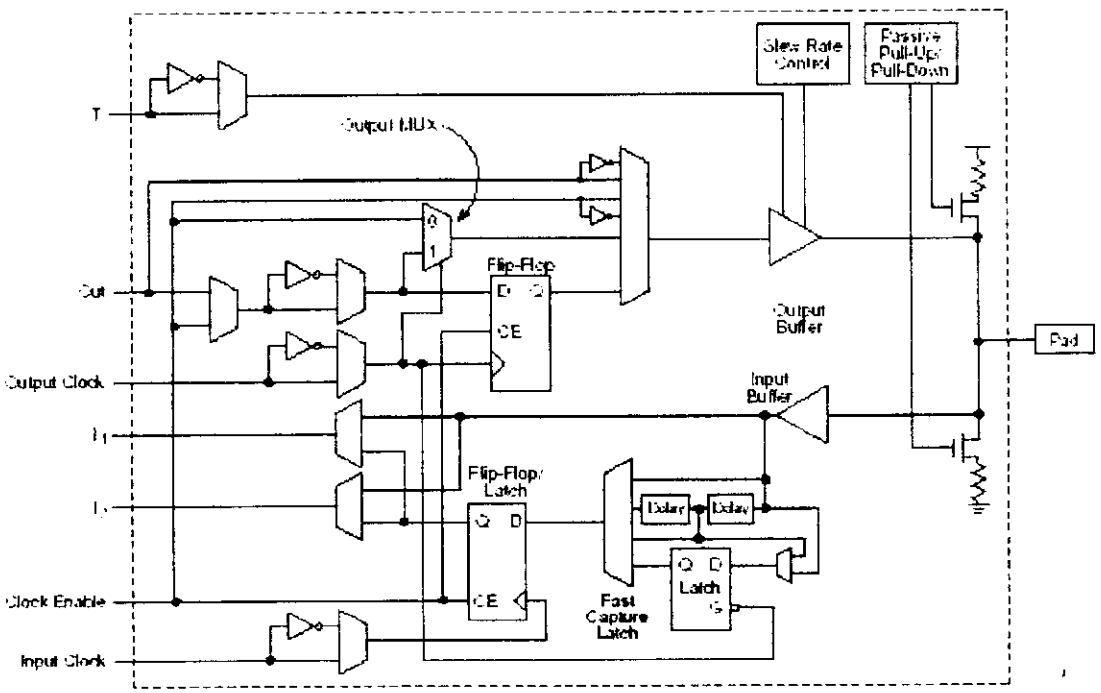
L'intégration de fonctions multi variables diminue le nombre de CLB nécessaires, les délais de propagation des signaux et par conséquent augmente la densité et la vitesse du circuit. Les sorties de ces blocs logiques peuvent être appliquées à des bascules au nombre de deux ou directement à la sortie du CLB (sorties X et Y). Chaque bascule présente deux modes de fonctionnement : un mode 'flip-flop' avec comme donnée à mémoriser, soit l'une des fonctions F', G', H' soit l'entrée directe DIN. La donnée peut être mémorisée sur un front montant ou descendant de l'horloge (CLK). Les sorties de ces deux bascules correspondent aux sorties du CLB XQ et YQ. Un mode dit de " verrouillage " exploite une entrée S/R qui peut être programmée soit en mode SET, mise à 1 de la bascule, soit en Reset, mise à zéro de la bascule. Ces deux entrées coexistent avec une autre entrée laquelle n'est pas représentée sur la figure II.2 appelée le global Set/Reset. Cette entrée initialise le circuit FPGA a chaque mise sous tension, à chaque configuration, en commandant toutes les bascules au même instant soit à '1', soit à '0'. Elle agit également lors d'un niveau actif sur le fil RESET lequel peut être connecté à n'importe quelle entrée du circuit FPGA.

3.1. UTILISATION DES LUT EN RAM OPERATIONNELLE

Un mode optionnel des CLB est la configuration en mémoire RAM de 16x1, 16x2 ou 32x1 bits. Les entrées F1 à F4 et G1 à G4 deviennent des lignes d'adresses sélectionnant une cellule mémoire particulière. La fonctionnalité des signaux de contrôle est modifiée dans cette configuration, les lignes H1, DIN et S/R deviennent respectivement les deux données D1, D0 d'entrée et le signal de validation d'écriture WE dans le cas (RAM 16x2 bits), par contre dans le cas de (RAM 32x1 bits) les lignes H1, DIN, S/R deviennent respectivement la cinquième adresse, donnée d'entrée D0, et WE. Le contenu de la cellule mémoire (D0 et D1) est accessible aux sorties des générateurs de fonctions F' et G'. Ces données peuvent sortir du CLB à travers ses sorties X et Y ou alors en passant par les deux bascules.

4. LES IOB (INPUT OUTPUT BLOC) [7]

La figure II.3 présente la structure d'un bloc IOB. Ces blocs d'entrée/sortie permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils 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 inutilisé (haute impédance).



FigureII.3 : le bloc IOB de la famille XC4000X.

4.1. CONFIGURATION EN ENTREE

Premièrement, le signal d'entrée traverse un buffer qui selon sa programmation peut détecter soit des seuils TTL ou des seuils CMOS. Il peut être routé directement sur une entrée directe de la logique du circuit FPGA ou sur une entrée synchronisée. Cette synchronisation est réalisée à l'aide d'une bascule de type D, le changement d'état peut se faire sur un front montant ou descendant. De plus, cette entrée peut être retardée de quelques nanosecondes pour compenser le retard pris par le signal d'horloge lors de son passage par l'amplificateur. Le choix de la configuration de l'entrée s'effectue grâce à un multiplexeur (program controlled multiplexer). Un bit positionné dans une case mémoire commande ce dernier.

4.2. CONFIGURATION EN SORTIE

Nous distinguons les possibilités suivantes :

- inversion ou non du signal avant son application à l'IOB,
- synchronisation du signal sur des fronts montants ou descendants d'horloge,
- mise en place d'un « pull-up » ou « pull-down » dans le but de limiter la consommation des entrées sorties inutilisées,
- signaux en logique trois états ou deux états. Le contrôle de mise en haute impédance et la réalisation des lignes bidirectionnelles sont commandés par le signal de commande Out Enable (T) lequel peut être inversé ou non. Chaque sortie peut délivrer un courant de 12mA. Ainsi toutes ces possibilités permettent au concepteur de connecter au mieux une architecture avec les périphériques extérieurs.

5. LES DIFFERENTS TYPES D'INTERCONNEXIONS [7]

Les ressources de routage associées avec chaque CLB sont schématisées en figure II.4. Cette structure se répète d'un CLB à l'autre de manière à obtenir une structure régulière bien adaptée aux algorithmes des outils de routage.

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, celles-ci sont assurées par 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 plusieurs types d'interconnexions selon la longueur et la destination des liaisons. Nous disposons :

- d'interconnections selon la longueur,
- d'interconnexions d'E /S,
- d'interconnexions globales,
- d'interconnexions directes.

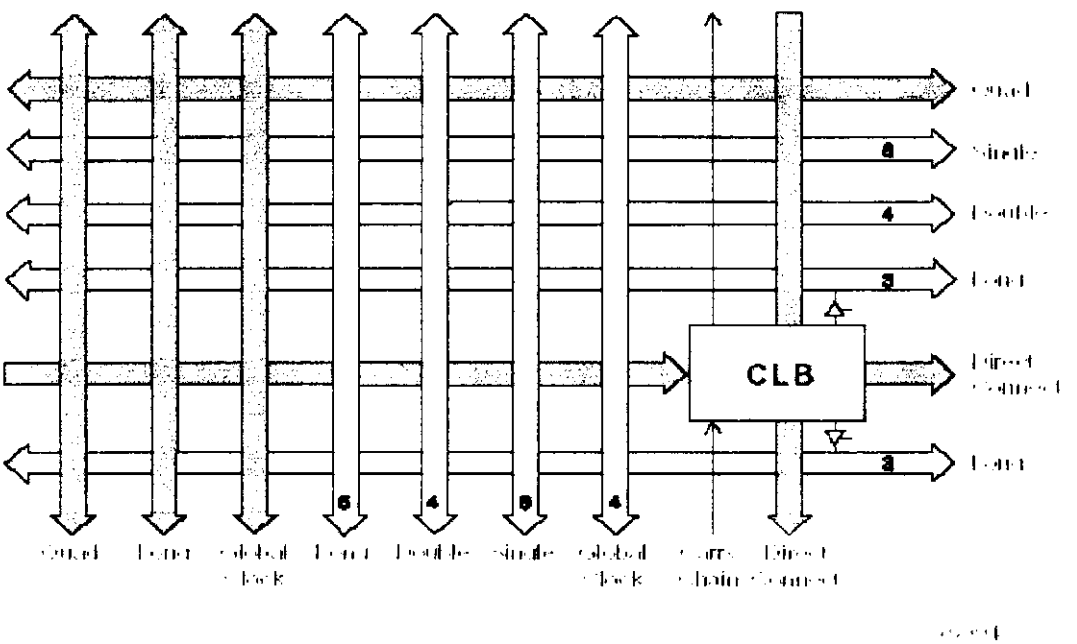


Figure II.4 : Les ressources de routage associées à chaque CLB

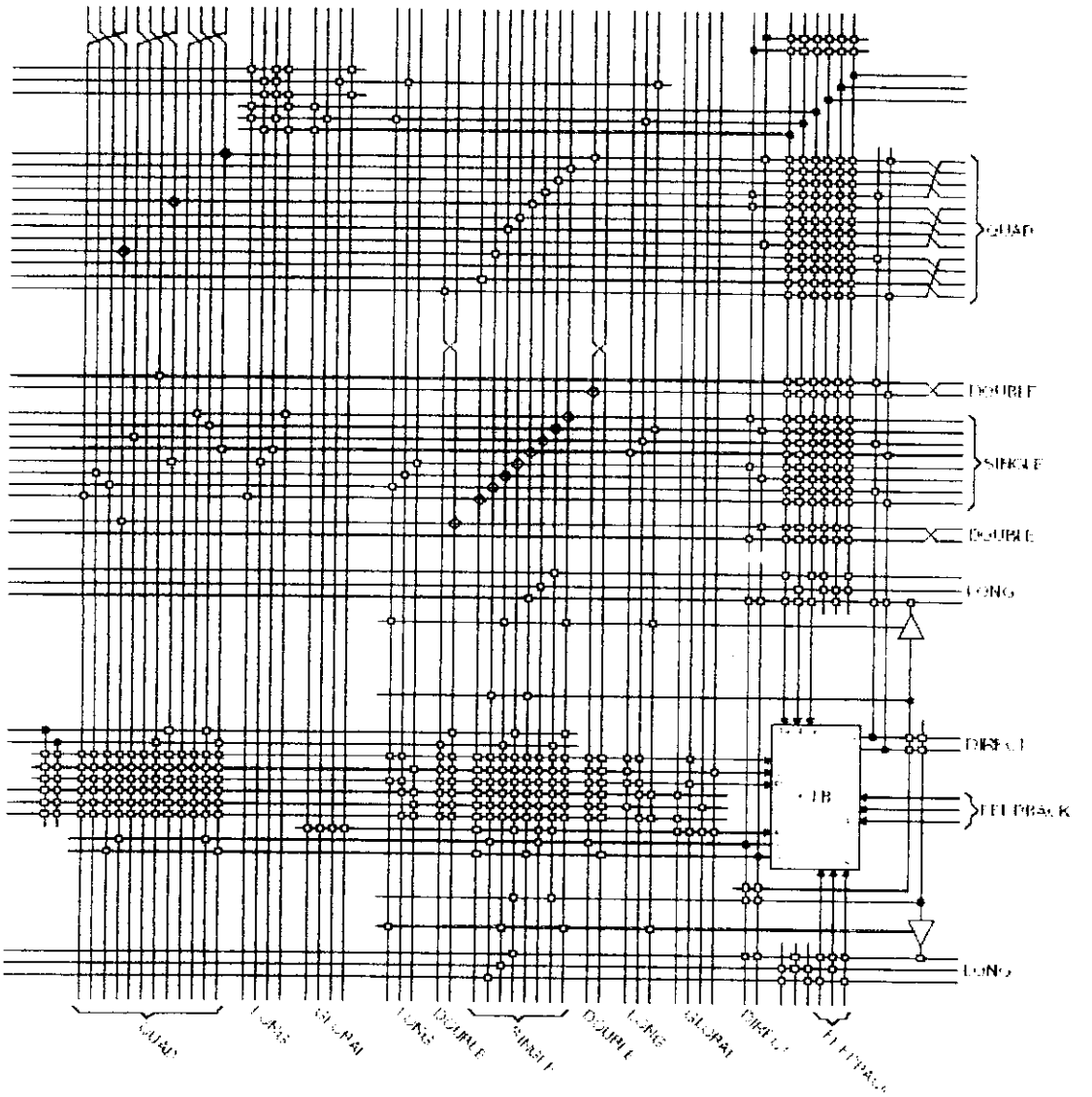


Figure II.5 : L'ensemble de ligne d'interconnexion associés à chaque CLB de la série XC4000X.

5.1. PERFORMANCES DES INTERCONNEXIONS

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions globales, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connexion de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ils possèdent une faible résistance mais une capacité importante. De plus, si on utilise un aiguilleur, sa résistance s'ajoute à celle existante.

6. LES DECODEURS ETENDUS [7]

Il s'agit ici d'une ressource atypique des FPGAs du type SRAM. Sur chaque bord du circuit est implanté un décodeur réalisant quatre termes produit (comme dans les circuits PAL). Les variables d'entrée sont constituées de signaux issus de broches situées sur le bord concerné. Il est ainsi possible de réaliser de très large 'ET'. Cette fonctionnalité sera très utile pour des fonctions de décodage nécessitant des critères de vitesse sévères. Les quatre termes produits peuvent être dirigés aussi bien vers les IOB que vers les CLB (et réaliser par exemple la sommation de ces termes produits). La figure II.6 donne une vue partielle du bord gauche du circuit.

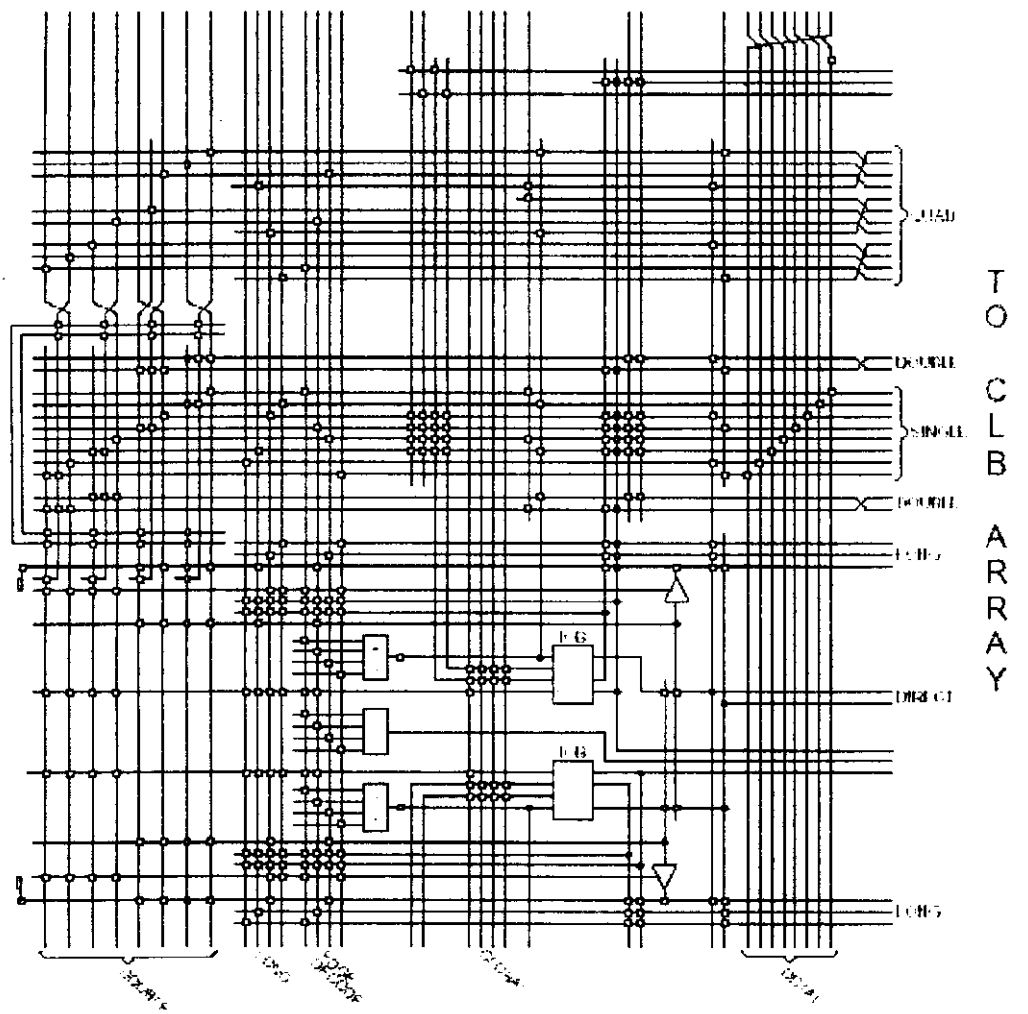


Figure II.6 : Une vue partielle du bord gauche du circuit.

7. OSCILLATEUR INTERNE

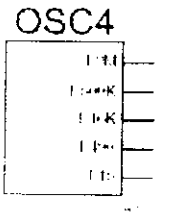


Figure II.7: Symbole d'oscillateur de la série xc4000

Lors du mode maître de configuration, l'oscillateur interne du FPGA maître génère l'horloge aux autres esclaves (ROM, FPGA,...), afin de charger des données d'une ROM, ou bien pour configurer des FPGAs esclaves, mis en série, au rythme de l'horloge CLK.

8. LE RESEAU D'HORLOGES [7]

Huit broches d'entrées sont spécifiquement dédiées à des signaux d'horloge. Le signal de chacune de ces broches est dirigé vers deux buffers de nature différente. Les huit buffers appelés BUFGLS sont capables de supporter un très fort taux de charge et les horloges qu'ils génèrent sont accessibles par tous les CLB et les IOB du composant. Les huit buffers appelés BUFGE supportent un taux de charge moins élevé mais leur temps de transfert est plus faible que celui des BUFGLS. Pour chacun des deux types de buffers, l'horloge a une très faible variation de phase, ce qui est important pour concevoir des systèmes synchrones. Il est ainsi possible de synchroniser l'ensemble du composant à partir d'une horloge fournie sur une des broches GCK1 à GCK8.

Quatre autres broches spécifiques sont dédiées à des horloges appelées FASTCLK et disposent d'un buffer très rapide (BUFFCLK). Se référer à la figure II.8. Chacune de ces horloges n'accède qu'à certains blocs IOB.

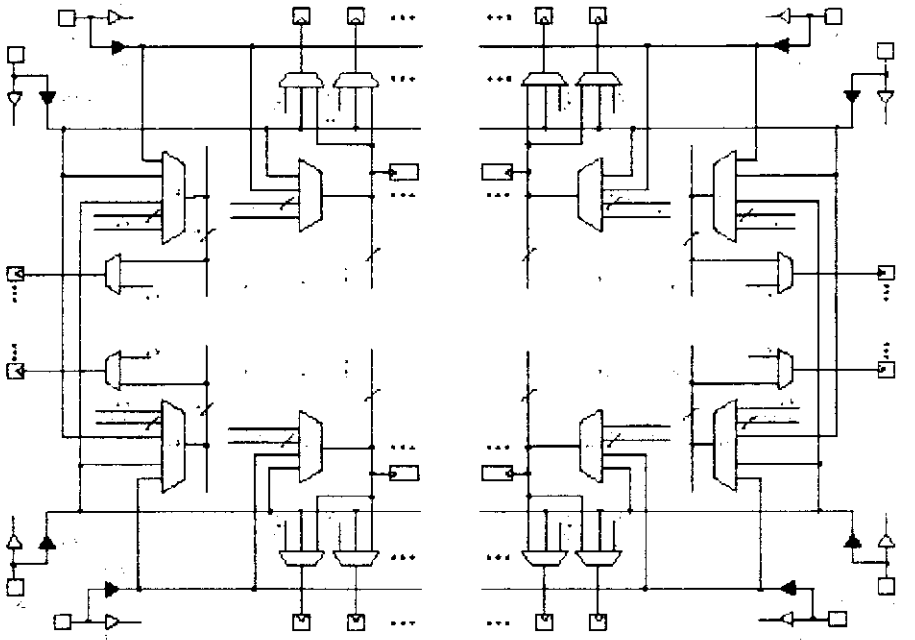


Figure II.8 : Le réseau d'horloges

9. SOURCE DE TENSION [7]

La distribution de l'alimentation dans le composant est disposée suivant une grille spécifique, cela assure alors une grande immunité au bruit ainsi qu'une isolation entre la logique et les entrées/sorties à l'intérieure du FPGA. Pour avoir un découplage adéquat, des capacités sont montées entre V_{CC} et GND.

10. BROCHAGE [7]

Trois différents type de broches sont fournis dans le cas de la famille XC4000 :

- Des broches dédiées de façon permanente à des tâches déterminées ;
- Broches E/S qui peuvent être aussi utilisées dans des fonctions spécifiques.
- Broches E/S.

-Avant ou durant configuration, les broches non utilisées sont en état haute impédance.

-Les fonctions S/R ou GSR peuvent être associées à n'importe quelles broches E/S.

11. BOUNDARY SCAN [7], [8]

Fonctionnement normal, programmation et test : l'idée s'est imposée d'incorporer ces trois modes de fonctionnement dans les circuit eux-mêmes, comme partie intégrante de leur architecture. Pour le test de cartes, une norme existe : le standard IEEE 1149.1 plus connu sous le nom de ***boundary scan*** du consortium JTAG (*join test action group*). Face à la quasi-impossibilité de tester de l'extérieur les cartes multicouches avec des composants montés en surface, un mode de test a été défini, pour les VLSI numériques. Ce mode de test fait appel à une machine d'état intégrés dans tous les circuits compatible JTAG, qui utilise cinq broches dédiées :

- TCK, une entrée d'horloge dédiée au test, différente de l'horloge du reste du circuit.
- TMS, une entée de mode qui pilote l'automate de test.
- TDI, une entrée série.
- TDO, une sortie série.
- TRST (optionnelle), une entrée de réinitialisation asynchrone de l'automate

L'utilisation première de ce sous-ensemble de test est la vérification des connexions d'une carte. Le mode test est activé, via des commandes sur les entrées TMS et TRST, le fonctionnement normal du circuit est inhibé. Les broches du circuit sont connectées à des cellules d'entée/sortie dédiées au test, chaque cellule est capable de piloter une broche en sortie et de capturer les données d'entrée, conformément au schéma de principe de la figure II.9.

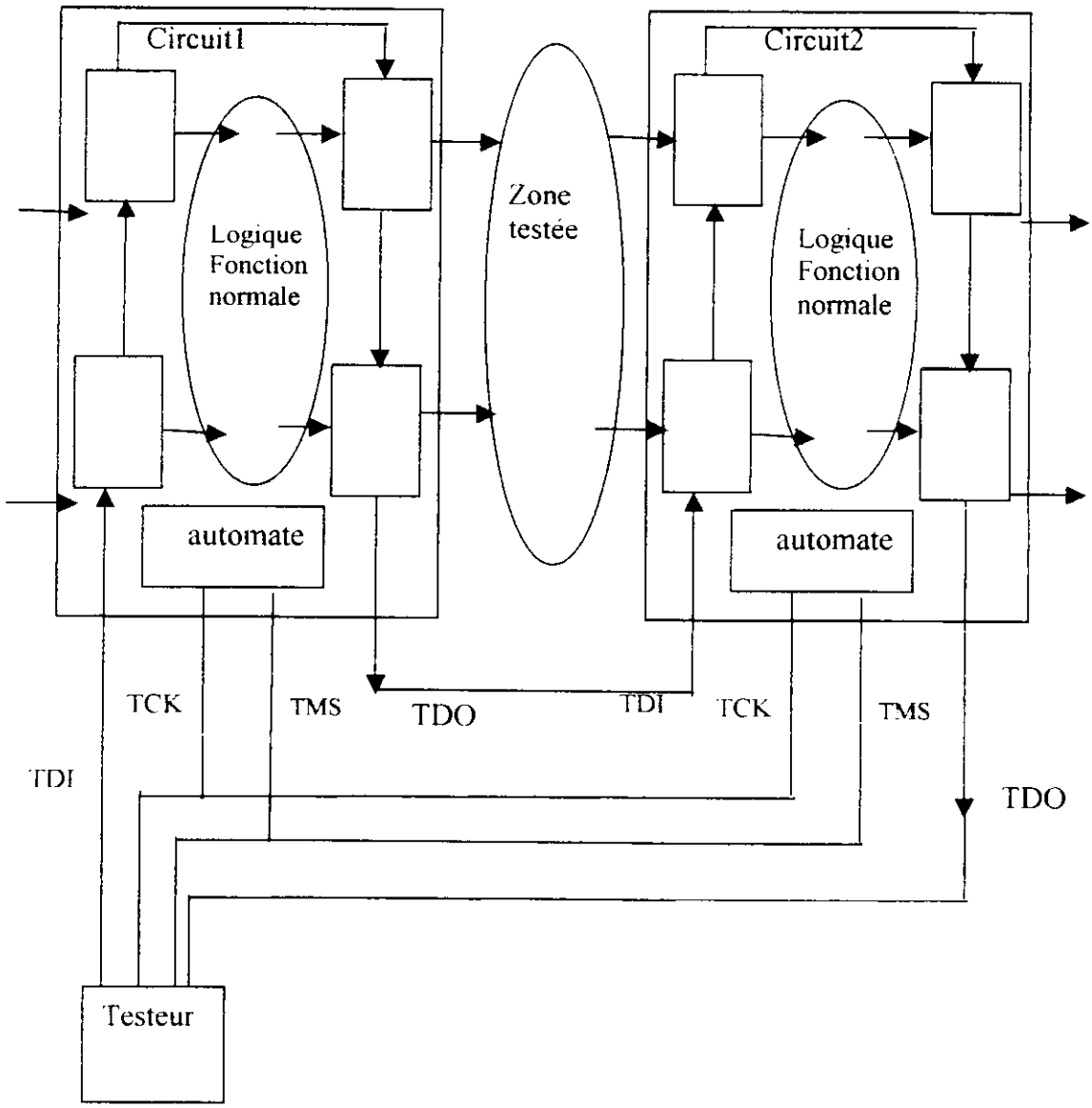


Figure II.9: Boundary scan

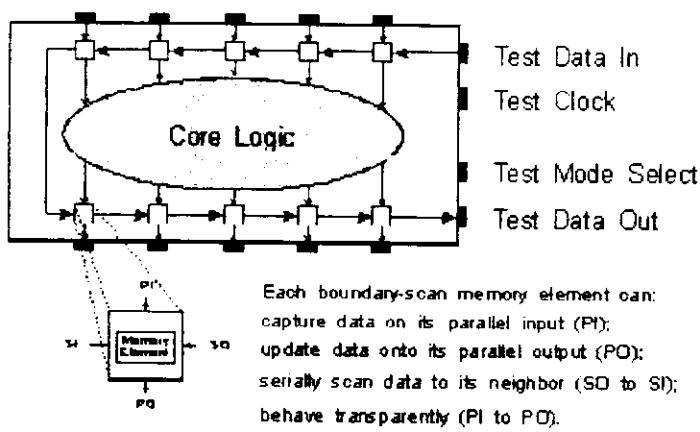


FIGURE II.10: PRINCIPE DE FONCTIONNEMENTDU JTAG

12. MODE DE CONFIGURATION D'UN FPGA [7]

12.1. CONFIGURATION

La configuration est le processus de chargement des données de programmation pour une conception spécifique dans un ou plusieurs FPGA pour rendre fonctionnel un nombre déterminé de blocs logiques internes. Chaque bit de configuration définit l'état d'une cellule SRAM qui peut contrôler un LUT, multiplexeur ou un transistor d'interconnexion.

La manière dont un FPGA est configuré dépend des valeurs de M0, M1, M2, qui peuvent définir 3 principaux modes : **maître, esclave, périphérique** associés aux modes **série, parallèle (up, down), synchrone, asynchrone, daisy chaine ;** Un composant de la famille XC 4000 aura 6 différents modes de configuration

M0	M1	M2	Mode sélectionné	CCLK
0	0	0	maître série	Sortie
0	0	1	maître parallèle haut	Sortie
1	1	0	maître parallèle bas	Sortie
1	0	1	Périphérique Asynchrone	Sortie
0	1	1	Périphérique Synchrone	Entrée
1	1	1	esclave série	Entrée

Tableau II.1: Modes de configurations

Les circuits FPGA ne possèdent pas de programme résident. A chaque mise sous tension, il est nécessaire de les configurer. Leur configuration permet d'établir des interconnexions entre les CLB et IOB. Pour cela, ils disposent d'une RAM interne dans laquelle sera écrit le fichier de configuration. Le format des données du fichier de configuration est produit automatiquement par le logiciel de développement sous forme d'un ensemble de bits organisés en champs de données. Le FPGA dispose de six modes de chargement et de trois broches M0, M1, M2 (trois bits d'entrée) lesquelles définissent les différents modes. Ces modes définissent les différentes méthodes pour envoyer le fichier de configuration vers le circuit FPGA, selon deux approches complémentaires :

- configuration automatique, le circuit FPGA est autonome,
- configuration externe, l'intervention d'un opérateur est nécessaire.

12.1.1. MODE MAITRE SERIE

Le mode maître série utilise une mémoire à accès série de type registre à décalage. Le programme est préalablement chargé par le système de développement utilisé pour Le circuit FPGA. Le FPGA génère tous les signaux de dialogue nécessaires pour la copie du contenu de la PROM dans sa RAM interne, lorsque la copie est terminée, il bascule le signal DONE pour le signaler au circuit. Comme nous pouvons le remarquer sur la figure II.11, une seule PROM peut configurer plusieurs circuits FPGA avec la même configuration ou plusieurs PROM peuvent configurer plusieurs FPGA en chaîne où le premier des circuits FPGA est le maître et génère l'horloge. Les données en provenance des PROM sont envoyées aux autres circuits FPGA par la sortie DOUT de ce premier (figure II.12).

12.1.2. MODES MAITRE PARALLELE

On dispose aussi de deux modes maître parallèle haut et modes maître bas où le FPGA est relié en parallèle à une EPROM classique, de même qu'un mode passif type périphérique dans lequel le FPGA est considéré comme un périphérique de µP et peut-être configuré à partir de celui-ci. Dans le mode maître parallèle bas, ce mode est sélectionné en plaçant les entrées M2, M1, M0 respectivement à <110>. Les adresses de l'Eprom débutent à 3FFFF et décrémente. Dans le mode maître parallèle haut, ce mode est sélectionné en plaçant les entrées M2, M1, M0 à <100>. Les adresses de l'Eprom débutent à 00000 et increment.

12.1.3. MODE ESCLAVE SERIE

Dans ce mode, le programme de configuration peut être envoyé à partir d'un PC, d'une station de travail ou à partir d'un autre circuit FPGA. Le circuit FPGA maître peut être interfacé à une mémoire en mode parallèle ou série sans apporter aucune modification au niveau du câblage des circuits FPGA esclaves. C'est souvent le mode exploité pour la mise au point d'une configuration.

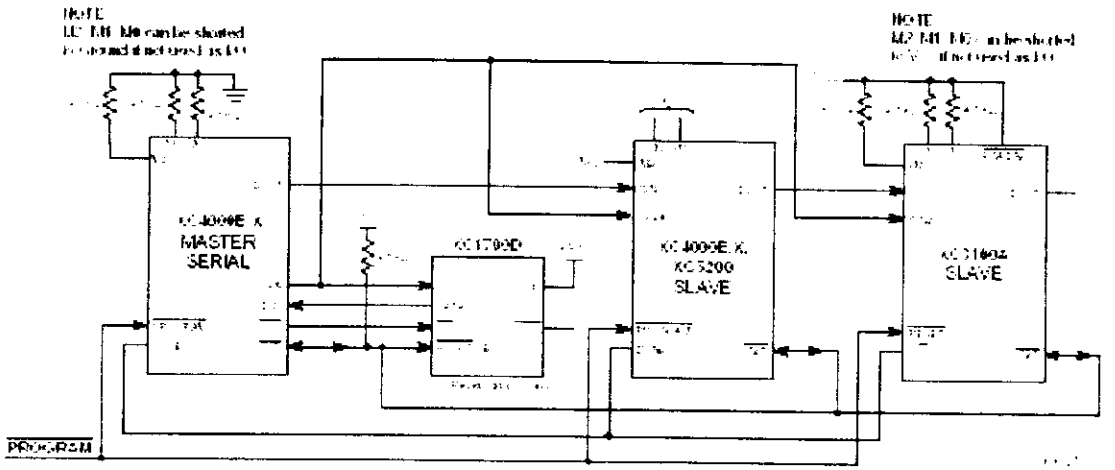


Figure II.11 : Mode esclave/ maître série

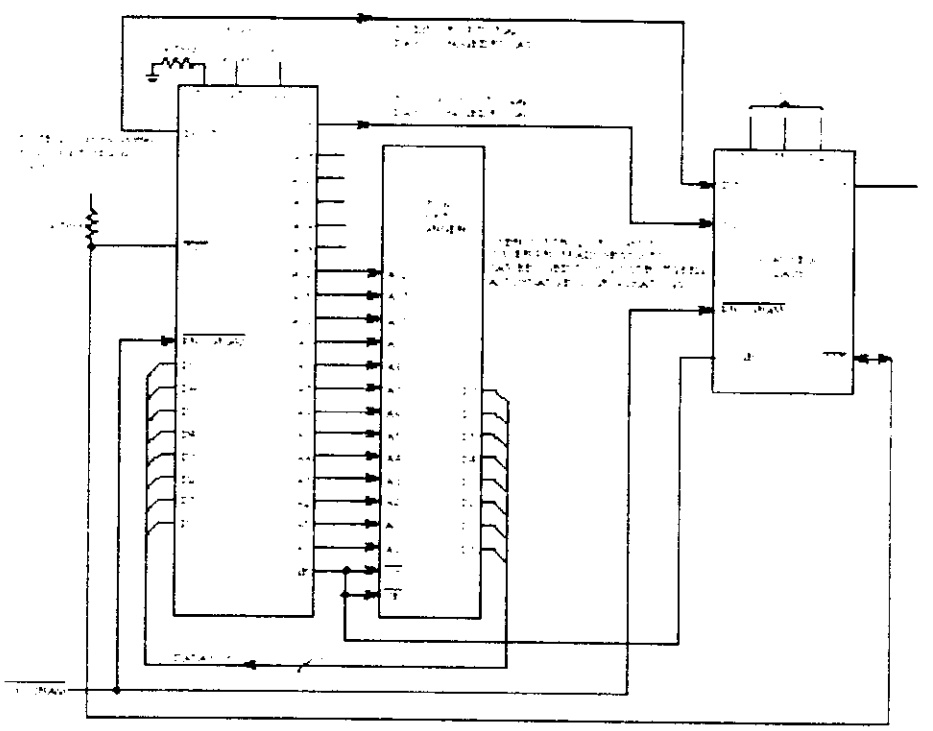


Figure II.12 : Mode maître parallèle

12.1.4. MODE PERIPHERIQUE SYNCHRONE

Le mode périphérique synchrone peut-être considéré aussi comme étant un mode parallèle esclave. Un signal externe pilote l'entrée CCLK des FPGAs. Le premier octet de

données d'entrée de la configuration parallèle doit être disponible quelque instant avant le front montant du signal CCLK.

Le même front d'horloge accepte les données et pousse aussi le signal RDY/$\overline{\text{BUSY}}$
A l'état haut pour un cycle d'horloge.
La ligne RDY/$\overline{\text{BUSY}}$ se mis à l'état haut pour chaque cycle d'horloge CCLK après avoir reçu la donnée d'entrée.

Le FPGA en mode périphérique synchrone s'occupe du transfert en série de ces données à travers sa sortie DOUT après 1.5 cycles d'horloge CCLK une foie la donnée est reçue par le circuit FPGA en mode périphérique synchrone.

Le mode périphérique synchrone est sélectionné par le <011> des broches (M2, M1, M0) Respectivement.

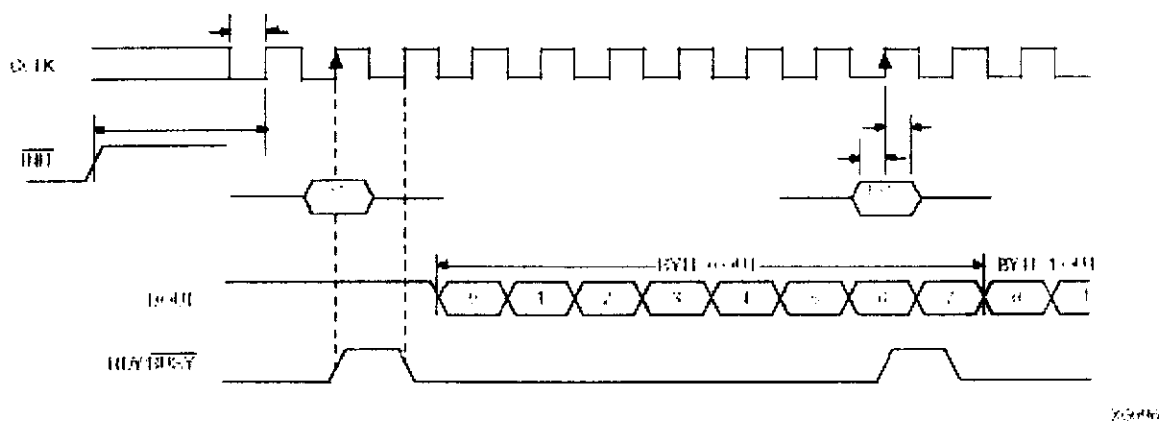


Figure II.13 : Caractéristiques temporelles du mode périphérique synchrone

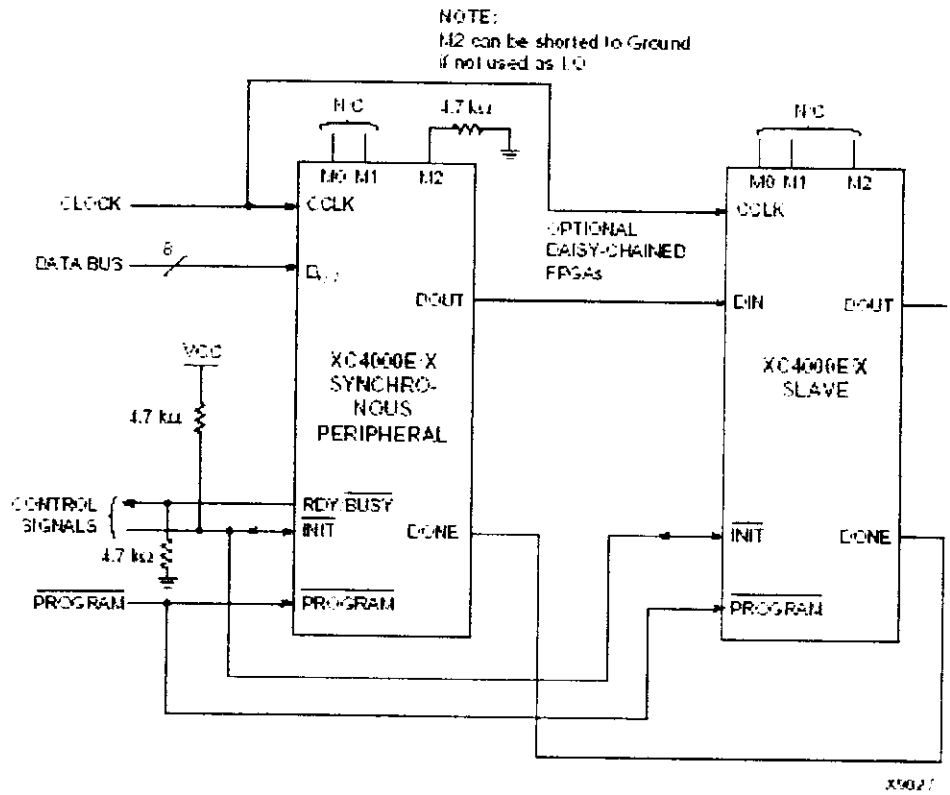


Figure II.14:Diagramme du circuit du mode périphérique synchrone

12.1.5. MODE PÉRIPHÉRIQUE ASYNCHRONE

- **Ecriture dans le FPGA**

Le mode périphérique asynchrone emploi le déclenchement sur front de la condition de la logique AND, pour laquelle /WS et /CS0 étant à l'état bas, et /RS et CS1 étant à l'état haut afin d'accepter l'octet de données prévenant du bus de données du microprocesseur.
Puis ces données parallèles sont converties en données sous forme série décalées dans la logique interne.

Le FPGA présente les données reçues sur la broche de sortie DOUT.
RDY/$\overline{\text{BUSY}}$ passe à zéro au moment de la réception de l'octet, et reste à 1 pendant que le buffer d'entrée qui reçoit les données, s'occupe du transfert de ces données dans le registre à décalage, et de nouveau le buffer d'entrée est prés à recevoir de nouvelles données.
Et l'écriture ne peut pas être terminée avant que RDY/$\overline{\text{BUSY}}$ revienne à l'état haut.

La longueur du signal RBY/$\overline{\text{BUSY}}$ dépend de l'activité interne (registre de décalage). Si le registre à décalage était vide quand le nouvel octet a été reçu, le signal $\overline{\text{BUSY}}$ dure pendant deux cycles d'horloge CCLK. Par contre si le registre à décalage était encore plein quand le nouveau octet a été reçu le signal $\overline{\text{BUSY}}$ peut être aussi long que neuf périodes CCLK.

- **Statut de lecture**

Les conditions de la logique AND du /CS0, CS1, /WS, et /RS met le statut du dispositif sur le bus de données.

- D7 à l'état haut indique prêt ;
- D7 à l'état bas indique occupé ;
- D0 jusqu'à D6 passent sans réserve à l'état haut.

D7 représente le statut de RDY/$\overline{\text{BYSY}}$ quant RS est bas, WS est haut, et les deux lignes chip select (CS1, CS0) sont actives.

Le mode périphérique asynchrone est sélectionné par le <101> des broches (M2, M1, M0) respectivement.

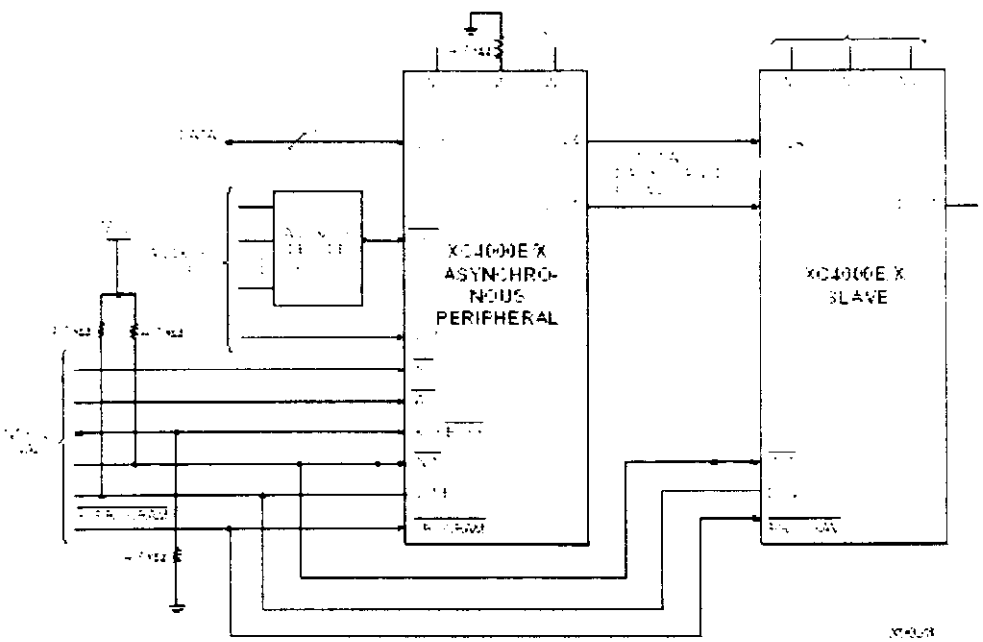
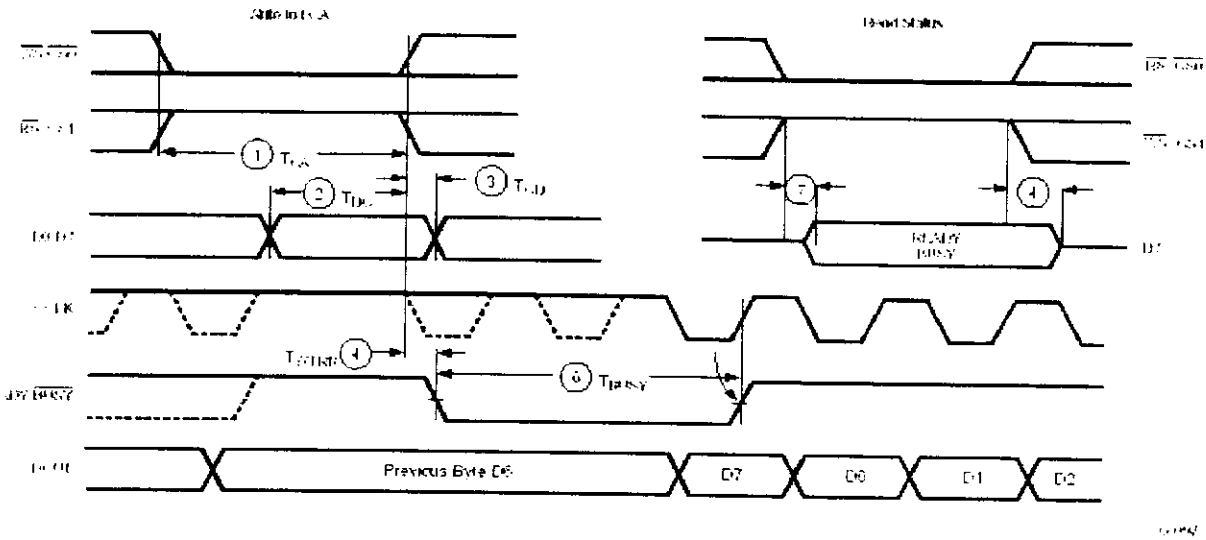


Figure II.15: Diagramme du circuit du mode périphérique asynchrone



FigureII.16: Caractéristiques temporelles du mode périphérique Asynchrone

12.2. CYCLIC REDUNDANCY CHECK (CRC)

CRC est une méthode destinée à la détection d'erreurs lors de la transmission des données .à la fin de chaque trame de donnée de configuration on conçoit quatre bits d'erreurs calculés par le système émetteur et vérifiés par le FPGA (un calcul identique que celui de la source est effectué par le FPGA) .Si une erreur est détectée, le processus de configuration est arrêté suivi d'un état d'attente (/INIT=0).

12.3. SEQUENCE DE CONFIGURATION

Pour pouvoir configurer un FPGA, on doit passer par les 4 étapes suivantes :

- Clear de la mémoire de configuration ;
- Initialisation
- Configuration
- Démarrage (Start up)

12.4. CLEAR DE LA MEMOIRE DE CONFIGURATION

Les composants de la famille XC4000 disposent d'un circuit interne destiné au clear de la mémoire de configuration trame par trame soit après :

- la mise sous tension et que cette dernière atteint un niveau Vcc suffisant ; le circuit alors génère un signal test de la valeur M0 et une pulsation dont la durée dépend de M0 par exemple si M0=0(mode Maître) cette durée est 4 fois plus importante que l'autre cas (pour permettre aux différent esclaves d'atteindre une tension stable)

- ou après détection d'un niveau bas sur la broche /PROGRAM.

Remarque :

Le clear de la mémoire se continue tant que le niveau bas est maintenue sur /PROGRAM ou la durée de la pulsation n'est pas encore expirée ; une opération test est alors lancer à chaque fin du clear d'une trame mémoire.

12.5. INITIALISATION

L'état du composant pendant l'étape d'initialisation et de configuration est donné par les sorties HDC, /LDC, /INIT, DONE (resp. 1000).

Après le dernier clear d'une trame mémoire, un basculement de l'/INIT vers l'état Haut permet la préparation de la configuration (dans le cas du mode ''maître'' ce passage est retardé d'une durée de 30 à 300µs) qui consiste en un chargement du mode configuration ainsi que l'activation des lignes d'interface

12.6. CONFIGURATION

Cette étape commence par un code préambule "0010" qui indique que les bits suivants (24 bits) définissent le nombre de cycle d'horloge nécessaire au chargement des données de configuration ; et DOUT est mis à l'état haut pour que les autres composants de la chaîne ne soient pas affecté par les données de configuration

Chaque trame mémoire de configuration est composée d'un ensemble de bits pour la définition de l'états des cellules SRAM ainsi qu'un code de détection d'erreur. Si une erreur est survenue le chargement des données est arrêté momentanément et la sortie /INIT indique cette erreur .

12.7. START_UP

C'est une étape de transition entre l'étape de configuration et celle qui met le circuit ainsi obtenu dans son état opérationnel.

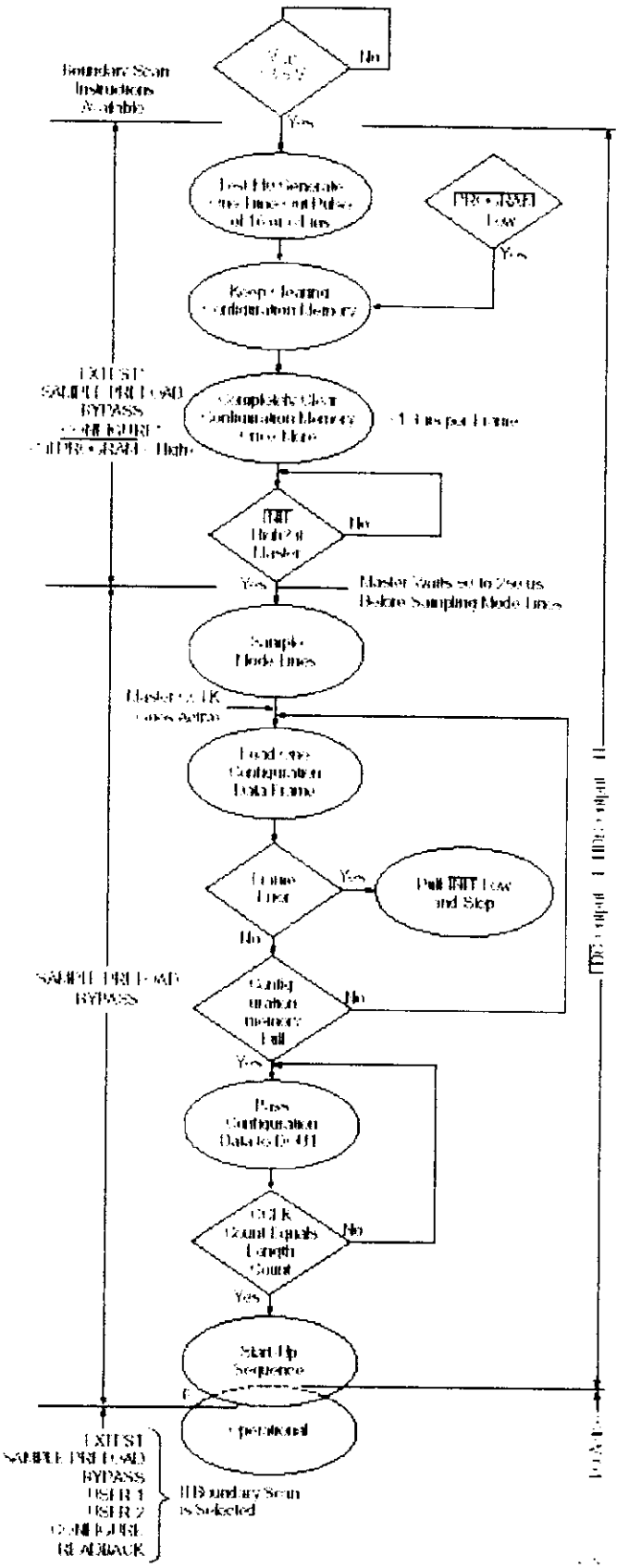


Figure II.17: Processus de configuration

13. CONCLUSION

On vient de traiter dans ce chapitre deux parties importantes. La première consiste en l'étude des différents blocs de base constitutifs du circuit XC4000 ; la seconde représente l'automate dédié à la communication, le test et la programmation du circuits FPGA afin d'aborder l'étude de la carte XS40 à base du circuit FPGA XC4005XL aussi bien sous l'angle du fonctionnement normal qu'en mode test ou en mode configuration.

INTRODUCTION

Dans ce chapitre, nous abordons l'étude d'une carte d'implémentation XS40 à base du circuit FPGA XC4005XL. Cette étude consiste en la description des différents éléments de base constitutifs de cette carte ainsi que son fonctionnement et sa programmation.

La demande croissante des différents circuits à architecture programmable en particulier les FPGA est motivée par les nombreux avantages qu'offrent ces composants à savoir :

- système de taille réduite
- coût réduit
- accroissement de performances
- sécurité
- grande flexibilité

Ce dernier avantage est possible en utilisant simplement un simple ordinateur de type PC qui communique avec une **carte** qui contient le composant en question.

Ce chapitre alors sera consacré à l'étude de cette carte ainsi que la présentation des deux différents modèles utilisés.

2. TYPE DE CARTE

La programmation d'un FPGA nécessite un flot de données "bitstream" spécifique à la cible généré par un outil de développent associé. Ce bitstream est transmis à la cible soit directement par un connecteur PC ou via une mémoire ROM (programme fixe). La première solution est la plus utilisée car elle offre une souplesse d'utilisation. Un troisième type de carte possible est le résultat de la combinaison des deux premières.

Un autre type de carte est celle qui repose sur la présence d'un microcontrôleur. Cette carte permet en plus des conceptions de FPGA de réaliser :

- des applications de microcontrôleur
- ou des hardware /software codesign

3. ETUDE DE LA CARTE[16]

Cette étude concerne une carte XS40 de type carte sans microcontrôleur disponible au niveau du laboratoire.

La XS40 est une carte qui permet de mettre en application des conceptions pour circuits FPGA. La carte XS40 contient les composants additionnels tels qu'une SRAM de 32 K octets, un oscillateur de 12 mégahertz, un port parallèle, un port PS/2 pour une souris ou un clavier, et un port de moniteur VGA. Ceci réalise un système commode pour mettre en application des conceptions de FPGA avec des SRAM pour le stockage de données (brutes ou de configuration). La carte peut être programmée en utilisant le port parallèle. Le FPGA peut également être examiné en utilisant l'utilitaire XSPORT qui permet d'appliquer des signaux d'essai en utilisant le port parallèle.

Un schéma général de la carte représentant les éléments de base de cette carte est représenté sur la figure III.1.

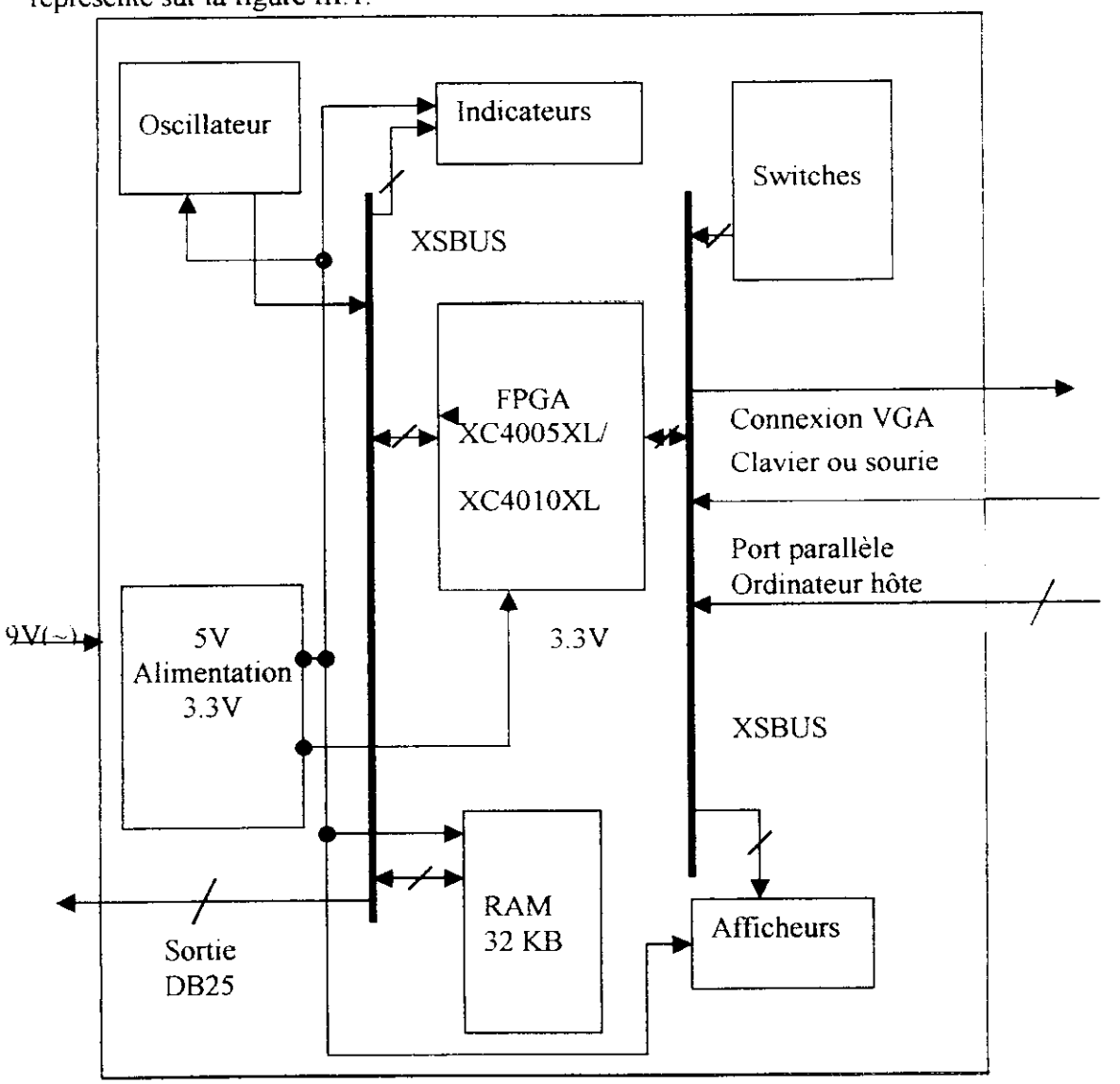


Figure III.1 : Le schéma général de la carte

On va exposer dans ce qui suit les blocs de base que comporte cette carte.

3.1. ALIMENTATION DE LA CARTE

La carte XS40 est alimentée par une tension 9V DC (courant continu). La carte a deux régulateurs de tension qui produisent du 3.3V pour le FPGA et le 5V pour le reste des composants. L'alimentation de la carte est représentée dans la figure suivante :

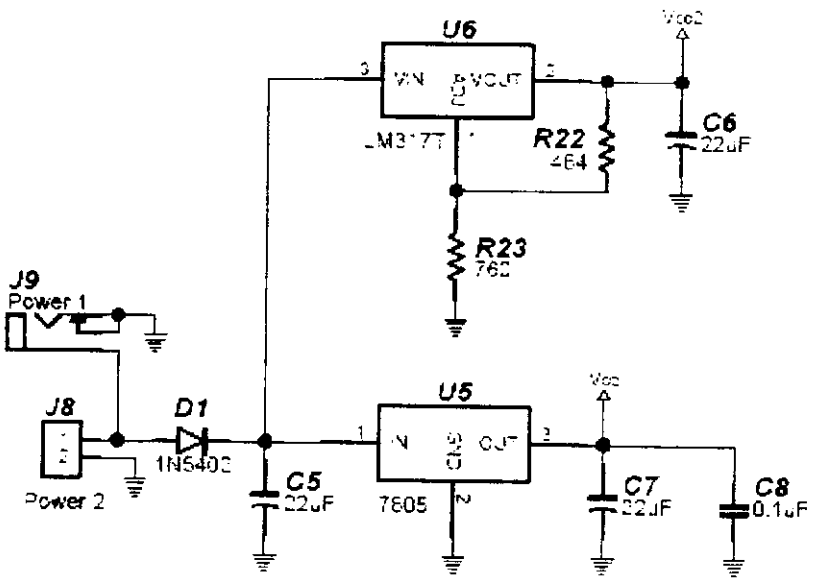


Figure III.2 : Alimentation de la carte

3.2. LE CIRCUIT FPGA XC4005XL

Ce circuit appartient à la famille XC4000 étudiée dans le chapitre précédent.

Le FPGA XC4005XL représente le cœur de la carte XS40.

Une fois que le circuit FPGA est placé sur la carte, on aura accès aux différentes broches du circuit. La première chose à faire est de connecter l'alimentation VCC (3.3 V), et la masse GND au circuit.

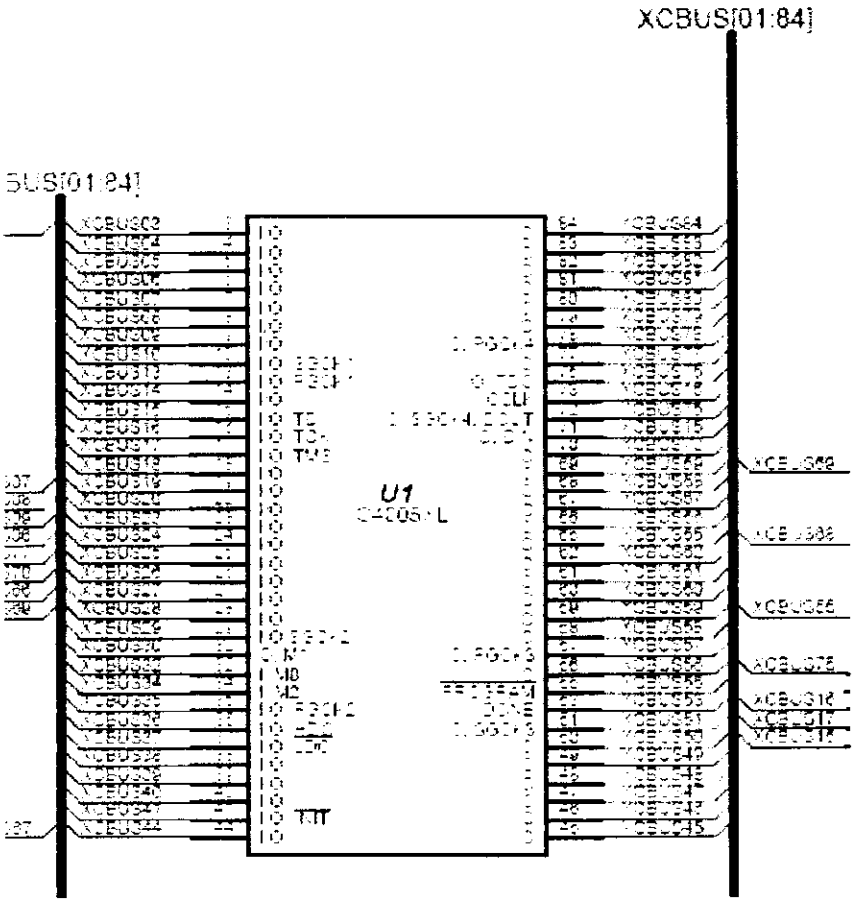


Figure III.3: Le circuit FPGA XC4005XL

3.3. LE PORT PARALLELE

La carte XS40 est programmée à travers le port parallèle du PC, à l'aide d'un connecteur femelle DB25 sur le XS40. Le PC peut alors transmettre des signaux à la carte XS40 pour programmer le FPGA ou l'examiner plus tard.

Les broches 1, 14, 17 du port parallèle sont utilisées pour configurer le circuit FPGA (télécharger le train de bits dans le circuit FPGA).

Une impulsion de niveau bas sur la broche 1 du port parallèle est transmise à l'entrée /PROGRAM du FPGA (broche 55). Celle-ci initialise le FPGA et le prépare pour accepter une autre configuration, comme expliqué dans le chapitre II (pins du circuit XC4000). Le train de bits est transmis à travers l'entrée 17 du port parallèle, vers l'entrée DIN (broche 71 du FPGA). Le chargement des données de configuration se fait au rythme du signal d'entrée d'horloge CCLK (broche 73). CCLK est pilotée par l'entrée 14 du port parallèle du PC, passant à travers le trigger de Schmitt et le troisième jumper de J4.

La borne 16 du port parallèle est connectée à TMS (la broche 17) du circuit FPGA XC4005XL.

Les broches 14 et 17 du port parallèle sont également connectées aux broches d'entrée TCK et TDI du port JTAG, puisque TCK est connecté au CCLK et TDI connecté au DIN.
Ces broches (14, et 17) sont utilisées pour mémoriser des données dans la mémoire SRAM de 32 K octets.
Les broches 2-9 du port parallèle sont utilisées pour appliquer des signaux de test, après avoir configuré un nouveau circuit. Les signaux appliqués sur les bornes 2 et 3 du port parallèle sont de préférence utilisés pour router des signaux d'horloge pour des circuits séquentiels.

3.4. LA MEMOIRE SRAM

Les connections entre le FPGA XC4005XL et la SRAM de 32 K Octets sont présentées sur la figure suivante.
Le signal de validation chip select est mis à l'état haut pour neutraliser la RAM quand le circuit FPGA est configuré. Ceci nous permet de charger des données dans la RAM et configurer alors le FPGA sans corrompre les données de la RAM.

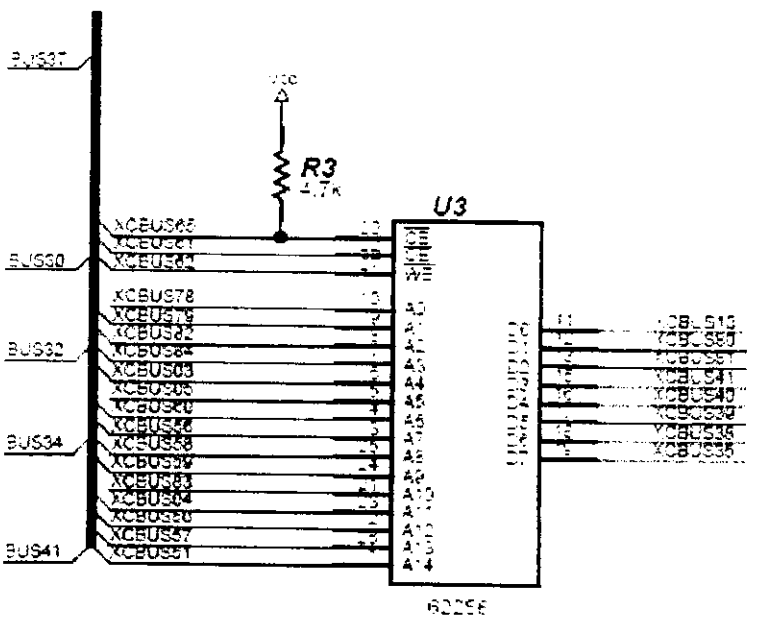


Figure III.4 : La mémoire statique (SRAM) de 32octets

3.5. L'OSCILLATEUR

Le circuit oscillateur présenté en figure III.5 génère un signal d'horloge de 12 MHZ pour le circuit FPGA XC4005XL. Le résonateur de 12 MHZ est connecté entre l'entrée et la sortie du premier inverseur. Les deux résistances polarisent l'inverseur pour assurer l'oscillation du résonateur. Le second inverseur applique le signal à l'une des entrées d'horloge globale primaire du circuit FPGA (PGCK1 à la borne 13).

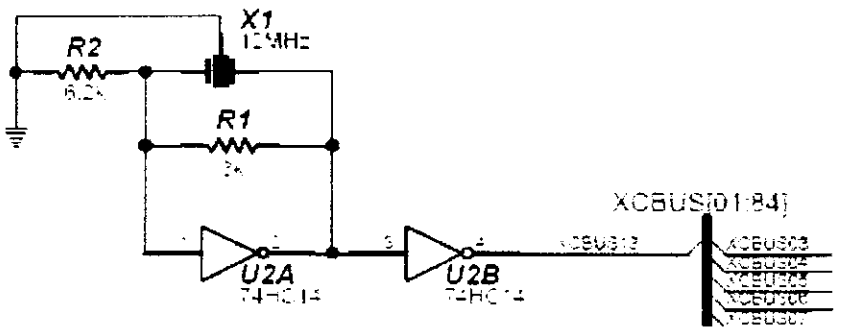


Figure III.5 : Le schéma de l'oscillateur

3.6. CONNECTEUR PS/2 POUR CLAVIER OU SOURIE (J5)

La carte XS40 a un connecteur (J5 sur le schéma de la carte) qui accepte des entrées d'un clavier ou d'une souris. On aura besoin d'un circuit de conducteur de clavier qui peut être téléchargé du homepage de XESS, (http://www.xess.com/). Les lignes de données et d'horloge, sont connectées aux bornes 69 et 68 du FPGA, respectivement.

3.7. CONNECTEUR AU MONITEUR VGA (J2)

La broche 15, connecteur (J2 sur le schéma de la carte) est utilisée pour relier un moniteur de VGA. Ceci peut être employé pour l'affichage sur le moniteur. Afin de faire ainsi, on devra d'abord télécharger un circuit de conducteur de VGA a la carte XS40.Les exemples des conducteurs sont disponibles sur le homepage de XESS Corp, (http://www.xess.com/).

Les deux connecteurs J2 et J5 sont intégrés dans la carte, afin de pouvoir réaliser un système complet répondant à certaines applications.

L'intégration des switches et des afficheurs dans la carte, nous permettent de vérifier des conceptions réalisées sur la carte, avec comme entrées les positions des switches, et comme sorties, l'état des afficheurs.

3.8. LES JUMPERS

Si on utilise la carte dans un environnement normal de conception pour télécharger des données de configuration d'un PC par le port parallèle, on utilise les jumpers tel qu'ils sont définis par défaut. Pour n'importe quel autre usage, choisir les arrangements des différents jumpers comme expliqué dans le tableau III.1.

Jumper	Arrange-ment	Description
J41	**ON** (defaut)	Placer le shunt dans le cas d'utilisation d'une seule carteXS40, ou si celle-ci (XS40) est la dernière carte mise en cascade avec d'autres cartes XS40 en chaîne.
	Off	Enlever le shunt sauf dans le cas de la dernière carte XS40 placée en cascade.
J43	**ON** (default)	Installer un shunt pour télécharger à travers la port paralléle1 du PC.
	Off	Enlever le shunt pour configurer le XS40 de l'EEPROM, qui est à bord de la carte.
J45	**ON** (default)	Le shunt doit être installé si la configuration de la carte se fait à travers le port parallèle du PC
	Off	Enlever le shunt, dans le cas où la carte XS40est configurée à partir d'une EEPROM série résidente sur la carte.
J6	On	Installer le shunt pour programmer l'EEPROM, qui est à bord de la carte.
	Off (default)	Enlever le shunt pour l'utilisation normal de la carte (télécharger le fichier de configuration dans le FPGA à travers le port paralléle1 du PC).
J8	On	Placer le shunt, dans le cas d'utilisation des 3.3V d'alimentation pour FPGA de la famille XC4000XL.
	Off	Enlever le shunt pour l'utilisation des 5V d'alimentation pour FPGA type XC4000E.
J10	On	Pour configurer la XS40 d'une EEPROM qui est à bord de la carte.
	Off (default)	Enlever le shunt pour configurer la carte à travers le port paralléle1du PC.

Tableau III.1 : Arrangement des jumpers [16].

4. TEST DE LA CARTE

On peut utiliser le XSPORT [3] pour tester la carte à travers le port parallèle du PC. Le XSPORT peut supporter jusqu'à huit entrées.

Par exemple pour mettre en entrée les valeurs binaires 1,0 et 1 aux trois bits les moins signifiants du port parallèle, on va utiliser la commande suivante :

C:\XCPROJ-V\NOM DU PROJET\XSPORT 101

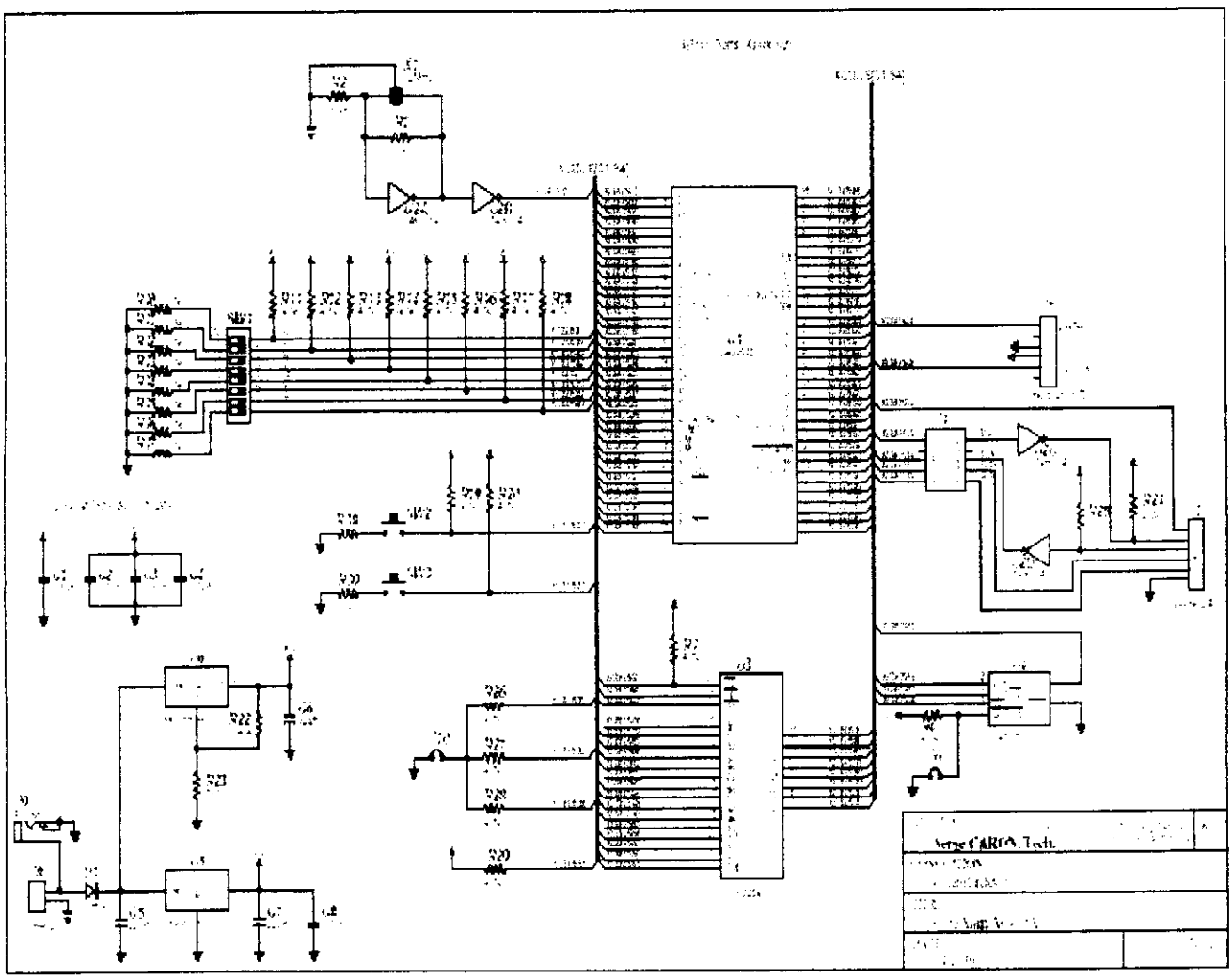


Figure III.6 : Diagramme schématique de la carte

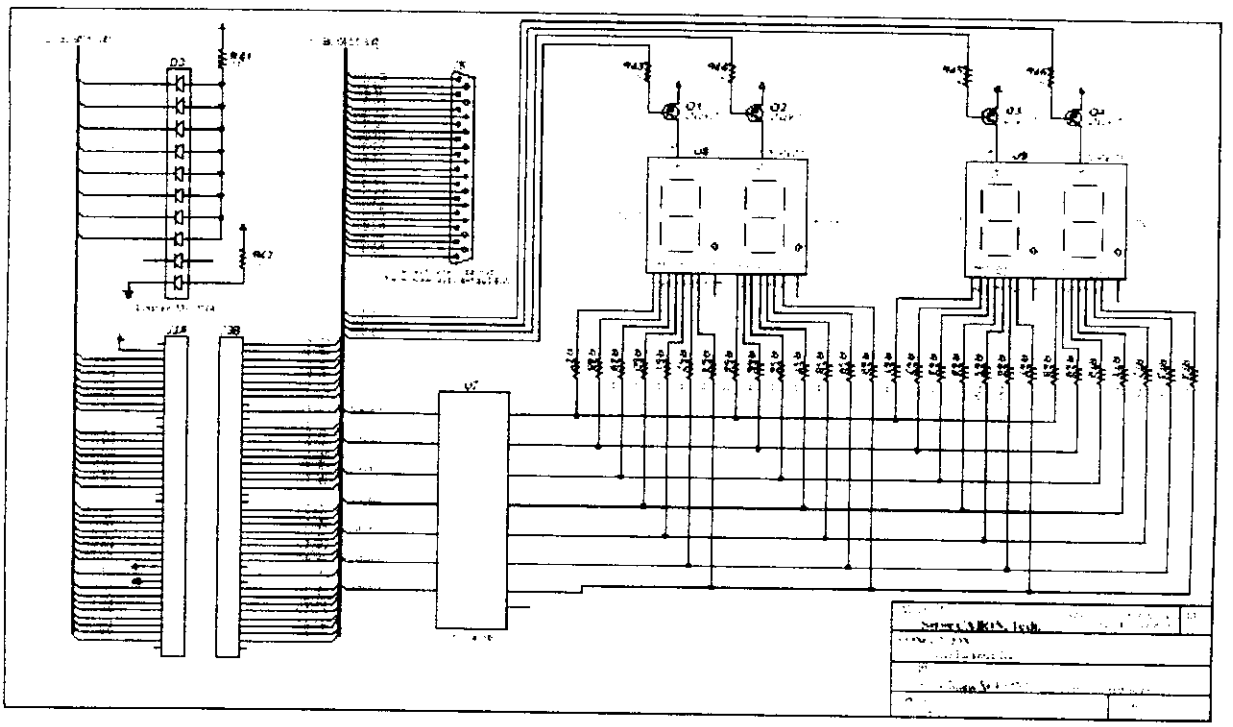


Figure III.7 : Suite du schématique de la carte

5. CONCLUSION

La carte étudiée dans ce chapitre a été réalisée à des fins pédagogiques mais constitue déjà une brique de base pour toute conception FPGA ciblant la famille XC4000. L'intégration d'un ensemble de switches, d'afficheurs, par exemple permet de vérifier des conceptions sur la carte, avec comme entrées les positions des switches, et comme sorties l'état des afficheurs. Cette carte peut être intégrée dans un système complet, constitué de clavier, de souris et d'écran afin d'implémenter des conceptions diverses.

L'exploitation de la carte ainsi étudiée est réalisée en chargeant des données de configuration associées à notre conception. Cela nous impose la maîtrise des techniques de synthèse architecturale, la méthodologie de conception ainsi que les outils de synthèse correspondants. Ces aspects seront entièrement traités dans le chapitre IV.

1. INTRODUCTION

Nous aborderons dans ce chapitre une application qui permettra la mise en œuvre de la carte XS40 étudiée dans le chapitre précédant, pour cela nous proposons de présenter en premier lieu la méthodologie de conception. Cette première partie du chapitre expose les différents outils et notions qu'un concepteur doit avoir pour réussir son projet. Nous entamerons ensuite la deuxième partie qui est consacrée à la conception et l'implémentation d'un codeur/décodeur.

1. INTRODUCTION

Depuis la naissance du premier circuit intégré dans les années soixante, le domaine de la microélectronique n'a cessé de se développer. Le perfectionnement constant des procédés de fabrication et des logiciels de conception assistée par ordinateur (CAO) a conduit aux densités d'intégration que l'on connaît aujourd'hui : jusqu'à dix millions de transistors par puce. Un cycle complet de fabrication a pour objectif l'obtention d'un circuit fiable, conforme aux spécifications initiales, respectant les exigences de moindre coût en un temps le plus court possible. La phase de conception du circuit intégré, partie intégrante de ce cycle qui précède la fabrication proprement dite, doit suivre cette évolution et s'adapter aux critères cités.

Ainsi, il est indispensable de vérifier *a priori* le comportement électrique du circuit qui va être soumis au fabricant : le fonctionnement est-il conforme à celui que l'on s'est fixé? Cette étape appelée simulation repose sur la connaissance de modèles des composants constitutifs du circuit.

La première génération des simulateurs commercialisés s'inspire essentiellement des principes et algorithmes de leur précurseur SPICE. Une nouvelle génération apparaît aujourd'hui, avec de nouveaux algorithmes optimisés pour simuler dans des délais raisonnables des circuits complexes.

En fait, cette stratégie a initialement été adoptée pour la conception des circuits numériques et a donné naissance au langage standard VHDL

2. DOMAINES D'EXPRESSION D'UNE ARCHITECTURE

Un circuit est destiné à résoudre un problème. On peut donc voir ce circuit d'une manière plus ou moins proche de la réalité physique ou de l'aspect algorithmique. On distingue trois angles de vue, plus couramment désignés par les termes domaines d'expression pour exprimer une architecture (figure IV.1)

Ces domaines d'expression sont eux-mêmes découpés en cinq niveaux de description plus ou moins abstraits par rapport aux détails du circuit. Les cinq degrés d'abstraction du domaine comportemental par exemple sont : interrupteur, logique, transfert de registres,

algorithme et système. Les niveaux système et algorithmique sont quelquefois regroupés dans un seul niveau[5] Les trois domaines d'expression sont :

- Le domaine comportemental. Il décrit le comportement d'un circuit, autrement dit ce qu'il fait.
- Le domaine structurel. Il voit le circuit comme une interconnexion d'objets architecturaux de base et fait le lien entre le domaine comportemental et le domaine physique.
- Le domaine physique, souvent représenté par le dessin de masque du circuit. Il présente la structure de la réalisation physique de l'architecture.

La spécification de départ d'un circuit relève généralement du domaine comportemental. Diverses étapes de synthèse mènent ensuite à une description physique.

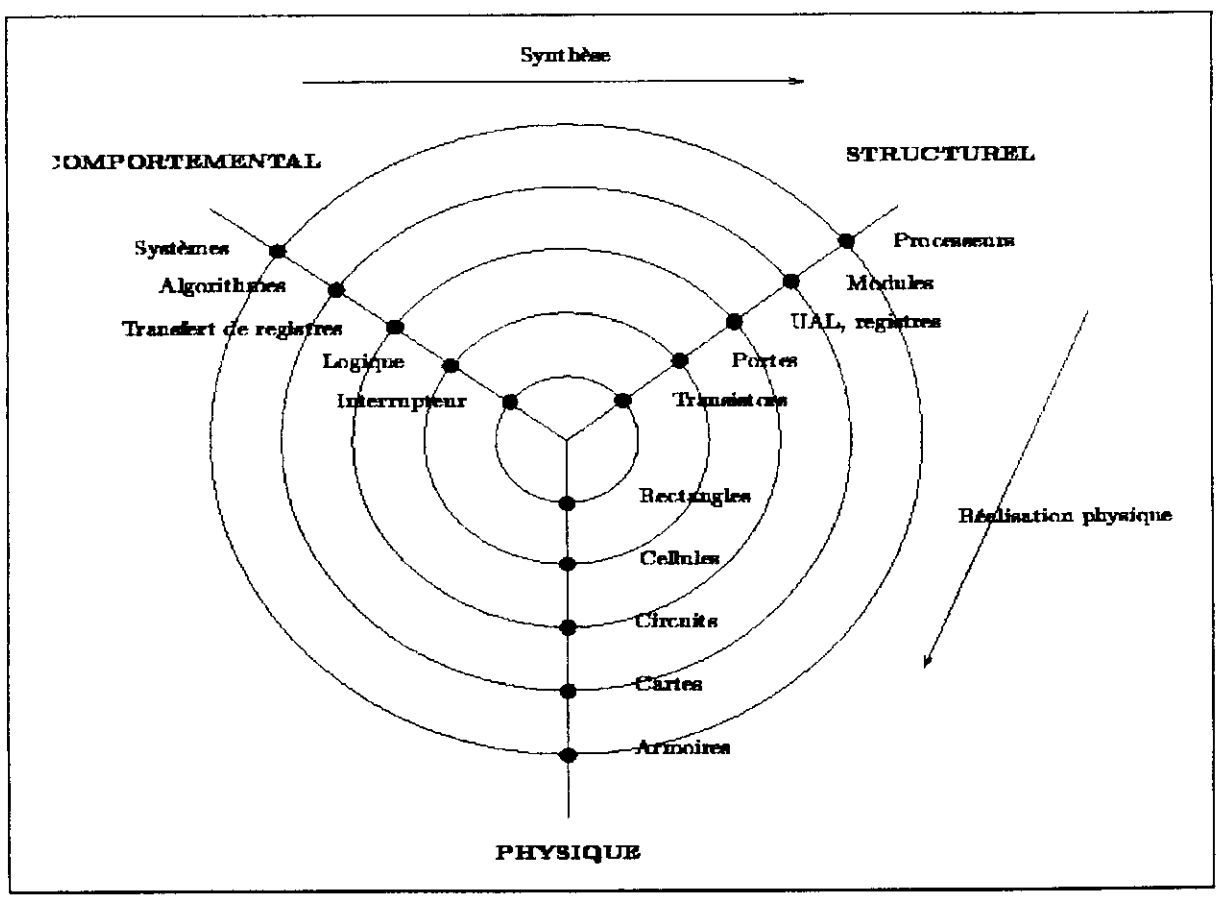


Figure IV.1 : Diagramme en Y introduit par Gajski décrivant les trois vues d'une architecture

3. ETAPES DE CONCEPTION

Le flot de conception standard comprend les trois étapes suivantes:
Spécification, synthèse, réalisation; après chaque étape une opération de test et vérification est exécutée, on peut alors donner le schéma de la figure IV.2 qui illustre le cycle de vie d'un système matériel [10]. On donne aussi le schéma de la figure IV.3 qui met en évidence le détail de chaque étape de conception [17].

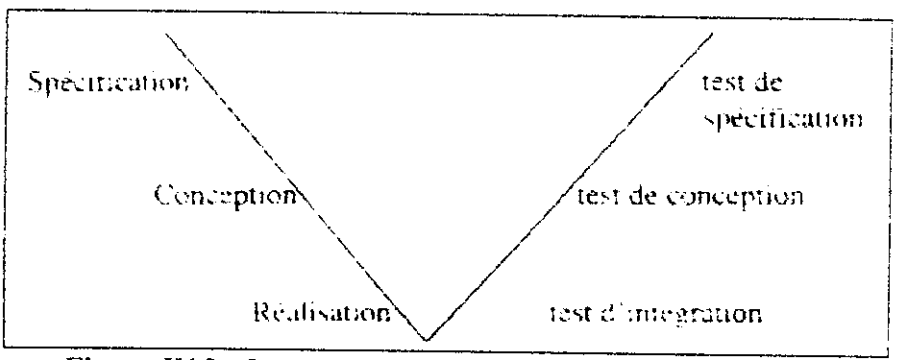


Figure IV.2 : Le cycle de conception d'un système matériel

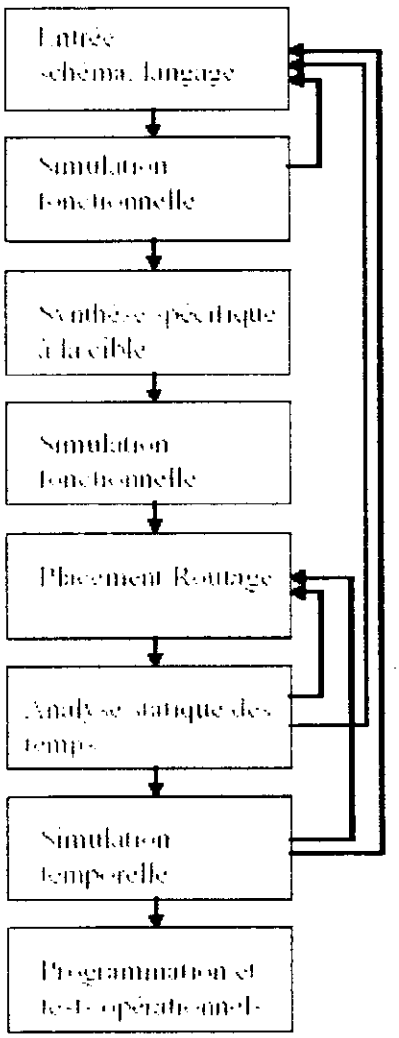


Figure IV.3 : Etapes de conception

3.1. SPECIFICATION DE SYSTEMES

Cette première étape consiste d'une part à situer notre système suivant les trois critères données ci-dessous [5] :

- Large choix de technologies. Le concepteur de systèmes doit faire face à une palette de choix de conception en constante augmentation (technologies de fabrication, méthode de conception, choix architecturaux).
- Partage matériel/logiciel. Les systèmes comportent une part de plus en plus significative de logiciel, en raison de la disponibilité de cœurs de processeurs (RISC, DSP, microcontrôleurs), comme éléments de bibliothèque. Il s'agit de pouvoir réaliser

une spécification indifféremment sous forme matérielle ou logicielle, de pouvoir déterminer le partitionnement et d'engendrer automatiquement des interfaces.

- Complexité des systèmes. La complexité croissante des systèmes est une conséquence directe de l'évolution des technologies cibles vers des niveaux d'intégration plus élevés.

La spécification d'un système permet d'autre part de description au niveau comportemental. Autrement dit le comportement de l'application est exprimé par des fonctions. Ces fonctions décrivent le calcul à réaliser et les relations entre entrées et sorties. Différents niveaux d'abstraction peuvent être répertoriés : interrupteur, logique, transfert de registres, algorithmique et système.

3.1.1. NIVEAU INTERRUPTEUR (CIRCUIT)

Une description au niveau interrupteur manipule des composants de très bas niveau caractérisé par des délais. Ces différents éléments sont regroupés pour former le circuit. Il n'est pas raisonnable de spécifier un système complexe à ce niveau de nos jours.

3.1.2. NIVEAU LOGIQUE

Une spécification logique est un ensemble de fonctions booléennes ou de machines d'états. Là encore son utilisation se limite à de petits circuits.

3.1.3. NIVEAU TRANSFERT DE REGISTRE

Le niveau supérieur est appelé transfert de registres ou RTL (*Register Transfer Level*). La description porte sur les états des registres et les transferts d'états. On distingue trois grandes parties : la partie chemin de données qui représente les calculs, la partie contrôle, et la partie mémoire. Plusieurs langages permettent de décrire un circuit au niveau RTL.

3.1.4. NIVEAU ALGORITHMIQUE

Le niveau algorithmique est le niveau le plus abstrait pour décrire un circuit. On peut spécifier le comportement d'une architecture à l'aide de graphes de dépendances (contrôle et/ou données). L'autre possibilité consiste à utiliser un des nombreux langages de description de matériel.

Les besoins spécifiques du matériel (contraintes de temps, de ressources, de performances) ont conduit à la définition de langages spécialisés pour la description d'architectures appelés HDL (*Hardware Description Language*). On peut classer les langages utilisés en deux styles : procédural et déclaratif. Il existe des passerelles pour aller de l'un à l'autre.

3.1.5. NIVEAU SYSTEME

Il spécifie un système de façon formelle. La traduction des fonctionnalités d'un système en une spécification fonctionnelle précise et sûre n'est pas aisée. Au niveau le plus abstrait, la description consiste en un ensemble d'entités fonctionnelles et les relations qui existent entre elles. Différents modèles permettent de représenter un système :

- le modèle flot de données, DFG (*Data Flow Graph*), qui se prête bien aux applications de traitement du signal ;
- les machines d'états finis, FSM (*Finite State Machine*), souvent améliorées par la hiérarchie et la concurrence, qui décrivent bien les systèmes dominés par le contrôle ;
- le modèle des processus communicants, CSP (*Communicating Sequential Processes*), qui représente des systèmes complexes et parallèles et est donc plutôt utilisé pour générer du logiciel ;
- le modèle PSM (*Program-State Machine*), un FSM qui contient des instructions de programme, et qui a donc les mêmes visées que les modèles CSP et FSM.

3.1.6. CHOIX DU MODE DE DESCRIPTION

Le choix d'un mode de description [4] s'inscrit dans la méthodologie de conception .des raisons aussi bien technique que stratégiques vont orienter le concepteur vers une méthode de conception adaptée figure IV.4. D'un point de vue technique, on tient compte de la nature de la fonction à réaliser et du composant qui va accueillir cette fonction. Les outils disponibles et les habitudes de travail du concepteur seront aussi un critère de choix du mode

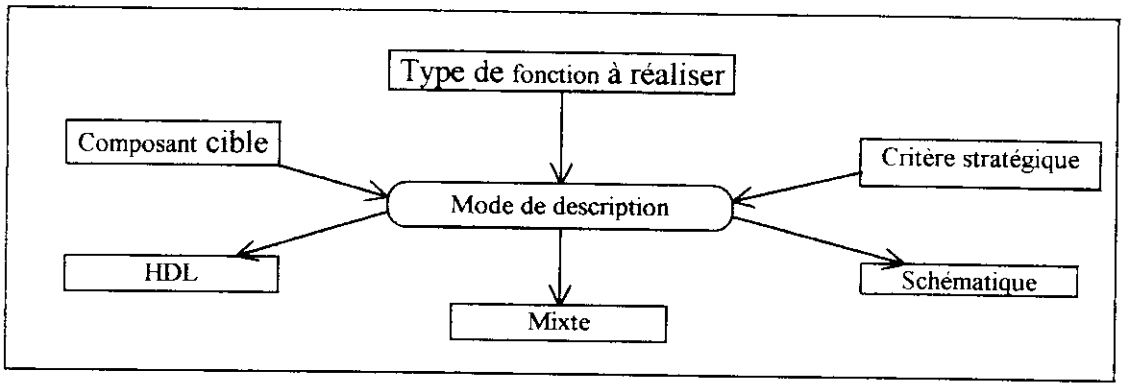


Figure IV.4 : Choix du mode de description

de description. Les critères stratégiques seront par exemple la 'portabilité'. Ce terme indique que le fichier source de la description n'est pas dédié à un composant particulier. On s'affranchit ainsi momentanément de la phase de choix du composant. La conception peut donc être testée sur plusieurs types de composants. Cette façon de procéder peut s'avérer stratégique lorsqu'on souhaite mettre à jour des conceptions antérieures dans des composants de technologie plus récente ou lorsque l'on souhaite reprendre des éléments d'une conception dans un projet différent.

Une entrée schématique utilise les composants de base (hard macro, composant optimisés) fournis pour un composant donné ou de moins une famille de composant. Le fichier source (schéma) est donc difficilement portable. Par contre la nature même d'un schéma (structurel) est bien adaptée pour imposer au synthétiseur une structure donnée. On utilisera ce type d'entrée lorsque la nature de la fonction à implanter est très structurée.

Une entrée HDL apporte de nombreux avantages. Un langage de type comportemental permet de rendre le code source indépendant de la cible. Alors qu'en schématique il existe autant de formats que d'outils, un code HDL normé (Verilog, VHDL, ABEL) sera au contraire accepté par la plus part des outils (simulateurs, synthétiseur).
Une entrée mixte consiste à définir certains éléments hiérarchiques et d'autre de manière syntaxique. Il pourra par exemple s'avérer judicieux de définir schématiquement la structure globale d'une conception et définir de manière comportementale chaque bloc fonctionnel

3.2. SYNTHESE D'ARCHITECTURES OU SYNTHESE COMPORTEMENTALE

Le flot de conception générales de circuit intégré (quelque soit l'architecture de la cible) est basée sur la notion de synthèse qui est le procédé automatisé permettant de faire passer la représentation d'une architecture d'un domaine d'expression à un autre.

En réalité le processus de conception [17] est la succession de trois différents type de synthèse dont la plus importante est "la synthèse architecturale appelée encore synthèse comportementale ".

3.2.1. SYNTHESE COMPORTEMENTALE

Les outils de synthèse d'architectures transforment une description comportementale (code) en une description au niveau transfert de registres (RTL pour Register Transfer Level) qui servira d'entrée pour l'outil de synthèse logique. Cette synthèse permet de déterminer l'assignation des fonctions du circuit à des opérateurs appelés aussi ressources, les interconnexions entre les différents opérateurs, ainsi que le moment d'exécution de chaque opération dans des intervalles de temps appelé *pas de contrôle*. La synthèse peut s'effectuer soit en considérant que la surface est bornée ou soit en considérant que c'est la vitesse qui est limitée.

3.2.2. SYNTHESE LOGIQUE

Elle détermine la vue structurelle d'un circuit au niveau logique. En synthèse logique, on manipule des spécifications logiques pour générer des interconnexions de primitives logiques. Elle détermine la structure microscopique du circuit, c'est-à-dire une interconnexion de portes. La dernière étape de la synthèse logique qui détermine quelles portes de la bibliothèque à utiliser est appelée "mapping technologique".

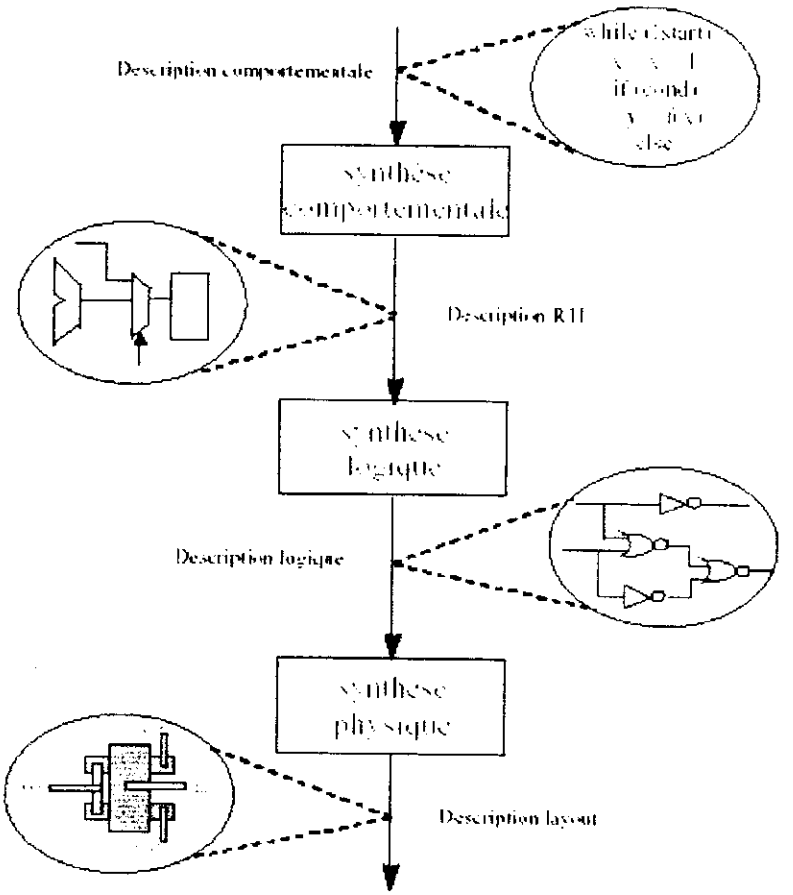


Figure IV.5 : Flot de conception général automatisé

3.2.3. SYNTHESE PHYSIQUE

Elle permet de générer les motifs géométriques des masques de fabrication des circuits à partir d'une description logique du système. La description du *layout* est la dernière étape de conception d'un circuit intégré, le fondeur utilisera ce *layout* lors de la fabrication du circuit intégré. Ce layout doit respecter des règles de dessin lesquelles sont typiques du procédé de fabrication.

3.2.3. SYNOPTIQUE DE LA SYNTHESE D'ARCHITECTURES

Le synoptique général de la synthèse d'architecture est résumé sur la figure suivante. Les différentes étapes de cette synthèse sont données comme suit :.

- **Analyse syntaxique et traduction**

La première étape du flot de synthèse comportementale, consiste à vérifier la syntaxe tolérée pour la synthèse, et de créer une représentation interne à l'outil de la description comportementale. Les outils de synthèse d'architecture actuels visent la conception de circuits digitaux synchrones.

- **Ordonnancement**

L'ordonnancement qui est une phase essentielles de la synthèse d'architecture définit une date d'exécution pour chaque opération de la description comportementale en tenant compte des dépendances fonctionnelles. Plusieurs méthodes d'ordonnancement existent pour résoudre les problèmes d'ordonnancement sous des contraintes matérielles ou temporelles.

- **Allocation**

L'allocation consiste à déterminer un ensemble de ressources matérielles nécessaires à l'implantation des opérations et des variables de façon à minimiser la surface totale du circuit. La surface est généralement exprimée par la somme des surfaces estimées pour chaque ressource (puisque le placement routage n'est pas encore effectué). La minimisation du nombre de ressources est basée sur les possibilités de partage (temporel) par les entités de la description comportementale.

- **Assignation (binding)**

L'assignation fait correspondre les opérations à des opérateurs (le plus souvent prédéfinis dans une bibliothèque : additionneurs, UALs,...), et les variables à des modules de mémorisation (registres). Le nombre maximal de variables vivantes à un instant déterminé donne une indication sur le nombre de registres nécessaires à l'implantation. Les opérateurs et les modules de mémorisation sont ensuite interconnectés à l'aide de bus, multiplexeurs, etc. de façon à pouvoir faire tourner la spécification initiale sur le chemin de donnée. Une fois le chemin de données déterminé, l'automate de contrôle est synthétisé.

- **optimisation de haut niveau**

Le rôle des optimisations présentées est de transformer la description comportementale en une description équivalente plus performante au niveau surface ou vitesse. Ces optimisations

peuvent s'effectuer soit au niveau de la traduction, ou soit au niveau du processus de la synthèse proprement dit, ce qui explique les deux flèches dans le synoptique de synthèse.

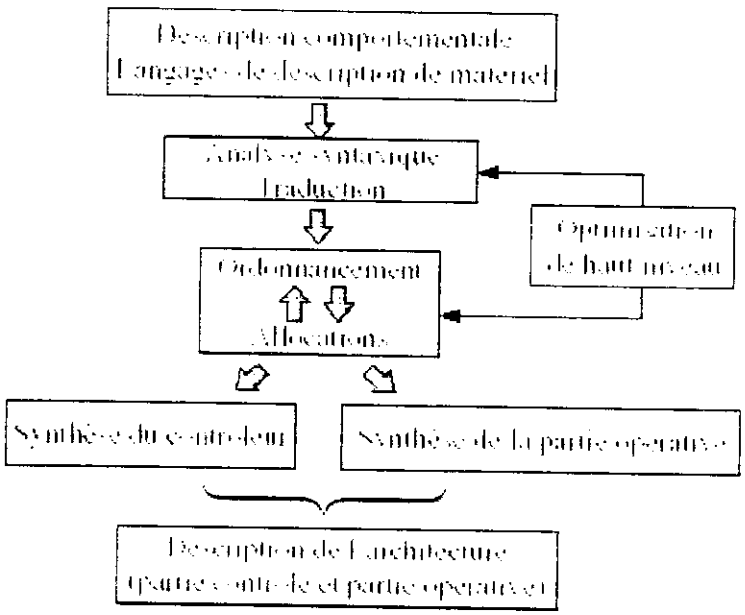


Figure IV.6 : Synoptique de la synthèse d'architectures

- **Les avantages de la synthèse d'architectures**

La synthèse d'architectures automatise et optimise la transformation d'une description comportementale en une description RTL. La spécification comportementale possède des instructions, des opérations, des variables, et des tableaux similaires et même identique à l'algorithme de départ. Il n'est plus nécessaire de décrire les registres de mémoires, les unités fonctionnelles telles que les additionneurs et les multiplieurs, les multiplexeurs, et même le contrôleur qui supervise les actions, la mémoire, et les entrées/sorties. Le concepteur n'a plus besoin de spécifier l'architecture exacte du circuit et peut automatiquement explorer plusieurs implémentations pour trouver l'architecture optimale.

Les principaux avantages de la synthèse d'architectures sont les suivants :

- une réalisation rapide de la spécification, car la description comportementale est plus courte (environ 10 fois plus courte qu'une description RTL) et plus intuitive (instructions similaires à l'algorithme de départ)
- une simulation plus rapide, car la description est plus abstraite qu'avec une description RTL.

- possibilité de créer et d'évaluer plusieurs implémentations rapidement, et donc de déterminer le meilleur compromis entre la vitesse et la surface du circuit.
- une meilleure qualité de résultats, car durant la synthèse, l'outil peut optimiser le compromis entre la surface et la latence en consultant la surface et les délais estimés de chaque composant qu'il utilise.
- réduction considérable du « time to market ».
- la possibilité pour des non experts de concevoir des circuits pour des applications particulières.

3.3. REALISATION

Cette dernière étape commence par le mapping et se termine quand la conception physique est routée avec succès et qu'un bitstream soit généré. Nous pouvons changer les contraintes durant l'implémentation comme nous faisions à l'étape d'entrée de conception.

La figure suivante montre une vue globale du processus d'implémentation [18] de la conception pour les FPGAs et les fichiers générés à chaque niveau.

3.3.1. MAPPING

C'est le processus d'assignement aux éléments de la logique d'une conception des éléments physiques spécifiques qui mettent en application réellement des fonctions logiques dans un dispositif.

Pour les FPGAs, le programme MAP fait le mapping pour un FPGA Xilinx. L'entrée au programme MAP est un fichier NGD, qui contient une description logique de la conception en termes de composants hiérarchiques utilisés pour développer la conception et de primitives du plus bas niveau de Xilinx, et de tout nombre de fichiers NMC (hard placed-and-routed macro), dont chacun contient la définition d'un macro physique. Le programme MAP fait alors le mapping de la conception logique sur les composants (cellules logique, cellules d'E/S, et d'autres composants) dans la cible FPGA Xilinx. Le résultat est un fichier (NCD:Native Circuit Description), qui est une représentation physique de la conception mappé sur les composants dans l'FPGA Xilinx. Le fichier NCD peut alors être placé et routé.

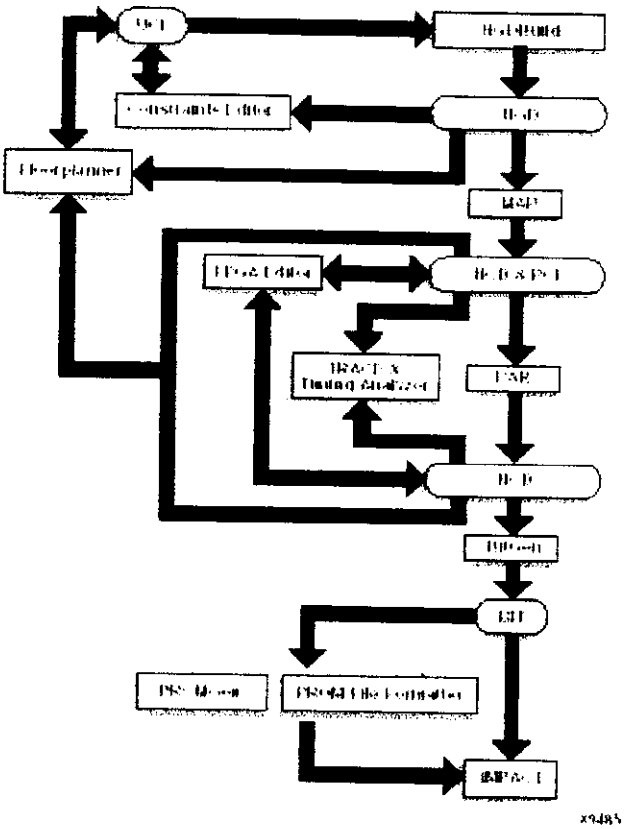


Figure IV.7: Processus d'implémentation

3.3.2. PLACEMENT ET ROUTAGE

Le placement est le processus d'assigner des endroits physiques de cellules de dispositif au logique dans une conception. Le cheminement est le processus d'assigner les filets logiques au fil physique segments dans les FPGA qui relient ensemble des cellules de logique.

Pour les FPGAs, le programme "PAR" prend un fichier "NCD" comme entrée, fait le placement et le routage de la conception, et donne comme résultat un fichier "NCD", qui sera utilisé par le générateur de bitstream, "BitGen". Le fichier "NCD" résultant sert comme un fichier de guide quand le processus de placement et routage est lancé de nouveau pour une conception dont peu de changements ont été faits après l'itération précédente.

3.3.3. GENERATION DE BIRSTREAM

Pour les FPGAs, le programme "BitGen" produit un bitstream pour la configuration . "BitGen" prend un fichier "NCD" entièrement routé en tant que son entrée et produit un fichier binaire avec une extension" .bit"). Le fichier "BIT" contient toute l'information de la

configuration du fichier "NCD" définissant la logique interne et interconnections du FPGA, plus les informations spécifiques du dispositif contenues dans les autres fichiers associés au dispositif cible.

3.4. VERIFICATION DE LA CONCEPTION

La vérification de la conception est le processus de tester la fonctionnalité et la performance de la conception. Nous pouvons vérifier les conceptions Xilinx de différentes manières dans différent niveau de conception :

1. Simulation (fonctionnelle et temporelle)
2. Analyse temporelle statique (Static timing analysis).
3. Vérification en circuit (In-circuit verification).

Les procédures de vérification de conception devraient se produire dans tout le processus de conception [18], comme représenté sur la figure suivante : .

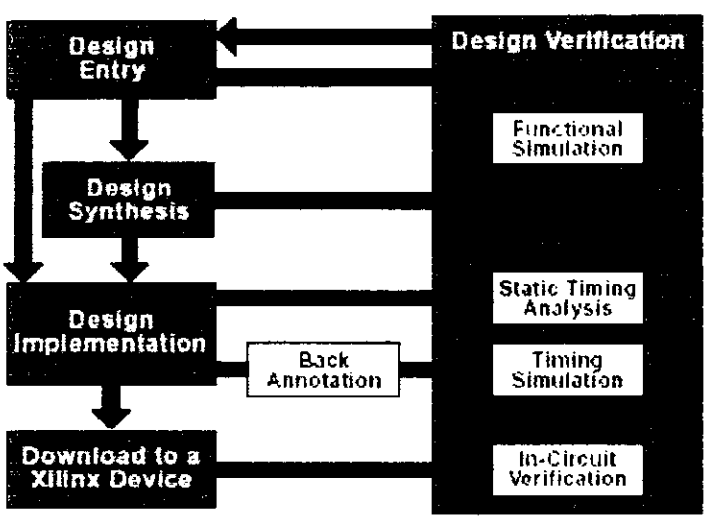


Figure IV.8:processus de vérification

3.4.1. SIMULATION

Cette section décrit le procédé de back-annotation qui doit se faire avant la simulation temporelle. Elle décrit également les méthodes de simulation fonctionnelle et temporelles pour les deux types d'entées de conception schématiques et HDL.

- **Simulation fonctionnelle**

Chaque conception est testée dans son niveau fonctionnelle [4]. Cette simulation fonctionnelle permet de tester la validité de notre conception. Aucune vérification temporelle

n'est ici effectuée. Cette première phase de simulation nous a apporté un double intérêt. Elle nous a permis de valider fonctionnellement la description (entrée schématique, syntaxique ou en machine d'états) en s'affranchissant des étapes parfois longues de placement et de routage. Elle nous a permis aussi de valider (fonctionnellement) des sous – ensembles de la conception au fur et à mesure de l'avancement des travaux. Une seconde phase de simulation fonctionnelle a été réalisée après la phase de synthèse. Cette deuxième phase permet essentiellement de valider le travail du synthétiseur et de mettre en évidence des problèmes éventuels.

- **Simulation temporelle**

La simulation temporelle [18] vérifie que la conception fonctionne à la vitesse désirée pour le dispositif. Ce processus est exécuté après le mapping, le placement et le routage de la conception. A ce stade, tous les retards dans la conception sont connus. La simulation temporelle est précédée du processus back annotation pour pouvoir créer des formats appropriés pour différents simulateurs.

3.4.2. ANALYSE TEMPORELLE STATIQUE

L'analyse temporelle statique [4] est la meilleure méthode pour les contrôles rapides des caractéristiques temporelles d'une conception après qu'elle soit placée et routée. Elle nous permet d'analyser les retards d'un point à un autre dans l'architecture du réseau de conception pour un ensemble de contraintes données sans tenir comptes des vecteurs de test.

3.4.3. VERIFICATION IN-CIRCUIT

Comme essai final, nous pouvons vérifier comment la conception se comporte dans l'application cible. La vérification in-circuit examine le circuit sous des conditions typique de fonctionnement [18].

4. LES OUTILS DE CONCEPTION [4]

La diversité des outils disponibles sur le marché ne nous permet pas de présenter ici toutes les solutions existantes. D'une manière générale on distinguera les outils *génériques* et des outils *spécifiques*. Les outils spécifiques sont généralement fournis par les constructeurs de composants. Les outils génériques sont en général indépendants de la cible. La figure

illustre ces deux types d'outils en les associant aux différentes étapes du flot de conception présentées précédemment et les différentes bibliothèques utilisées à chaque niveau.

La figure suivante présente un exemple fictif qui permettra d'appréhender les principes généraux. On distinguera plusieurs types d'entrées. L'entrée schématique consiste à réaliser un schéma à partir de composants issus d'une bibliothèque spécifique à la cible. Une saisie de schéma n'est en faite qu'un outils interactif permettant d'interconnecter des éléments. Ces éléments sont définis par le concepteur et peuvent être utilisés dans plus niveaux de hiérarchie. Quoi qu'il en soit, on trouvera au niveau le plus bas de la hiérarchie des éléments de base issus d'une bibliothèque fournie par le constructeur du composant. Cette bibliothèque doit être compatible avec le format de l'outil de saisie. Le simulateur peut quant à lui être tout à fait générique. D'une manière générale il est capable, à partir de modèle de base, de simuler toute netlist (le simulateur peut aussi être capable de simuler une description comportementale). Un fichier 'NETLIST' est généré à partir du schéma et du résultat de la synthèse d'un code comportementale. Il existe de nombreux format de netlist (VHDL, XNF, EDIF). Ces formats, devenu standard sont souvent issus d'outils propriétaires. Des passerelles permettent l'interface entre les différents niveaux de la compilation. Les outils de synthèse, de placement et de routage tiennent comptes de ces contraintes pour orienter leurs choix.

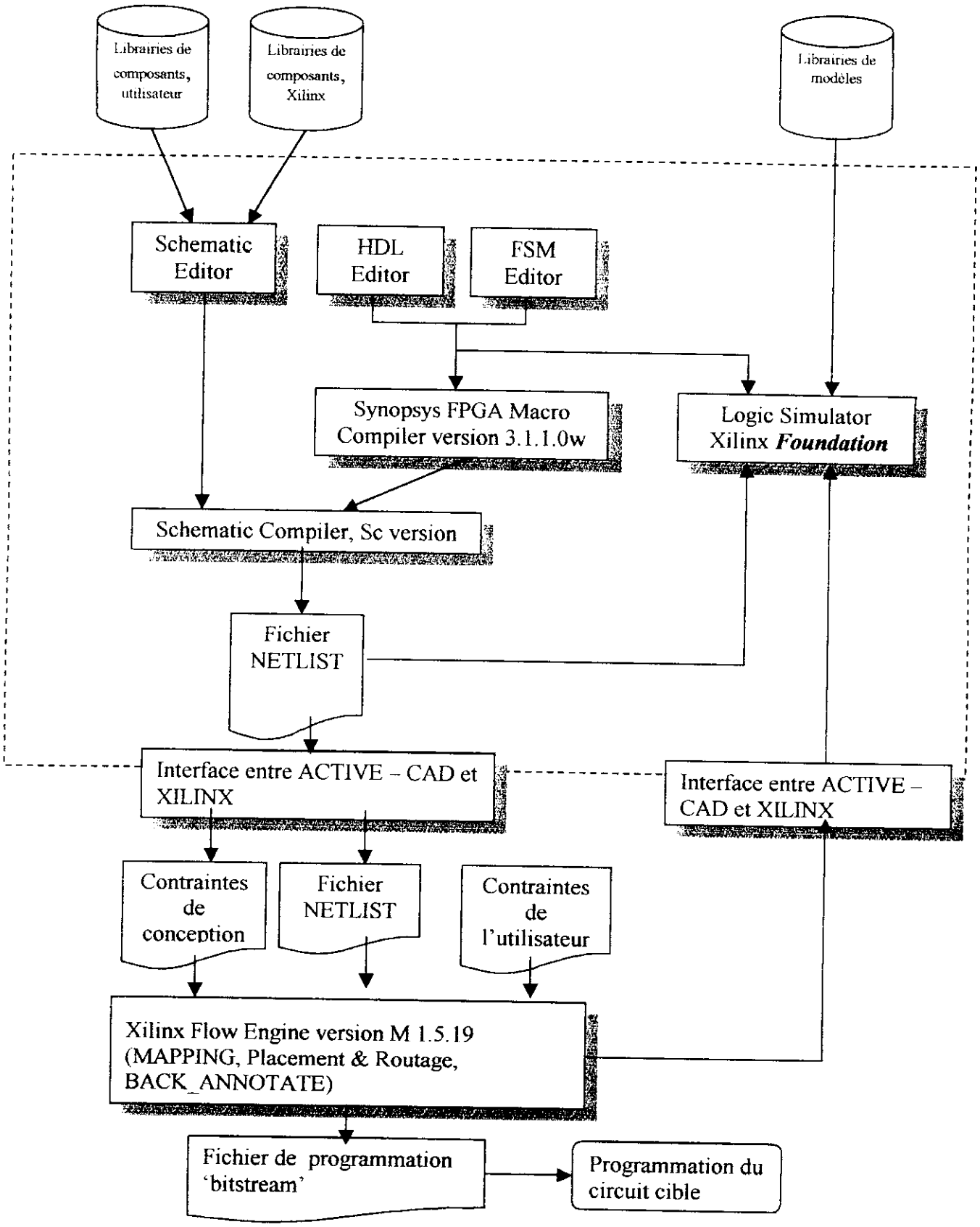


Figure IV.9: Outils de conception

1. INTRODUCTION

Comme application à la mise en œuvre de la carte XS40, on s'intéressera ci-après à l'implémentation d'un algorithme de codage ; le choix de l'algorithme restant toutefois arbitraire. L'objectif visé est de proposer une approche méthodique pouvant être appliquée à n'importe quel autre algorithme.

On présentera d'abord l'architecture globale de l'implémentation cible tenant compte d'un cahier de charges prédéfinie:

- implémentation d'un algorithme de codage;
- pouvoir intégrer la conception dans une structure maître/ esclave;
- accélération de l'exécution du traitement proposé.

2. DESCRIPTION GLOBALE DE L'ARCHITECTURE

Avant d'entamer cette description, on vous propose la structure qu'on a adopté pour concevoir un circuit associé à l'algorithme de codage choisi; le codage de HAMMING.

2.1. CONCEPTION ALGORITHMIQUE DES CIRCUITS VLSI

Cette technique est issue de la conception des ordinateurs qui permet le passage d'un algorithme à un circuit. Ce type de conception est adapté à la conception des circuits logiques complexes et rapides ou produits en très grande série comme elle peut s'appliquer à la conception des modules pouvant fonctionner en parallèle. Les systèmes qu'on obtient alors ressemblent à des microprocesseurs dits dédiés [15].

Le processus de conception consiste tout d'abord en l'adaptation des algorithmes en un ensemble d'opérations réalisables sur circuit; pour que ce dernier puisse reproduire identiquement la séquence de l'algorithme. Le circuit qu'on va concevoir alors peut être décomposé en deux blocs:

2.1.1. CHEMIN DE DONNEES

Cette partie du circuit est constituée d'un ensemble d'opérateurs, de registres et de bus, permettant ainsi la réalisation d'opérations arithmétiques, des tests et des indexations.

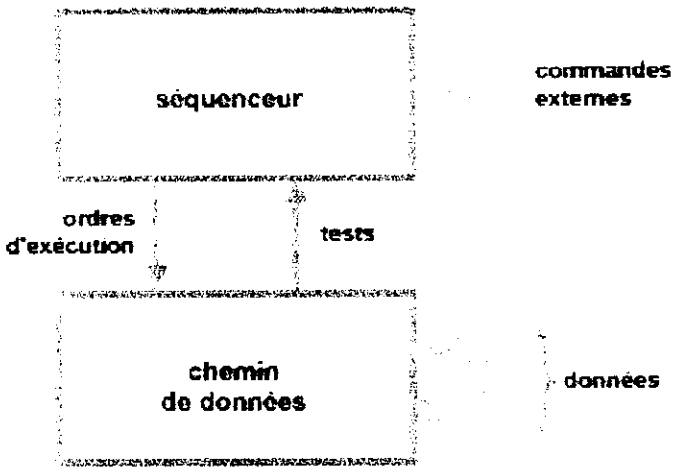


Figure IV.1: Décomposition d'un circuit

2.1.2. SEQUENCEUR

Le séquenceur est une machine d'états avec des entrées condition issues des résultats de test du chemin données. Il permet alors la génération de sorties de commandes pour arbitrer le chemin de données. La structure du séquenceur peut être câblée, microprogrammée ou mono-PLA.(L'architecture PLA est bien détaillée dans le chapitre I § 4).

Les figures ci-après illustrent les différentes architectures du séquenceur et la communication avec le bloc chemin de données.

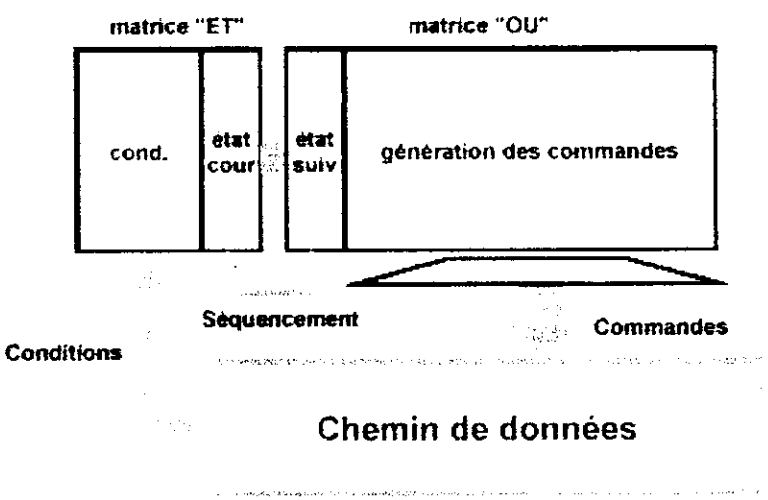


Figure IV.2: Architecture du séquenceur Mono-PLA

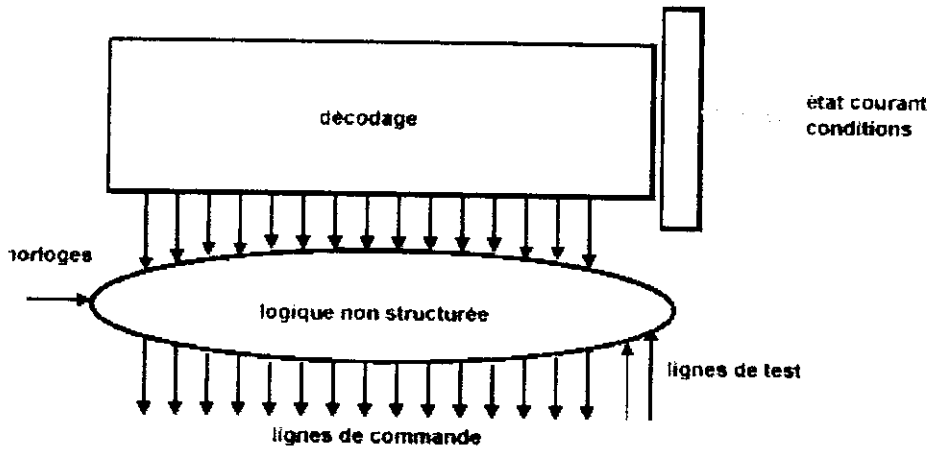


Figure IV.3: Architecture du séquenceur câblé

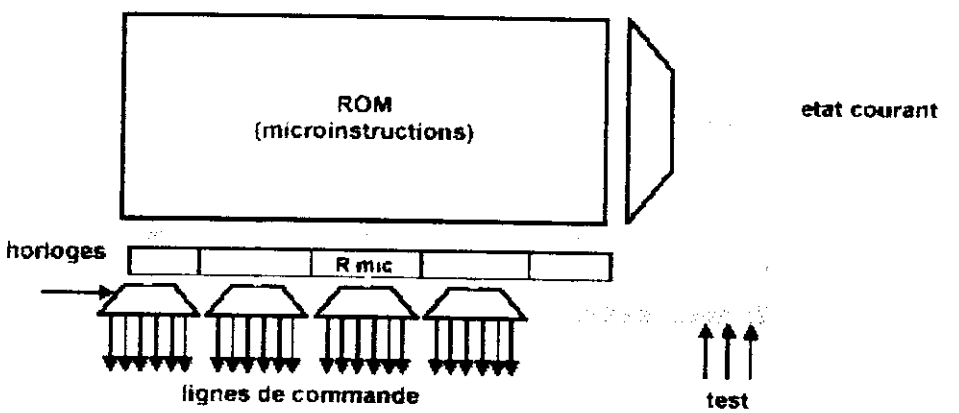


Figure IV.4: Architecture du séquenceur microprogrammé

2.2. DESCRIPTION DU SYSTEME

L'étape suivante consiste à décrire la structure générale associée au codeur (décodeur) pouvant être intégrée dans un environnement ou avec un organe maître qui peut lui envoyer des données à coder (à décoder) ou recevoir les données codées (décodées) . L'architecture de notre conception repose sur le principe de conception algorithmique déjà décrite avec un séquenceur câblé.

Pour que l'intégration de notre système dans une structure maître/esclave soit possible, il faut qu'on envisage une unité de réception des données à traiter (à coder ou à décoder) et une unité d'émission (données codées ou décodées) autrement dit un système d'échange d'information.

On doit aussi ajouter un registre qui permet à notre système de communiquer à l'extérieur son état pour que ce dernier puisse contrôler ses données et les résultats de traitements.

Pour que l'ensemble du chemin de données (codeur/décodeur, unité d'émission, unité de réception, registre d'état) et l'organe extérieur peuvent exécuter leurs tâches correctement on doit faire appel à un séquenceur qui va contrôler et commander le système en imposant des signaux de commandes sur le chemin de données. Ce séquenceur permet donc d'arbitrer notre circuit d'une part et la communication de ce dernier avec l'extérieur.

La figureIV.5 illustre la structure de l'architecture qu'on vient de décrire.

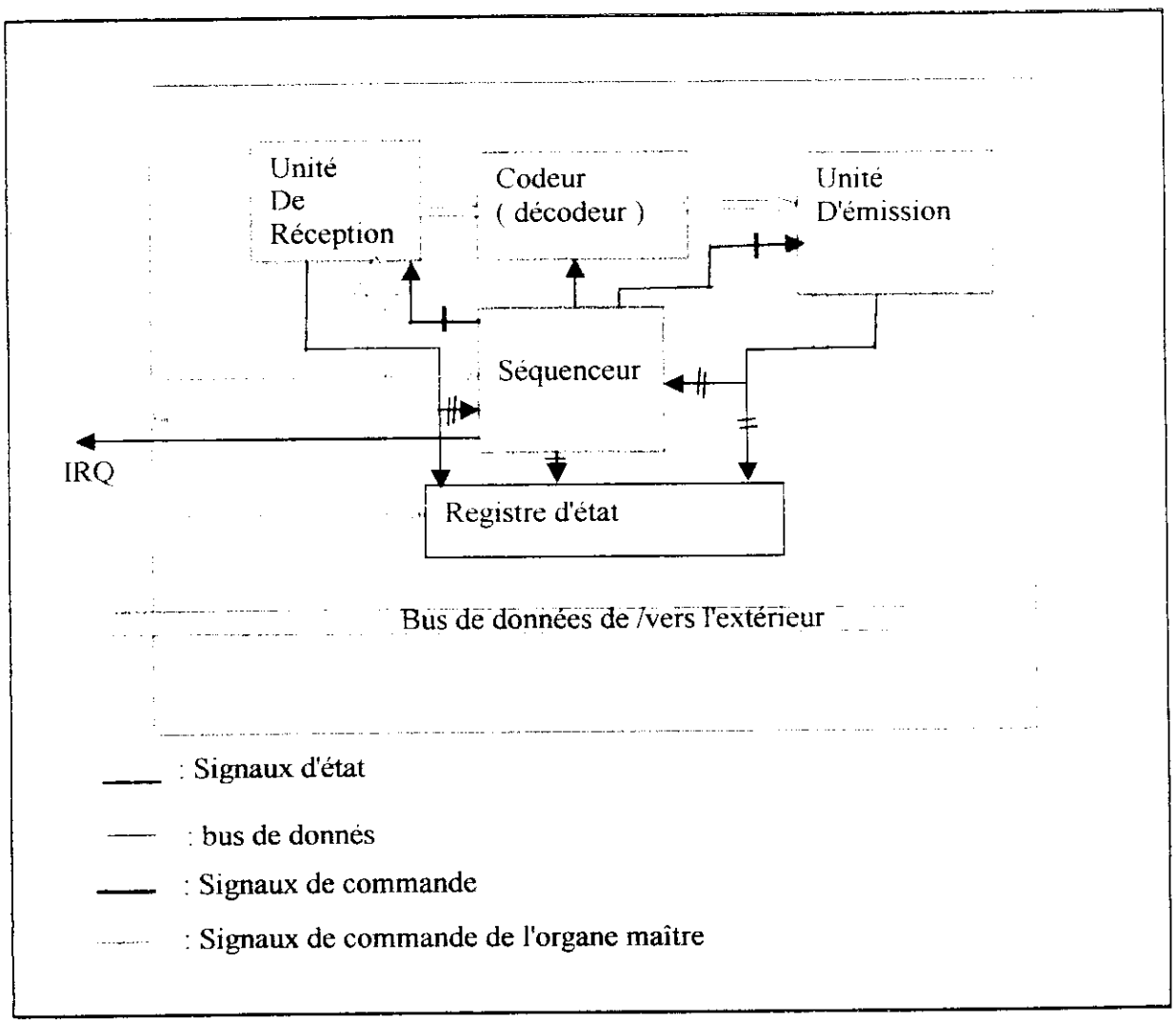


Figure IV.5 : Description structurelle du système codeur (décodeur) de Hamming

On va par le suite décrire chaque bloc de la structure ainsi défini séparément :

3. BLOCS PRINCIPAUX DU CIRCUIT

3.1. COMMUNICATION AVEC L'EXTERIEUR

Le système qu'on vient de concevoir doit communiquer avec l'extérieur, soit pour récupérer des données à traiter ou pour transmettre ses résultats de traitement, pour ce faire on envisage des signaux qui permettent cette communication.

- une entrée d'horloge CLK est utilisée pour synchroniser le fonctionnement de notre système
- R/W : cette ligne de commande définit l'ordre du système maître , soit la transmission d'une donnée à traiter ou d'une demande de récupération du résultat de traitement.
- B.D (16 bits): un bus de données bidirectionnel. Utilisé en entrée, ce bus permet de véhiculer les données à traiter .en sortie, il transmet soit le résultat de traitement ou le contenu du registre d'état.
- B.A: permettra la sélection de système d'une part et le type de données sur le B.D (données à traiter ou déjà traitées ou encore le contenu du registre d'état).
- IRQ: ligne de demande d'interruption.

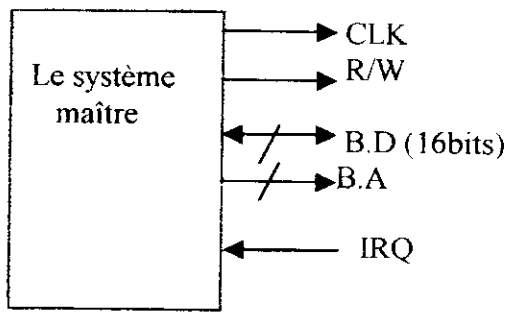


Figure IV.6: Communication avec l'extérieur

3.2. CHEMIN DE DONNEES

Le chemin de données qu'on a associé à cette conception est composé de trois blocs fonctionnels de base à savoir :

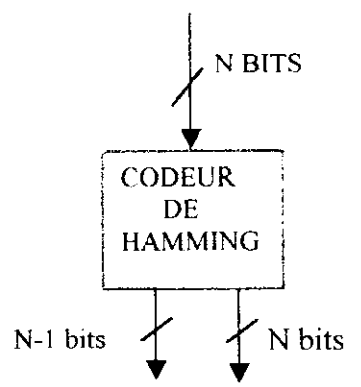
- Unité d'émission;
- Unité de réception ;
- Unité de traitement (codeur ou décodeur).

3.2.1. UNITE DE TRAITEMENT

L'unité de traitement conçue permet le codage canal de type HAMMING (une étude de l'algorithme associé est donnée dans l'annexe B).

- **Modèle comportemental de l'algorithme de Hamming**

L'algorithme de codage (ou de décodage) de Hamming, consiste en une fonction qui permet de générer à partir de N bits (ou 2N-1bits) en entrée ,2N-1 bits (ou N bits) en sortie, dont les N-1 bits supplémentaires dans le cas du codage permettent la correction et la détection d'une seule erreur comme le présente la figure ci-après :



FigureIV.7: Les entrées et les sorties du codeur

Le schéma du décodeur ne peut être que associé à la fonction inverse du codeur, il se présente alors comme dans le schéma de la figure IV.8.

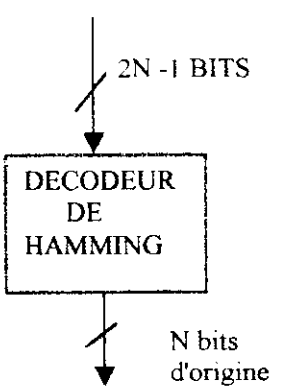
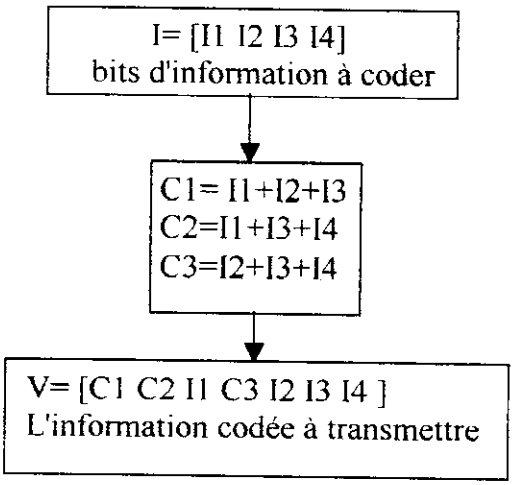


Figure IV.8 : Les entrées et les sortie du décodeur

Les figures IV.9 et IV.10 présentent respectivement les digrammes de codage et de décodage suivant l'algorithme de Hamming.

Dans le cas de notre implémentation on propose de coder des mots d'information de 8bits mais avec un traitement en deux blocs de 4 bits ce qui va donner en sortie du codeur 14 bits (7bits pour chaque bloc, 7=4+ (4-1)). Le même raisonnement a été adopté dans le décodage.



FigureIV.9 : Digramme de codage (émission)

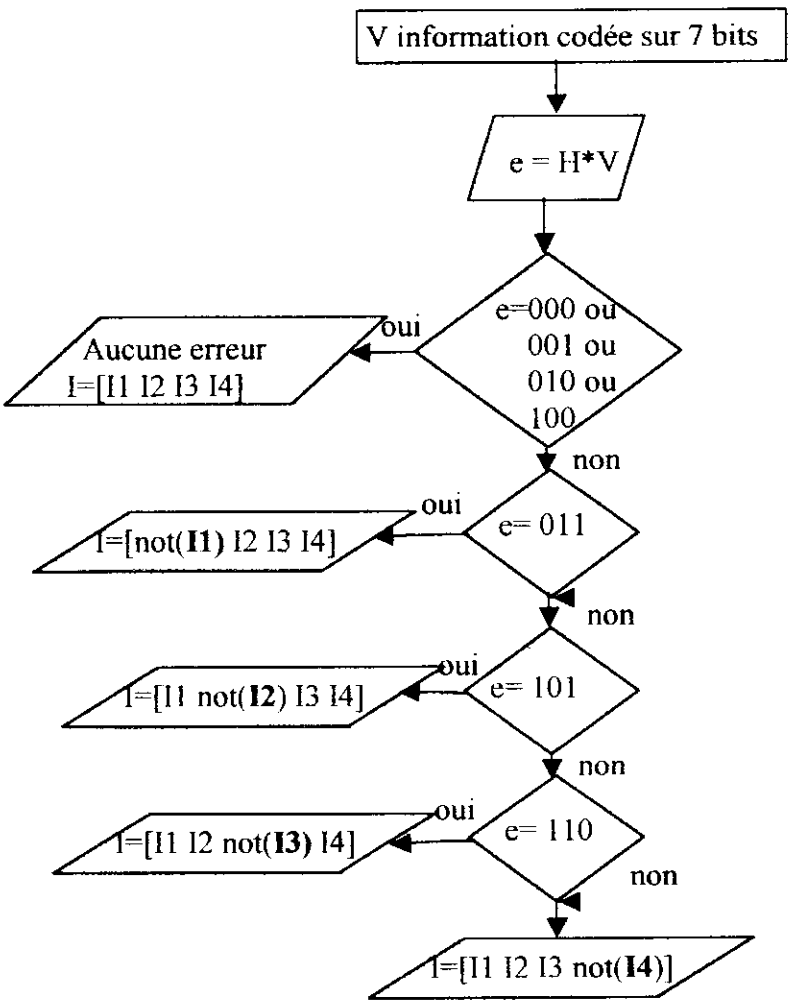
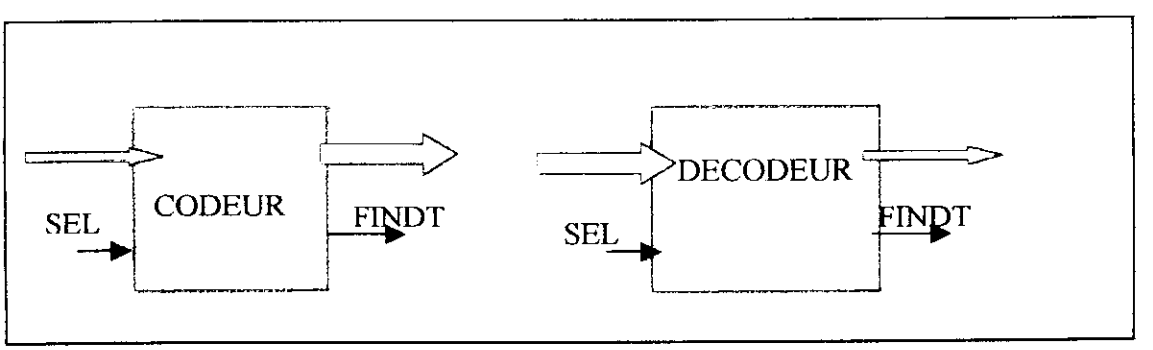


Figure IV.10 : Diagramme de décodage (réception)

L'unité de traitement dans le cas de notre conception peut être soit un codeur ou un décodeur suivant la nature des données à traiter. La description de ces blocs a été faite en traduisant les organigrammes donnés par les figures IV.9 et IV.10 en code VHDL.



FigureIV.11 : Unité de traitement

L'unité de traitement permet d'associer à une donnée (à coder ou à décoder) en entrée, une donnée (codée ou décodée) en sortie si cette unité est sélectionnée (SEL: signal de sélection), on attribue à ce bloc aussi un signal de fin de traitement (FINDT) qui informe l'extérieur s'il est entrain de traiter ou il peut recevoir une nouvelle information à traiter.

3.2.2. UNITE D'EMISSION ET DE RECEPTION

Ces unités permettent la récupération des données pour les traiter, stocker ou véhiculer les résultats de traitement, du/au système extérieur appelant (mémoire, microprocesseur).

Ces unités peuvent être de simples tampons d'entrée sortie; mais dans le but d'augmenter la bande passante de ce système en rapprochant le chargement de données, on a fait appel à des mémoires RAM internes contrôlées par des piles FIFO "First in First out buffers".

Donc le bloc de base des unités d'émission et de réception est la pile FIFO et les mémoires associées.

- **First in first out buffers**

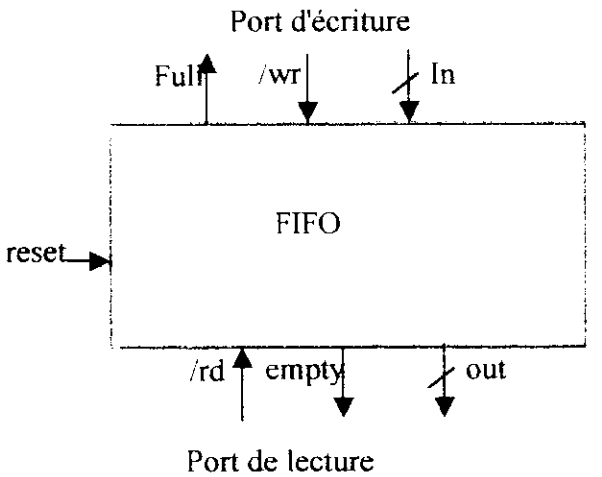
Dans les systèmes de communication, l'émetteur peut envoyer les informations avec une vitesse beaucoup plus importante que celle du récepteur qui les communique par exemple à un organe de traitement relativement lent par rapport à l'émetteur.

Donc on peut dire qu'il est fortement recommandé d'utiliser des tampons (buffers) qui peuvent recevoir les données transmises avec une grande vitesse, et permettent la lecture de ces données à chaque moment. Cette façon d'échange de données doit obligatoirement mémoriser l'ordre de réception des données. Le composant qui offre toutes ces opération est appelé " **First_in, First_out buffers** " **FIFO.** Ce composant permet de rendre la contrainte de différence de vitesse un critère d'optimisation de l'architecture de notre système.

- **Réalisation d'une pile FIFO**

Une pile FIFO est constitué d'une mémoire RAM (random access memory) dans laquelle on peut lire et écrire; avec deux différents pointeurs, un pour la lecture "rdptr" et un autre pour l'écriture "wrptr"; ils permettent de pointer sur la prochaine case mémoire respectivement à lire ou à écrire. La pile FIFO présente aussi deux indicateurs "full" et "empty" pour indiquer l'état de chargement de la mémoire. Elle pose full à l'état "1" si toutes les cases mémoire sont chargées alors qu'elle présente un niveau haut sur empty si toutes les

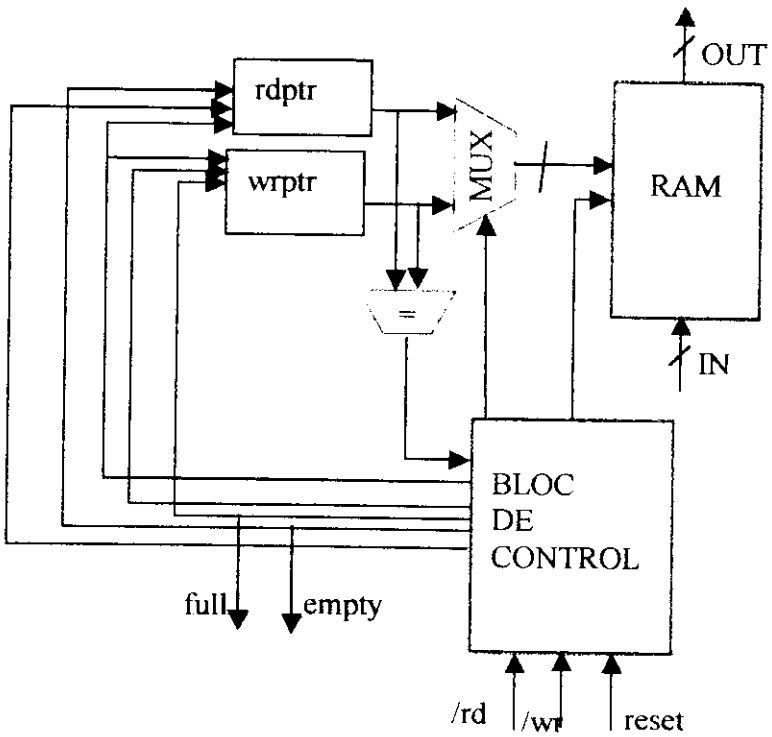
cases mémoires sont lues .La pile FIFO présente aussi un port de lecture et un autre port pour l'écriture [3].



FigureIV.12 : Les signaux d'interface d'une pile FIFO.

Les composants donc dont on a besoin pour réaliser un FIFO sont :

- Une RAM;
- Deux pointeurs d'adresses qui prennent leurs valeurs de la séquences '0 1 2 ....N 0 1...'; l'incrémentation de ces adresses ne peut se faire sauf si empty=0 pour rdptr et full=0 pour wrptr. Le circuit qu'on associe à ces pointeurs doit permettre son incrémentation, sa mise à zéros ou stopper son incrémentation.
- Un multiplexeur pour décider du type d'adresses (wrptr ou rdptr) à présenter sur le bus d'adresse de la mémoire selon l'opération qui aura lieu.
- Un comparateur qui teste l'égalité des pointeurs pour pouvoir indiquer l'état full ou empty du FIFO.



FigureIV.13 : Les composants d'une pile FIFO.

- Bloc de controle : cette unité permet le controle du multiplexeur, les lignes de lecture/écriture ainsi que l'incrémentation des adresses.

Ce bloc permet aussi de déterminer le nouvel état de full et empty. Ces deux indicateurs sont mis à zéro c.a.d la mémoire n'est pas pleine mais elle non plus vide dans le cas ou les deux pointeurs (wrptr et rdptr) sont différents.

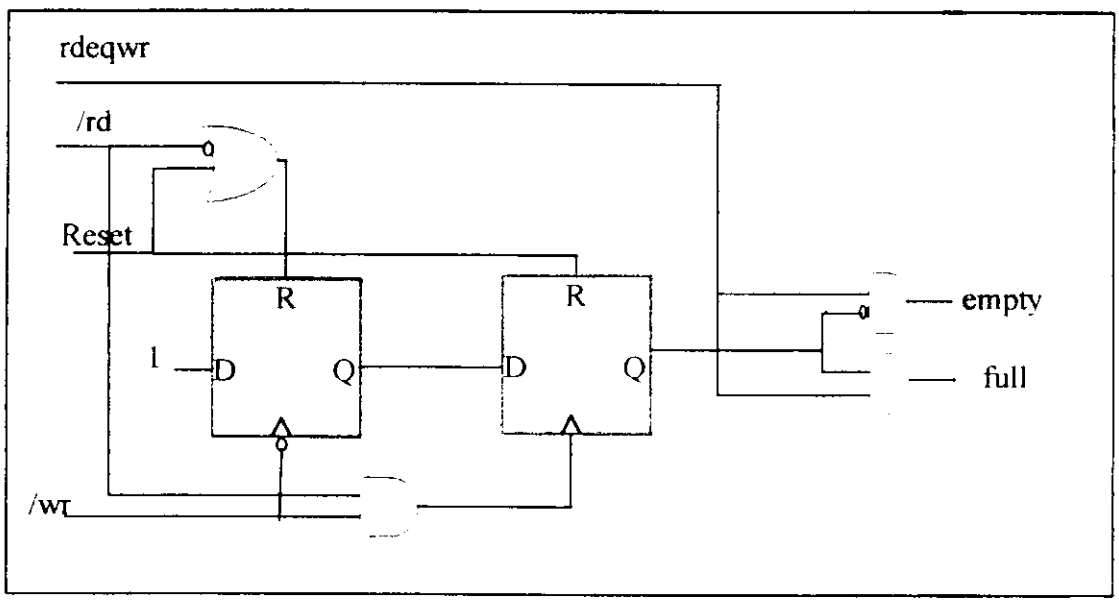
La question est comment déterminer l'état du FIFO quand les deux pointeurs sont égaux (indiqué par un état haut sur rdeqwr). La réponse à cette question est donnée par l'utilisation de l'état précédent, pour indiquer la dernière opération effectuée. Si cette opération est une écriture, le FIFO ne peut pas être vide (empty=1), puisqu'on vient d'écrire une donnée. Inversement si l'opération est une lecture. Le FIFO ne peut pas être plein (full=1) ; on vient juste de lire la mémoire.

On peut conclure donc que :

- si les deux pointeurs de lecture, écriture sont égaux, et la dernière opération est une écriture ⇒ `full=1;`

- si les deux pointeurs de lecture, écriture sont égal, et la dernière opération est une lecture ⇒ `empty=1;`

La logique de commande de la pile FIFO se base sur l'état précédent, ceci peut faire appel à une structure qui permet la mémorisation de l'état du FIFO, la figure IV.14 propose une solution qui utilise une bascule D.



FigureIV.14: Bloc de control

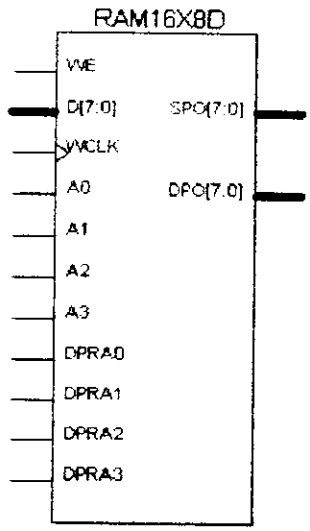
Supposant que les opérations de lecture et écriture sont activées au niveau bas.
La première bascule qui présente prevop0 comme sortie indique au système le type d'opération actuelle; la sortie prevop0 est mise à l'état haut si une opération d'écriture s'effectue et un niveau bas sinon (lecture), pour ce faire on relie le signal d'écriture (/wr) à l'horloge de la bascule dont l'entrée est mise à "1" de façon permanente; et on associe le signal de lecture (/rd) à l'entrée de mise à zéro asynchrone "reset".

Puisque on a besoin de l'état précédent, on ajoute une autre bascule qui permet de retarder la transmission de l'état prevop0 à la sortie prevop1 après que /wr et /rd reviennent à leurs états inactifs (niveau haut).

Mémoire RAM

Les mémoires qu'on a utilisé pour réaliser les FIFOs sont des mémoires de type RAM double port qui présentent un port S de lecture /écriture avec des lignes d'adresses Ai, un

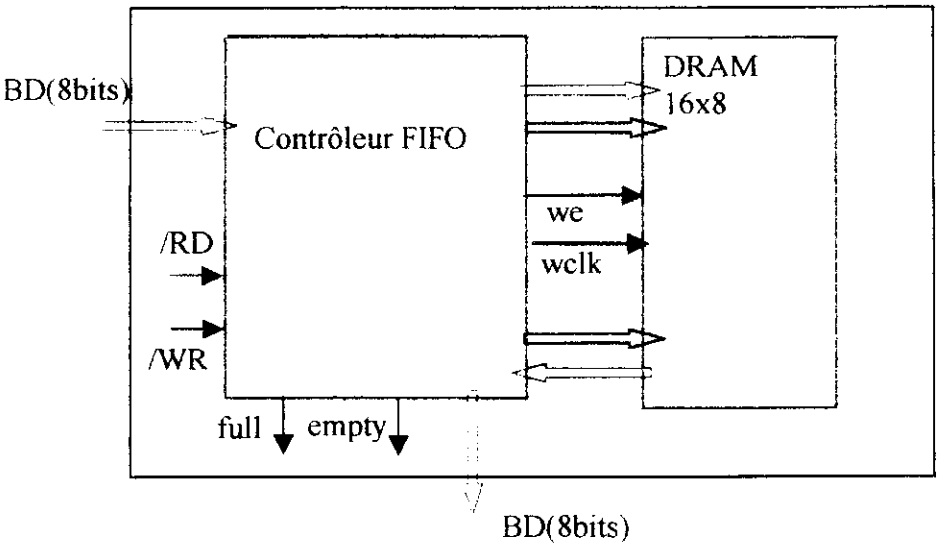
signal de lecture/écriture WE et une entrée horloge WCLK; et un autre port D pour la lecture avec des lignes d'adresses DPRAi.



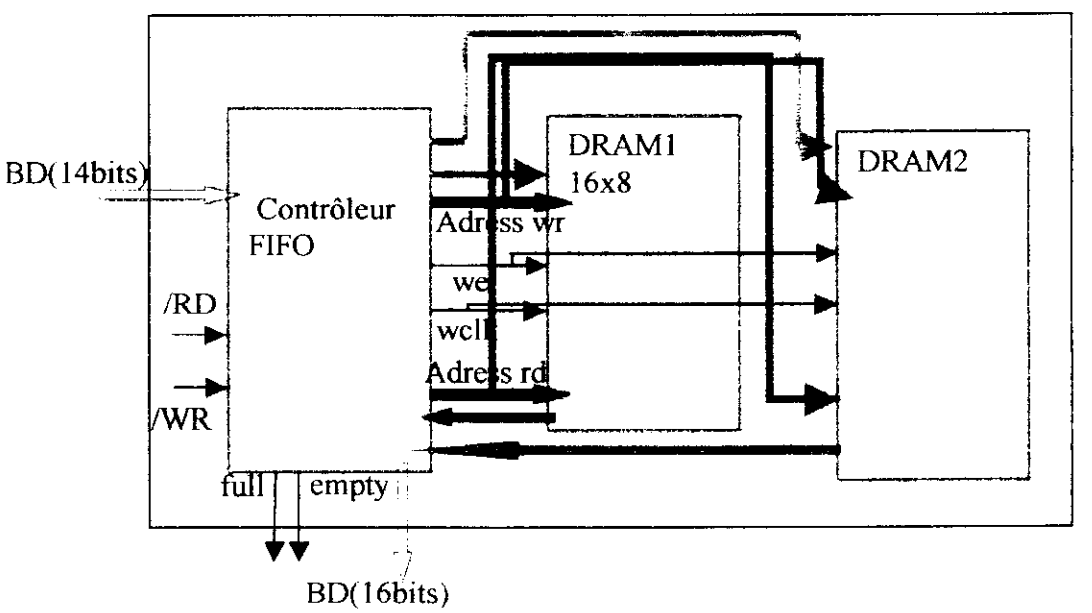
FigureIV.15: Mémoire RAM double port

La structure que nous avons donnée aux unités d'émission et de réception peut être associée à n'importe quelle bloc de traitement ; dans le but de rapprocher les données et de pouvoir paralléliser l'exécution de plusieurs traitements dans un système maître/esclave.

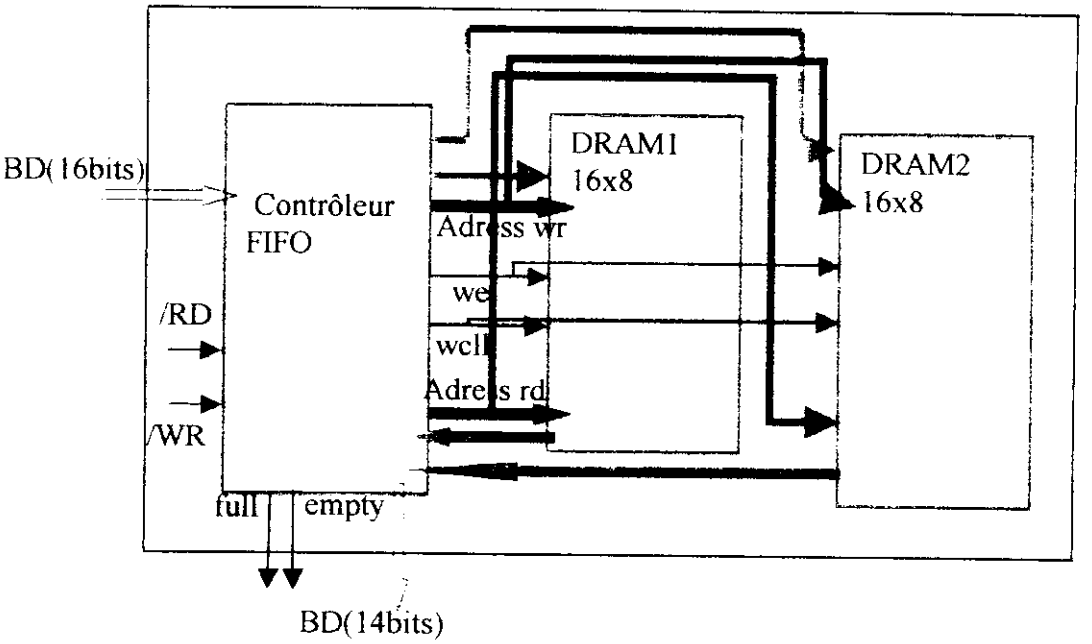
Les différentes structures FIFO qu'on a utilisées (elles sont au nombre de 3) dans notre système, diffèrent entre elles par la taille des bus de données d'entrée ou de sortie ainsi que les mémoires associées.



FigureIV.16:Unité de réception du bloc codeur ou unité d'émission du bloc décodeur



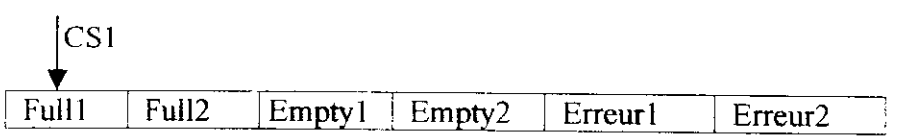
FigureIV.17 : Unité d'émission du codeur



FigureIV.18: Unité de réception du décodeur

3.3. REGISTRE D'ETAT

Le registre d'état contient des signaux qui indiquent l'état des différents blocs constitutifs du chemin de données, afin d'informer le système maître de l'état du système esclave.



FigureIV.19 : Le registre d'état

CS1 : le signal de sélection du registre d'état.

Full1, 2: indique respectivement que la mémoire de l'unité d'émission et de réception est pleine ou non.

Empty1, 2 : indique respectivement que la mémoire de l'unité d'émission et de réception est vide ou non.

Error1 : indique à 1 que la mémoire de réception est pleine, avec une donnée (reçue) sur le bus de données.

Error2 : indique à 1 que la mémoire démission est vide, avec une demande d'émission.

L'interruption IRQ à l'entrée du système maître: à l'état 1, indique au système maître soit la mémoire de réception est pleine ou la mémoire démission est pleine.

3.4. SEQUENCEUR

Le séquenceur génère les différents signaux de commande aux différents blocs constitutifs du chemin de données (la pile d'acquisition de données FIFOs, RAMs, Codeur/Décodeur, registres d'états), suivant le contenu des deux registres d'état et les signaux provenant du système maître(R/W, signal d'horloge, ...) , afin de recevoir des données à coder, ensuite les coder, pour le codeur, ou encore de recevoir des données codées puis les décodée, dans le cas du système décodeur.

Les différents signaux du chemin de données (R1, W1, R2, W2, SEL) sont générés par le séquenceur représenté en figure IV.20. Les entrées du séquenceur, est principalement le contenu du registre d'état et les signaux de contrôle générés par le système maître (R/W, CLK, Reset, sélection des boîtiers, etc....). La logique de fonctionnement du séquenceur est donnée par l'organigramme de la figureIV.21 et IV.22.

Les 3 principales phases de fonctionnement de notre système sont données comme suit :

- Phase de lecture

  Cette étape consiste à récupérer la donnée à traiter du bus de donnée, et faire l'écrire dans la mémoire de l'unité d'émission, au front montant de l'horloge CLK s'il y'a une demande d'écriture de l'extérieur (RW=0). Le séquenceur génère alors les signaux nécessaire qui permettent l'écriture (W1=0) dans l'adresse correspondante définie par le contrôleur de la pile FIFO.

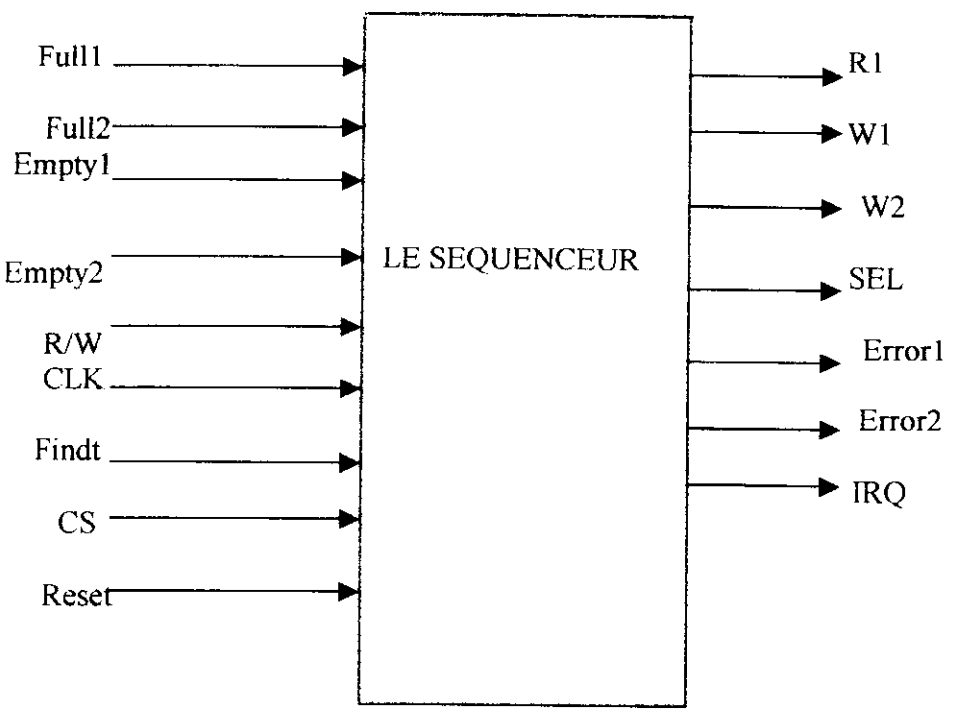
- Phase de traitement

  Au front descendant de l'horloge CLK, la donnée à traiter est lue à partir de la mémoire de réception dans l'adresse de lecture donnée par le FIFO suivie par un signal de sélection du bloc de traitement (SEL=1), le résultat alors se charge dans la mémoire du bloc d'émission dans l'adresse définie par le FIFO.

- Phase d'écriture

  Elle consiste à envoyer le résultat de traitement à l'extérieur si ce dernier transmet à notre circuit une demande de lecture de résultat (RW=1) au front montant de CLK , l'adresse de lecture est aussi déterminée par le contrôleur FIFO.

Notons biens que ces différentes phases de fonctionnement sont commandées par le séquenceur; qui peut aussi interrompre l'exécution de chacune en cas de problèmes (les mémoires sont pleines et il y'a une demande d'écriture, ou encore demande de lecture de l'extérieur avec des mémoire vides).



FigureIV.20 : Le séquenceur

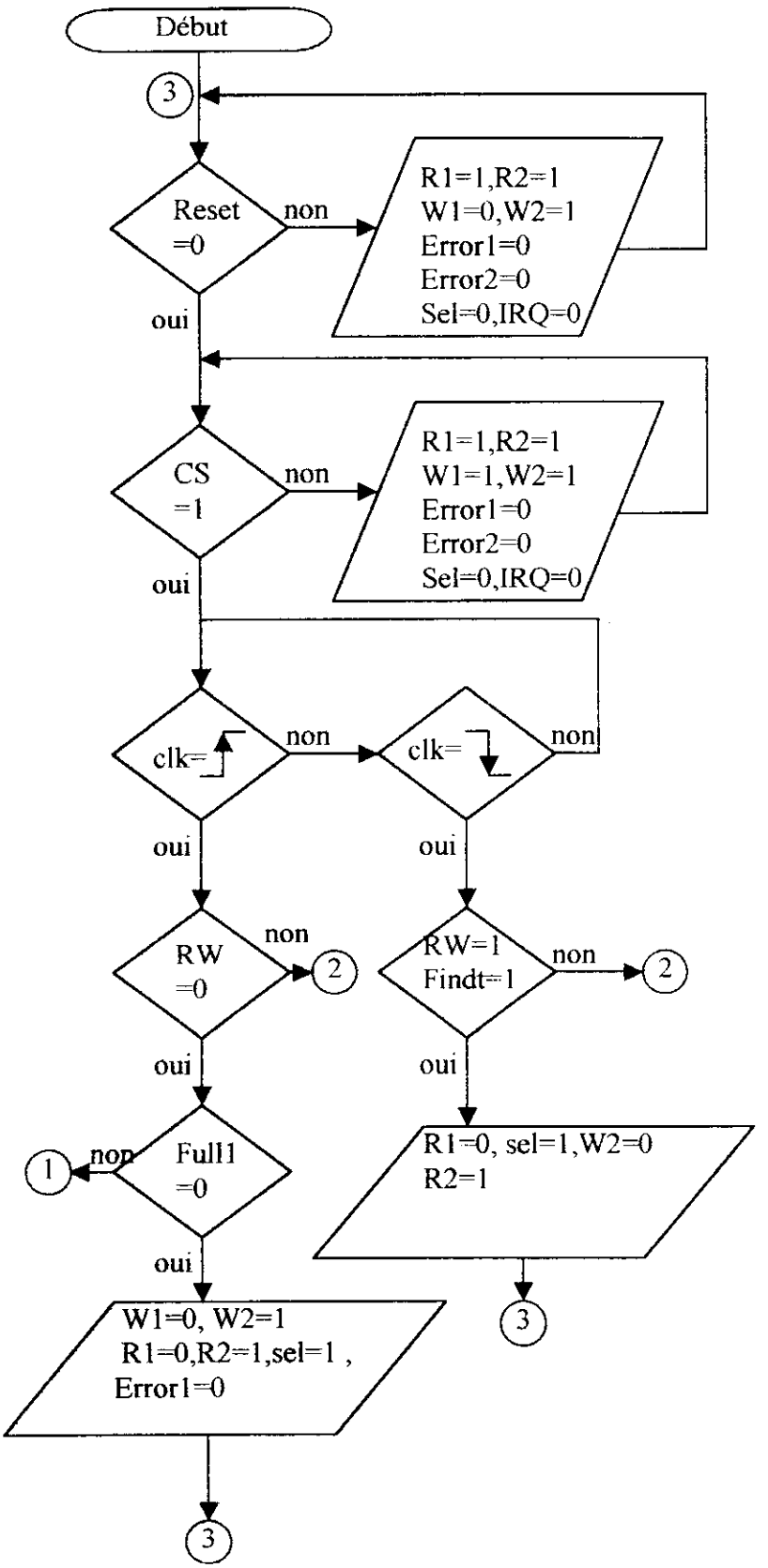


Figure IV.21: Organigramme associé au séquenceur

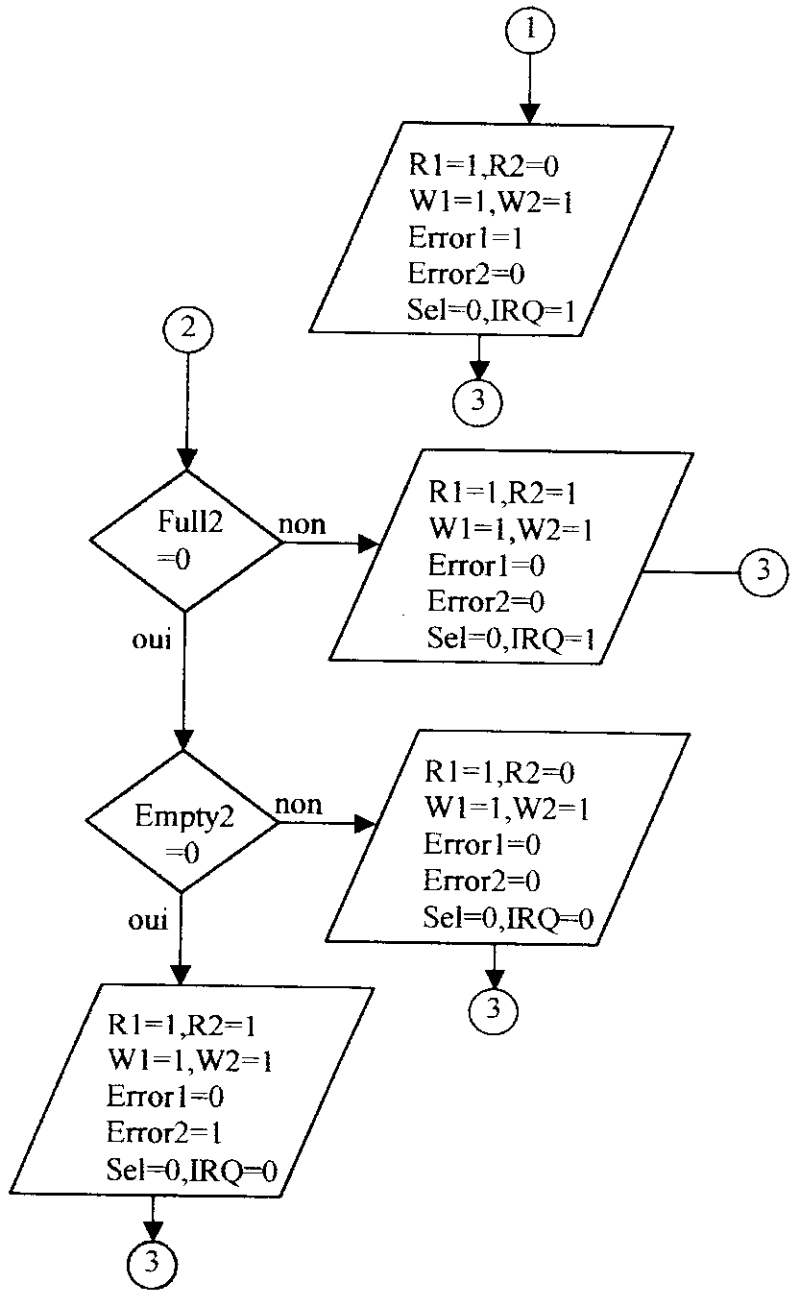


Figure IV.22: Suite de l'organigramme associé au séquenceur

4. RESULTATS DE SIMULATION

4.1. SIMULATION DE LA PARTIE CODEUR

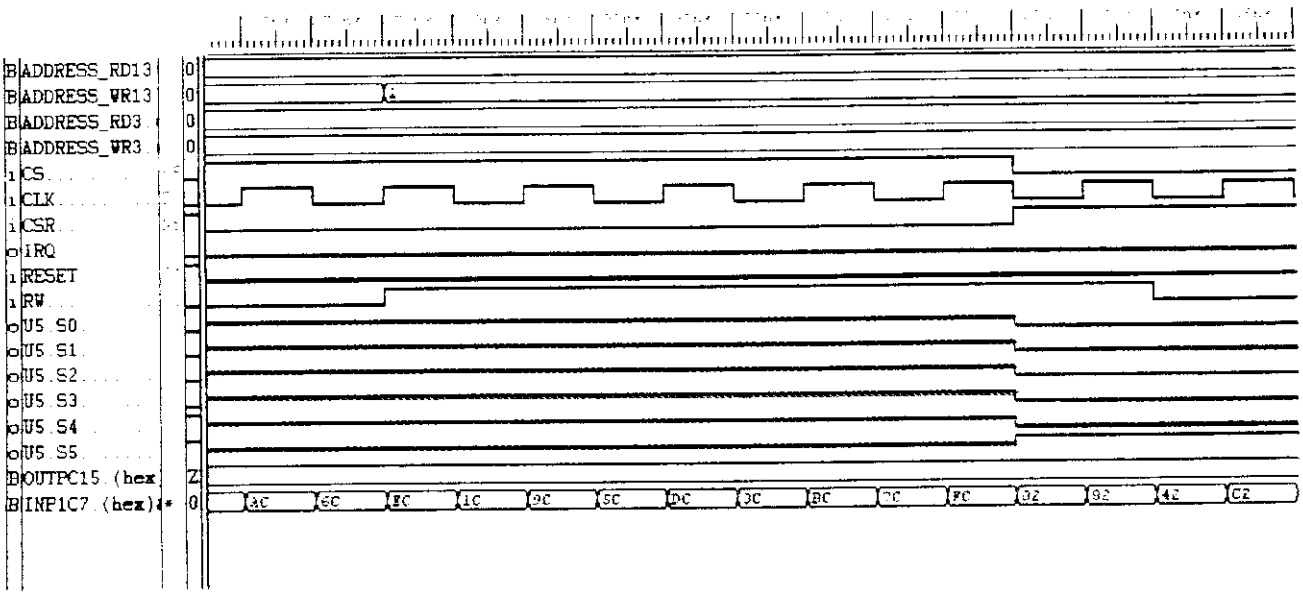


Figure IV.23 : Simulation de la partie codeur

ADRESSE_RD13 : l'adresse de lecture dans l'unité de réception du codeur.
ADRESSE_WR13 : l'adresse d'écriture dans l'unité de réception du codeur.
ADRESSE_RD3 : l'adresse de lecture dans l'unité d'émission du codeur.
ADRESSE_RW3 : l'adresse d'écriture dans l'unité d'émission du codeur.
CS : sélection du système de codage.
CSR : sélection du registre d'état.
IRQ : sortie d'interruption.
RESET : remise à zéro du système.

Dans le cas de sélection du registre d'état :
S0 : le bit error1 du registre d'état.
S1 : le bit error2 du registre d'état.
S2 : le bit full1 du registre d'état
S3 : le bit full2 du registre d'état.
S4 : le bit empty1 du registre d'état.
S5 : le bit empty2 du registre d'état.

Dans le cas de sélection du système codeur :
Les bits S0...S5 représentent les bits de sortie OUTPC(0)...OUTPC(15) respectivement.

RW : le signal lecture écriture.
OUTPC15 : sortie de données codées (sur 16 bits).
INP17 : entrée de données à coder (sur 8 bits).

CLK : l'horloge de synchronisation.

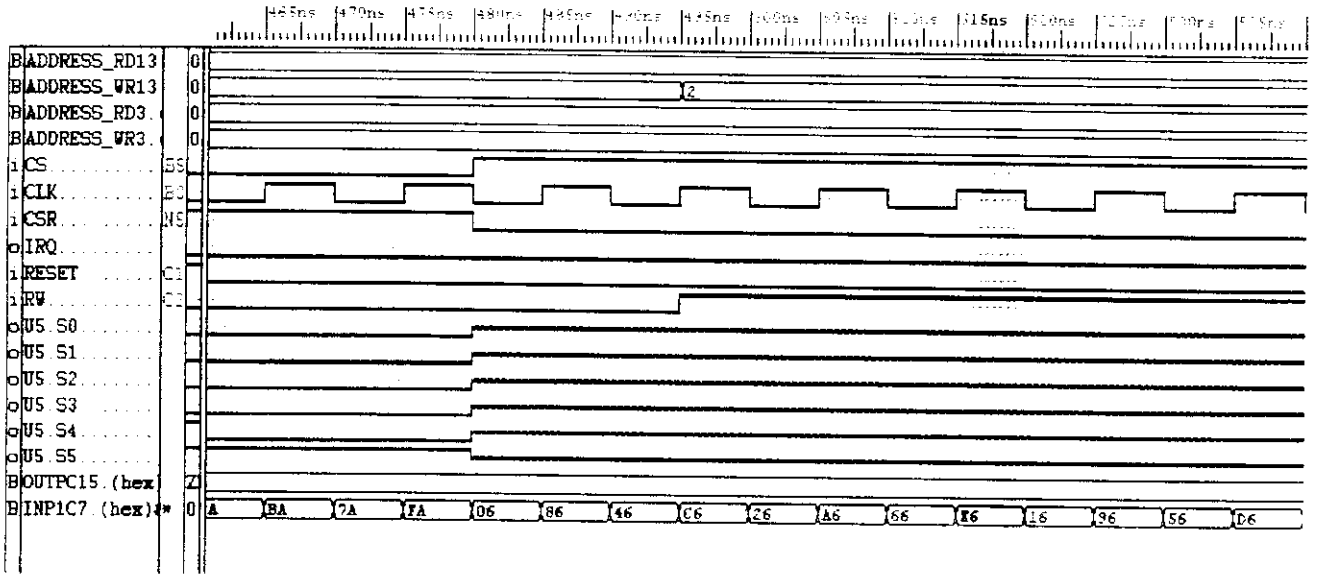


Figure IV.24 : Simulation da la partie codeur (suite).

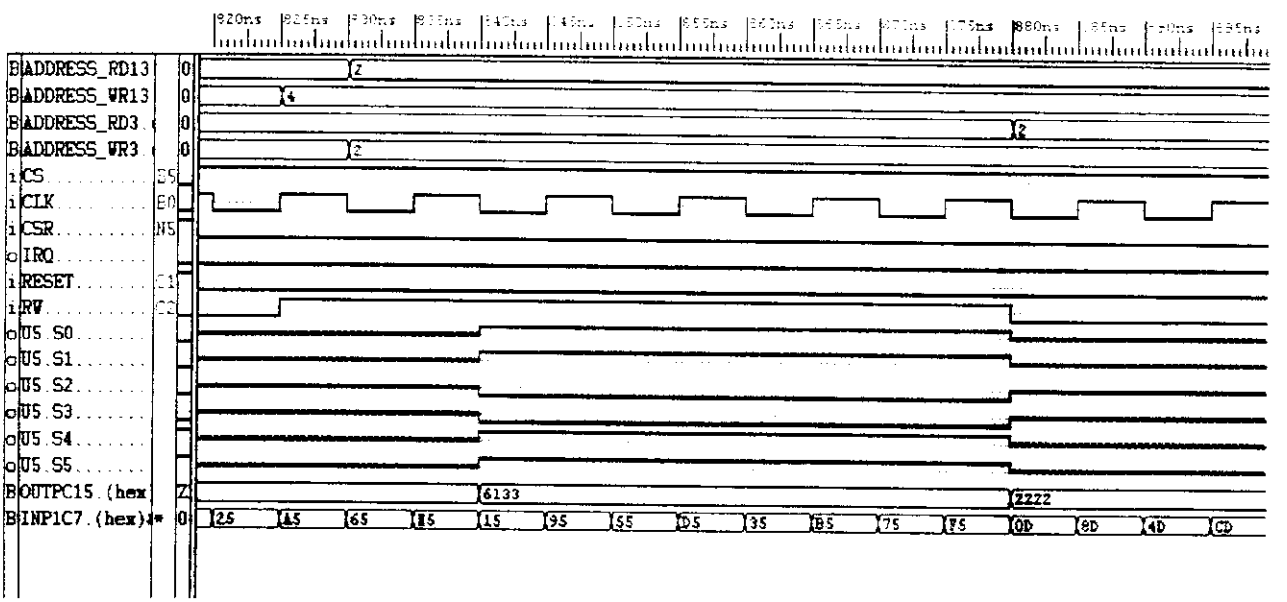


Figure IV.25 : Simulation de la partie Codeur (Suite)

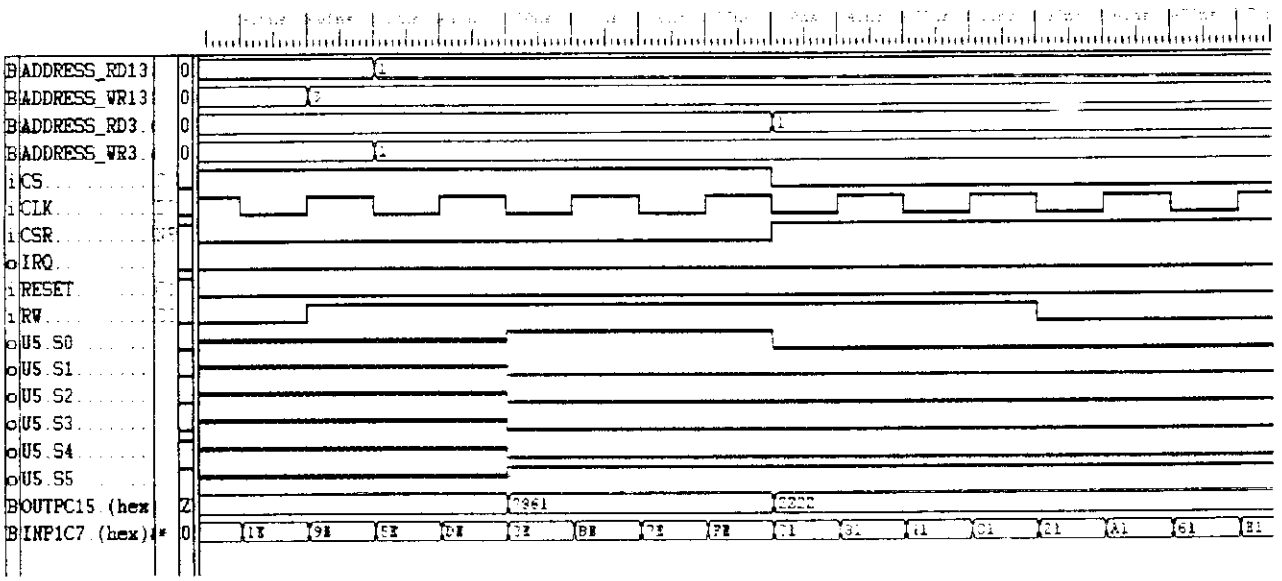


Figure IV.26 : Simulation de la partie Codeur (Suite)

SIMULATION DE LA PARTIE DECODEUR

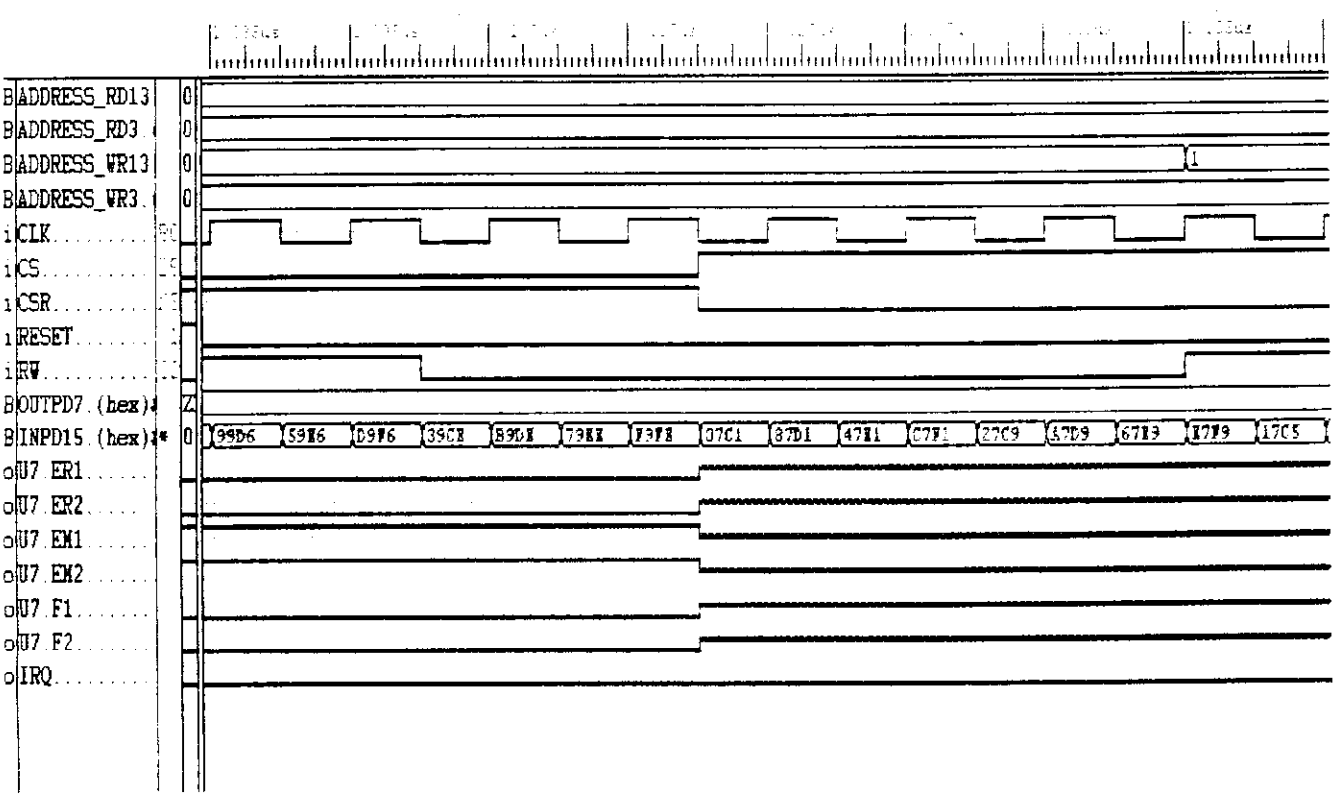


Figure IV.27 : Simulation de la partie décodeur

ADRESSE_RD13 : l'adresse de lecture dans l'unité de réception du décodeur.
ADRESSE_WR13 : l'adresse d'écriture dans l'unité de réception du décodeur.
ADRESSE_RD3 : l'adresse de lecture dans l'unité d'émission du décodeur.
ADRESSE_RW3 : l'adresse d'écriture dans l'unité d'émission du décodeur.
CS : sélection du système décodeur.
CSR : sélection du registre d'état.
RESET : remise à zéro du système.
RW : signal lecture/écriture.
CLK : horloge de synchronisation.
IRQ : interruption.
OUTPD7 : données de sortie décodées (sur 8 bits).
INPD15 : données d'entrée à décoder (sur 16 bits).

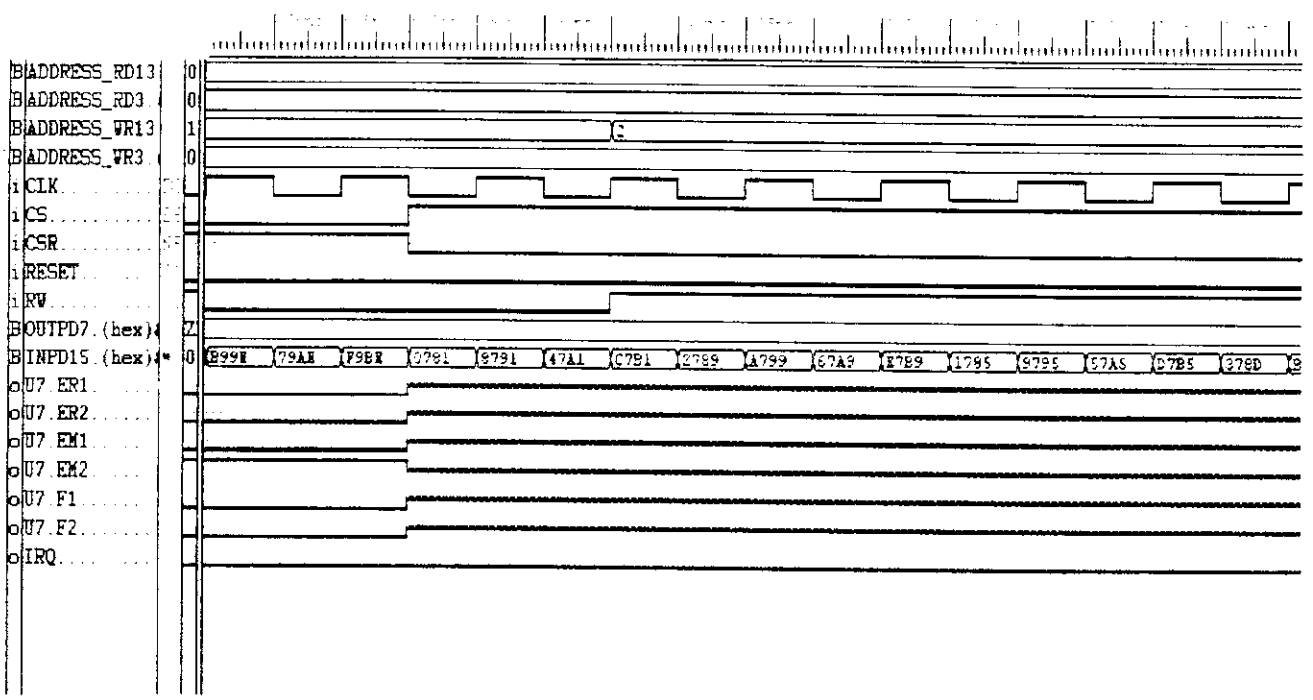


Figure IV.28 : Simulation de la partie décodeur (Suite)

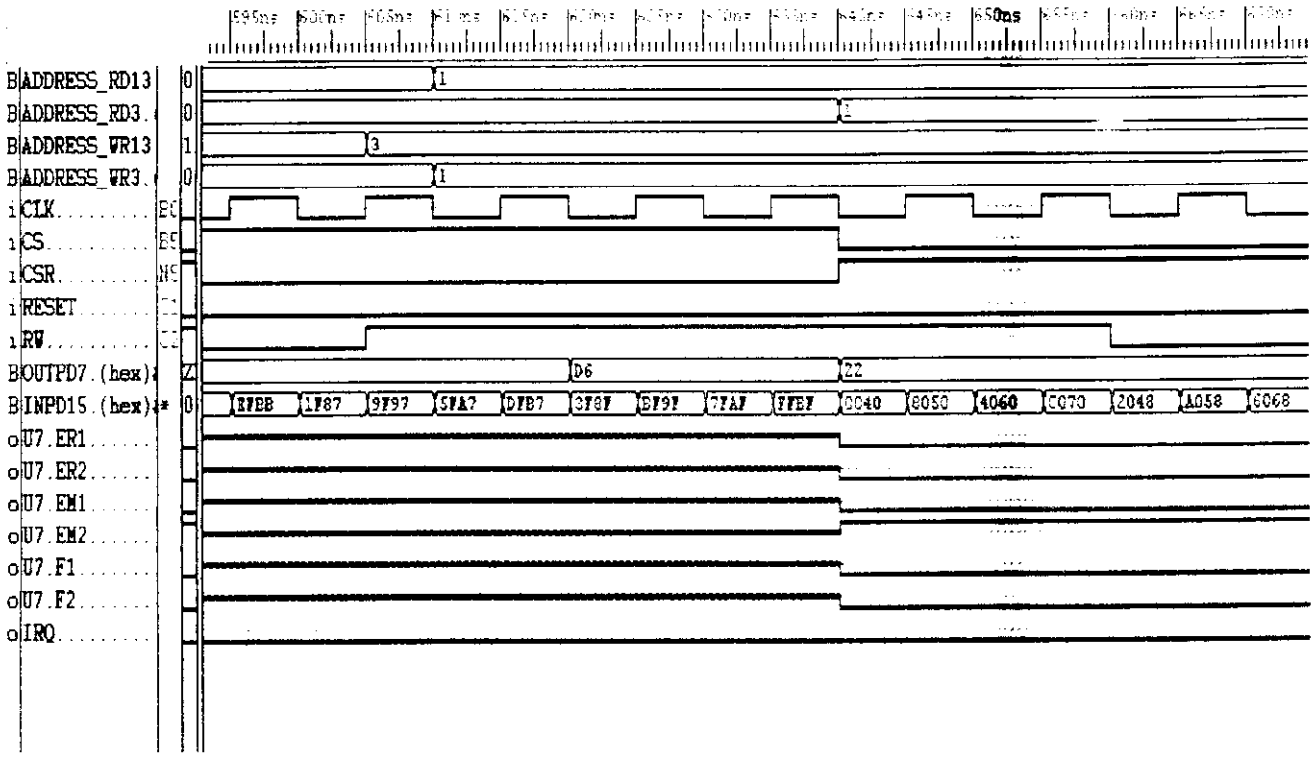


Figure IV.29 : Simulation du décodeur (Suite)

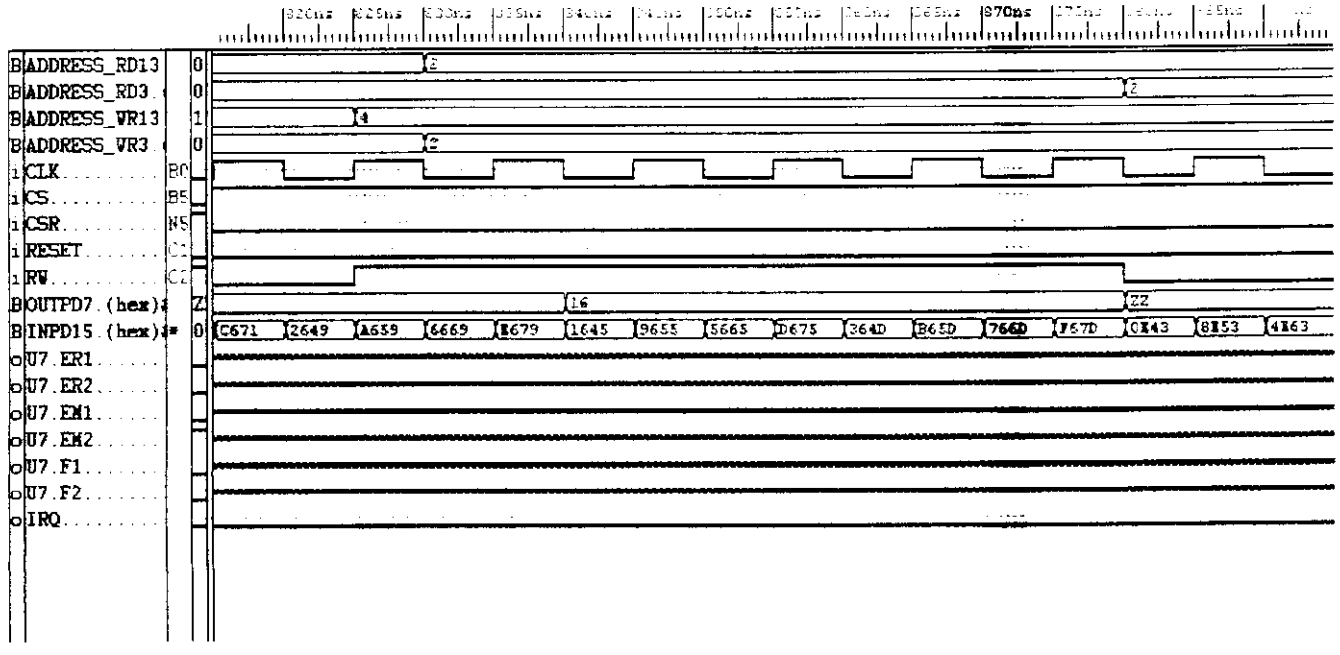


Figure IV.30 : Simulation de la partie décodeur (suite)

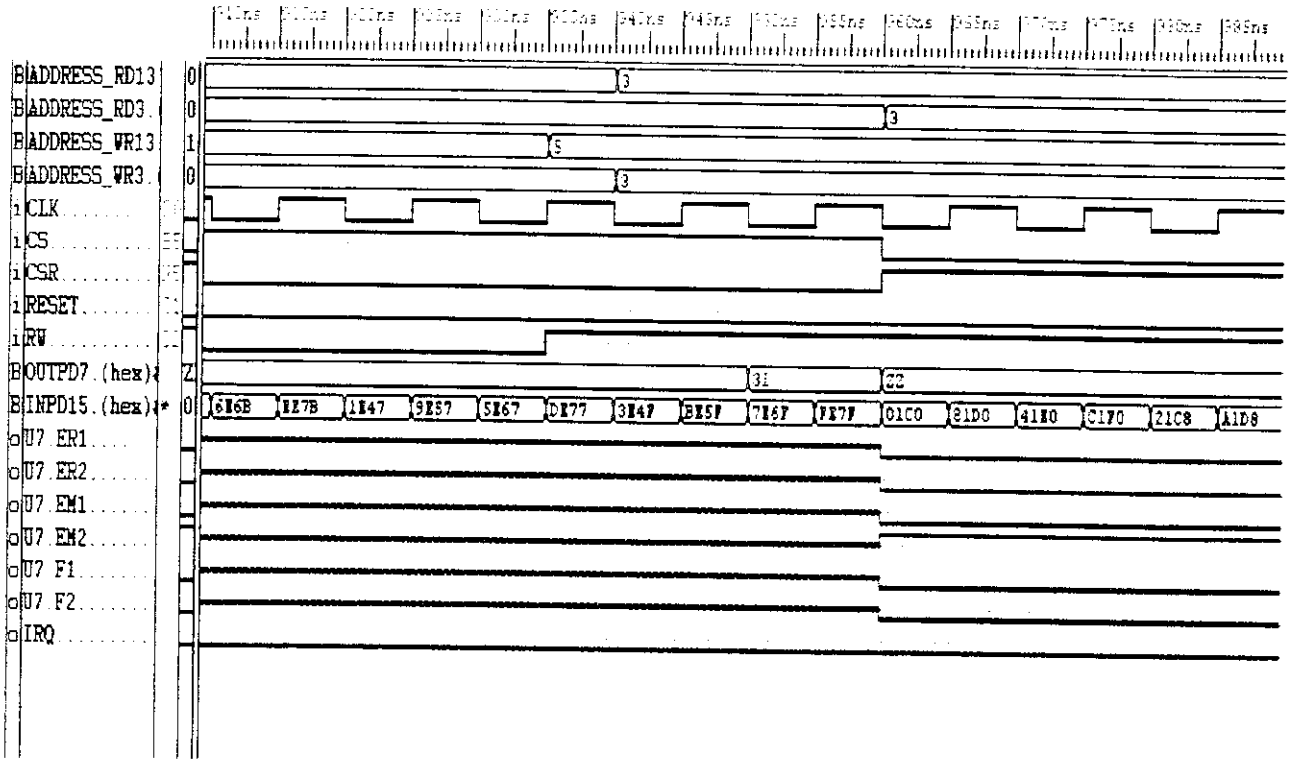


Figure IV.31 : Simulation de la partie décodeur (Suite)

5. LES RAPPORTS D'IMPLEMENTATION

5.1. LA PARTIE CODEUR

5.1.1. LE FICHIER DE CONTRAINTES

Nom de la Broche	Direction	Numéro de la Broche
BD0	**Bidirectionnelle**	**P25**
BD1	**Bidirectionnelle**	**P26**
BD2	**Bidirectionnelle**	**P27**
BD3	**Bidirectionnelle**	**P28**
BD4	**Bidirectionnelle**	**P29**
BD5	**Bidirectionnelle**	**P35**
BD6	**Bidirectionnelle**	**P36**
BD7	**Bidirectionnelle**	**P37**
BD8	**Bidirectionnelle**	**P38**
BD9	**Bidirectionnelle**	**P39**

DB10	**Bidirectionnelle**	**P40**
DB11	**Bidirectionnelle**	**P41**
DB12	**Bidirectionnelle**	**P44**
DB13	**Bidirectionnelle**	**P45**
DB14	**Bidirectionnelle**	**P46**
DB15	**Bidirectionnelle**	**P47**
IRQ	**Sortie**	**P56**
RESET	**Entrée**	**P60**
RW	**Entrée**	**P59**
CSR	**Entrée**	**P57**
CS	**Entrée**	**P65**

Tableau IV.1 : Fichier de contraintes (partie codeur)

5.1.2. LES RESSOURCES EXPLOITEES

Nombre de CLBs 8 sorties de 196 4 %
Nombre de Bascules de CLB : 0
Nombre de Latches de CLB : 0
LUTs à 4 entrées : 14
LUTS à 3 entrées : 0
Nombre d'IOBs 20 de 65 30 %
Bascules d'IOB : 0
Latches d'IOB : 0
Nombre de TBUFS 8 de 448 1 %
Nombre de chemins 33
Nombre de nœuds 25
Nombre de connexions 55

5.1.3. LES DELAIS

Le délai maximum de propagation entre noeuds 11,912 ns.

5.2. LA PARTIE DECODEUR

5.2.1. LE FICHIER DE CONTRAINTES

Nom de la Broche	Direction	Numéro de la Broche
BD0	**Bidirectionnelle**	**P25**
BD1	**Bidirectionnelle**	**P26**
BD2	**Bidirectionnelle**	**P27**
BD3	**Bidirectionnelle**	**P28**
BD4	**Bidirectionnelle**	**P29**
BD5	**Bidirectionnelle**	**P35**
BD6	**Bidirectionnelle**	**P36**
BD7	**Bidirectionnelle**	**P37**

BD8	**Bidirectionnelle**	**P38**
BD9	**Bidirectionnelle**	**P39**
DB10	**Bidirectionnelle**	**P40**
DB11	**Bidirectionnelle**	**P41**
DB12	**Bidirectionnelle**	**P44**
DB13	**Bidirectionnelle**	**P45**
DB14	**Bidirectionnelle**	**P46**
DB15	**Bidirectionnelle**	**P47**
IRQ	**Sortie**	**P56**
RESET	**Entrée**	**P60**
RW	**Entrée**	**P59**
CSR	**Entrée**	**P57**
CS	**Entrée**	**P65**

Tableau IV.1 : Fichier de contraintes (partie décodeur)

5.2.2. LES RESSOURCES EXPLOITEES

Nombre d' IOBs externes	25 de 65 38%
Bascules de IOB :	0
Latches de IOB :	0
LUTs à 4 entrées :	69
LUTS à trois entrées :	1
Nombre de CLBs	62 de 196 31%
Nombre de bascules CLB :	25 de 392 6%
Nombre de latches CLB :	0
LUTs à 4 entrées :	113 de 392 28%
LUTs à trios entrées :	1 de 196 1%
Nombre de BUFGLSs	4 de 8 50%
Nombre de TUFs	53 de 448 11%
Nombre de chemins 3095	
Nombre de nœuds 177	
Nombre de connexions 626	

5.2.3. LES DELAIS

Le délai maximum de propagation entre noeuds 13.266 ns.
Période minimale de 88.130 ns (fréquence maximale: 11.347MHz)

6. CONCLUSION

Le système décrit dans cette partie permet le codage ou le décodage de données de 8 bits suivant l'algorithme de Hamming. L'approche de conception que nous avons introduit peut aboutir à des adaptations diverses par le truchement du changement du bloc central dédié au traitement.

CONCLUSION GENERALE

Ce présent travail est fondé sur le circuit programmable FPGA XC4005XL représentant le cœur de la carte XS40.

Au début, on a traité des généralités sur les circuits programmables (différentes familles, leurs technologies, leurs architectures, etc....) ainsi que leurs avantages et inconvénients par rapport aux autres circuits programmables afin de situer notre circuit.
On a procédé par la suite à une étude complète d'une famille de FPGA, le XC4000. On a traité les différents éléments de base constitutifs de ce circuit dont les CLB, les IOB, les interconnexions ainsi que l'automate dédié à la programmation, au test et à la communication.

La souplesse et la facilité de programmation de ce type de circuit représente un avantage non négligeable dans toute conception adaptative ou visant le prototypage en réduisant le temps et le coût de développement.

La troisième partie de ce travail a consisté en l'étude et la mise en œuvre de la carte XS40 à base du circuit FPGA XC4005XL. Cette étude présente les différents éléments de base constitutifs de la carte, leurs fonctionnements ainsi que la programmation et le test de la carte.

La méthodologie de conception traitée dans la dernière partie, constitue un élément important qui permet d'illustrer les processus et les méthodes de développement d'une conception en tenant compte de la fonction à réaliser, de la cible, de la méthode de travail du concepteur, etc....

Comme application, on a implémenté un Codeur-Décodeur sur la carte. Celui-ci peut être intégré dans un système à base d'un microprocesseur (système maître), constitué de RAMs, de piles FIFO, de registres, de codeur ou de décodeur. L'ensemble de ces éléments est commandé par un séquenceur.
Cette conception peut être adaptée à d'autres algorithmes en remplaçant le codeur ou décodeur par l'application désirée.

L'architecture du circuit conçu ressemble à l'architecture d'un microprocesseur dédié à la détection et la correction d'erreur.

En terme d'amélioration, on peut redéfinir la séquence de fonctionnement en faisant appel à des techniques d'optimisation architecturales.

L'optimisation qu'on a citée peut être adaptée à notre système vue l'architecture adoptée. On peut par exemple avoir plusieurs tâches qui s'exécutent en même temps telle que la récupération de données à traiter ou l'envoie des résultats de traitement simultanément avec l'exécution d'un traitement donné (dans notre cas codage ou décodage). Ceci rejoint alors la notion de pipeline.

On peut aussi augmenter la performance de notre conception en multipliant les ressources utilisées (mémoires, bloc de traitement....) pour que plusieurs données soient traitées en parallèle.
La conception obtenue dans ce cas réalise le codage et le décodage en parallèle.
On peut aussi introduire une autre amélioration en intégrant le codeur et le décodeur dans le même circuit tout en diminuant les ressources utilisées (RAMs, FIFO, etc....); le codage ou le décodage se fera en mode alterné.

BROCHAGE DU CIRCUIT FPGA XC4000

Ce tableau montre la description, le nom le numéros des pins du circuit FPGA De la famille XC4000.

N° de la broche	Nom	description
1	GND	GROUND (la masse)
2	VCC (3.3 V)	Alimentation du circuit FPGA (3.3 V)
3	I/O (A8)	E/S GENERALE
4	I/O (A9)	E/S GENERALE
5	I/O (A10)	E/S GENARALE
6	I/O (A11)	E/S GENERALE
7	I/O (A12)	E/S GENERALE
8	I/O (A13)	E/S GENERALE
9	I/O (A14)	E/S GENERALE
10	SGCK(A15, E/S)	E/S GENERALE
11	VCC (3.3V)	Alimentation du circuit FPGA
12	GND	GROUND
13	PGCK1(A16, E/S)	Horloge de l'oscillateur externe, peut être utilisée comme E/S
14	I/O (A17)	
15	I/O (TDI)	TDI (test de la donnée d'entée)
16	I/O (TCK)	TCK (test de l'horloge)
17	I/O (TMS)	Test du mode
18	I/O	E/S GENERALE
19	I/O	E/S GENERALE
20	I/O	E/S GENERALE
21	GND	
22	VCC (3.3V)	
23	I/O	E/S GENERALE
24	I/O	E/S GENERALE
25	I/O	E/S GENERALE
26	I/O	E/S GENERALE
27	I/O	E/S GENERALE
28	I/O	E/S GENERALE
29	SGCK2(E/S)	E/S
30	M1	Entrée durant la configuration, et après configuration comme signal de sortie 3 états.

		Avec les signaux M0, M1 et M2 déterminent le mode de configuration du circuit FPGA.
31	GND	GROUND
32	M0	Entrée durant et après configuration. Les signaux M0, M1 ,M2 déterminent le mode de configuration du circuit FPGA.
33	VCC	VCC (3.3 V), alimentation du circuit FPGA.
34	M2	Entrée durant et après configuration. Les signaux M0, M1 et M2 sont définissent le mode de configuration du circuit FPGA.
35	PGCK2	Comme entrée utilisée pour piloter un buffer global BUFGP. Utilisé pour router des signaux d'horloge avec un minimum délai de propagation.
36	HDC	Comme sortie durant la configuration, cette broche indique que la configuration n'est pas terminée. Après configuration, HDC est utilisée comme broche d'E/S.
37	/LDC	Peut être utilisée comme entrée de contrôle lors de la configuration, qui indique que la configuration n'est pas terminée. Après configuration cette broche peut être utilisée comme E/S.
38	E/S	Broche d'E/S.
39	E/S	broche d'E/S.
40	E/S	Broche d'E/S
41	/INIT	Avant et durant configuration, /INIT est utilisé comme signal bidirectionnel. ce signal est utilisé pour effacer la configuration en mémoire interne.
42	VCC	Alimentation du circuit 3.3V
43	GND	La masse (GROUND)
44	E/S	Broche d'E/S.
45	E/S	Broche d'E/S
46	E/S	Broche d'E/S
47	E/S	Broche d'E/S
48	E/S	Broche d'E/S
49	E/S	Broche d'E/S
50	E/S	Broche d'E/S
51	SGCK3	Cette entrée globale secondaire, dédiée au pilotage des buffers BUFGS, et au routage des signaux d'horloge avec un minimum délai de propagation. Cette broche est utilisée comme broche d'E/S.
52	GND	GROUND
53	DONE	Comme sortie, DONE indique que la processus de configuration est terminé. Comme entrée, un niveau bas sur cette broche doit être configuré pour valider les sorties.
54	VCC	Broche d'alimentation 3.3V
		L'entré /PROGRAM est active à l'état bas, qui

55	/PROGRAM	force le FPGA pour effacer sa mémoire de configuration. Quand l'entrée /PROGRAM Passe à l'état haut, le FPGA termine le cycle d'effacement Courant, et se prépare pour exécuter un autre.
56	E/S	Broche d'E/S
57	PGCK3	Comme entrée utilisée pour piloter un buffer global BUFGP. Utilisé pour router des signaux d'horloge avec un minimum délai de propagation.
58	E/S	Broche d'E/S
59	E/S	Broche d'E/S
60	E/S	Broche d'E/S
61	E/S	Broche d'E/S
62	E/S	Broche d'E/S
63	VCC	Alimentation 3.3 V
64	GND	GROUND
65	E/S	Broche d'E/S
66	E/S	Broche d'E/S
67	E/S	Broche d'E/S
68	E/S	Broche d'E/S
69	E/S	Broche d'E/S
70	RDY//BYSY	Comme sortie durant la configuration, et comme entrée/sortie après configuration. Durant le mode de configuration périphérique, Cette broche indique quand il est possible d'écrire un autre octet de données dans le circuit FPGA. Le même statut est observé en D7 dans le mode de configuration périphérique asynchrone. après configuration, RDY//BUSY est utilisé comme broche d'E/S.
71	DIN	Durant le mode de configuration esclave série, et le mode master de configuration, DIN est l'entrée de données au front montant de l'horloge CCLK.
72	(SGCK4, DOUT)	DOUT est le donnée de sortie, qui permet de piloter l'entée DIN de programmation des FPGA en mode esclave série. SGCK4 : Cette entrée globale secondaire, dédiée au pilotage des buffers BUFGS, et au routage des signaux d'horloge avec un minimum délai de propagation. Cette broche est utilisée comme broche d'E/S après configuration.

73	CCLK	Comme E/S durant la configuration, et comme Entrée après configuration. Durant la configuration, la signal CCLK est configuré comme sortie dans le mode maître et dans le mode périphérique asynchrone, et se signal d'horloge est configuré comme entrée dans le mode esclave et le mode périphérique synchrone.
74	VCC	Alimentation du FPGA 3.3 V.
75	TDO	Comme sortie durant et après configuration. Quand la boundary scan est utilisé, cette broche permet le test de la donnée de sorti. Si Le boundary scan n'est pas utilisé, cette broche sera une sortie trois états, après que la configuration est terminée.
76	GND	GROUND
77	I/O	Broche d'E/S
78	PGCK4	Comme entrée utilisée pour piloter un buffer global BUFGP. Aussi Utilisé pour router des signaux d'horloge avec un minimum délai de propagation.
79	I/O	Broche d'E/S
80	I/O	Broche d'E/S
81	I/O	Broche d'E/S
82	I/O	Broche d'E/S
83	I/O	Broche d'E/S
84	I/O	Broche d'E/S

Tableau A.1 : Brochage du FPGA famille XC4000

ANNEXE B

CODE DE HAMMING DETECTEUR ET CORRECTEUR D'UNE ERREUR

L'algorithme de hamming permet de détecter et de corriger une erreur dans une trame de bits [2].

Pour la correction d'une seule erreur, le nombre des correcteurs 2^n doit être égal à m+1 ou même plus grand, afin de pouvoir indiquer une erreur dans un des m symboles du mot réceptionné ou pour indiquer qu'il n'y a pas d'erreurs, Donc :

$$2^n \geq m + 1 \qquad \textbf{(B.1)}$$

Ou bien

$$2^n \geq k + m + 1 \qquad \textbf{(B.2)}$$

sont les relations déterminant le nombre des symboles de contrôle lorsque le nombre k des symboles d'information est donné, pour le cas de la correction d'une seule erreur.

Le code Hamming est caractérisé par une matrice de contrôle **H** où la colonne h_i, est la représentation binaire du nombre i :

$$\mathbf{H}=[h_1\, h_2\, h_3 \ldots . h_m]=\begin{pmatrix} 0 & 0 & 0 & \ldots\ldots 1 \\ 0 & 0 & 0 & \ldots\ldots 1 \\ M & M & M & M \\ 0 & 0 & 0 & \ldots\ldots 1 \\ 0 & 1 & 1 & \ldots\ldots 1 \\ 1 & 0 & 1 & \ldots\ldots 1 \end{pmatrix} \qquad \textbf{(B.3)}$$

Remarque que $h_i + h_j \neq 0$ pour $\forall\, i \neq j$, donc le code peut corriger une erreur.

Le mot erreur avec une seule erreur sera :

$$\epsilon=[\ldots\ldots \alpha_i \ldots\ldots]$$

Si l'on transmet v_j on réceptionne :

$$\mathbf{V_j'} = \mathbf{V_j} + \epsilon \qquad \textbf{(B.4)}$$

$$\mathbf{z} = \mathbf{H}\, \mathbf{V_j'}^{T} = \mathbf{H}_{\epsilon}{}^{T} \qquad \textbf{(B.5)}$$

Ou bien

$$\mathbf{z} = [h_1\ h_2\ h_3 \ldots . h_m] \begin{pmatrix} \mathrm{M} \\ \alpha i \\ \mathrm{M} \end{pmatrix} = h_i \qquad \textbf{(B.6)}$$

C'est-à-dire que le correcteur est la représentation binaire du nombre i qui indique la position dans laquelle il y a erreur.

De ce fait, le décodage devient tout à fait simple, tout ce qu'il y a à faire étant une conversion du binaire au décimal.

Le code Hamming peut corriger toutes les erreurs simples mais ne peut corriger aucune erreur double, raison pour laquelle on l'appelle un code parfait.

Le code pouvant corriger e erreurs en n'import quelle position mais qui ne peuvent corriger aucune configuration particulière de $e+1$ erreurs ou plus s'appellent des *codes parfaits.*

1. CODAGE DU CODE HAMMING

Pour simplifier le calcul, les n positions des symboles de contrôle seront choisies de manière à correspondre aux vecteurs colonne h_i à une seule composante différente de zéro.
Ces positions sont : $2^0, 2^1, 2^2, \ldots, 2^{n-1}$. Les symboles de contrôle seront notés par c_i tandis que ceux d'information par i_j. Avec ces notations un vecteur code pourra être écrit sous la forme :

$$\mathbf{V} = [\, c_1\ c_2\ i_1\ c_3\ \ldots .\ i_j]$$

Les symboles de contrôle seront donnés par la relation :

$$H.V^T = 0 \qquad \textbf{(B.7)}$$

Et donc :

$$[h_1\ h_2\ h_3 \ldots . h_m] \begin{pmatrix} c_1 \\ c_2 \\ i_1 \\ \mathrm{M} \\ i_j \end{pmatrix} = 0 \qquad \textbf{(B.8)}$$

Ou bien :

$$c_1\begin{pmatrix}0\\M\\0\\1\end{pmatrix}+c_2\begin{pmatrix}0\\M\\1\\0\end{pmatrix}+i_3\begin{pmatrix}0\\M\\1\\1\end{pmatrix}+\ldots.+i_j\begin{pmatrix}1\\M\\1\\1\end{pmatrix}=\mathbf{0}\ ; \qquad \textbf{(B.9)}$$

Relation équivalente à **n** équations où les symboles $c_1,c_2,c_3....$ n'interviennent qu'une seule fois, donc peuvent être exprimés en fonction des symboles d'information .

En partant depuis la dernière ligne, on peut écrire :

$c_1 = i_1 + i_2 + \ldots + i_j$

$c_2 = i_1 + i_3 + \ldots + i_j$

$c_3 = i_2 + i_3 + \ldots + i_j$

M M M M

. . . .

. . . .

2. DECODAGE DU CODE HAMMING

A la réception, le mot réceptionné **V'** est introduit dans une mémoire à cellules binaires et puis on calcule le correcteur avec la relation

$$\mathbf{Z = H.V'^T} = \begin{pmatrix}e_1\\M\\e_n\end{pmatrix} = [h_1\ h_2\ h_3....h_m]\begin{pmatrix}c'_1\\c'_2\\M\\i'_j\end{pmatrix} \qquad \textbf{(B.10)}$$

Compte tenu de la structure de la matrice ***H***, on obtient, de façon analogue au calcul précédent :

$e_n = c'_1 + i'_1 + i'_2 + \ldots + i'_j$

$e_{n-1} = c'_2 + i'_1 + i'_3 + \ldots + i'_j$

M M M M M

Le nombre binaire ainsi calculé $(e_1e_2 \ldots e_n)$ sera introduit dans un convertisseur binaire décimal à la sortie duquel on obtiendra un signal indiquant la position de l'erreur et permettant de la sorte de la corriger.

3. AVANTAGES ET INCONVENIENTS DE L'ALGORITHME DE HAMMING

Pour l'avantage, l'algorithme de Hamming permet de détecter une erreur dans un mot de bits d'information, et en suite de corriger cette erreur.

Pour les inconvénients :

- Cette méthode permet de détecter une erreur seulement.
- Pour chaque mot de bits transmis on ajoute des bits de contrôle qui représentent des pertes de bits.

4. EXEMPLE

Considérant le cas du code de HAMMING avec k= 4 symboles d'information. il s'ensuit que n = 3 et que m = 7. La matrice de HAMMING correspondante sera :

$$\boldsymbol{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Le mot code est :

$$\mathbf{V} = [\mathbf{c1}\ c_2\ i_1\ c_3\ i_2\ i_3\ i_4]$$

Les relations déterminant les symboles de contrôle sont :

$c_1 = i_1 + i_2 + i_4$

$c_2 = i_1 + i_3 + i_4$

$c_3 = i_2 + i_3 + i_4$

Le codage peut être effectué au moyen du schéma de la figure suivante, le calcul des symboles de correction sera fait avec des additionneurs modulo 2.

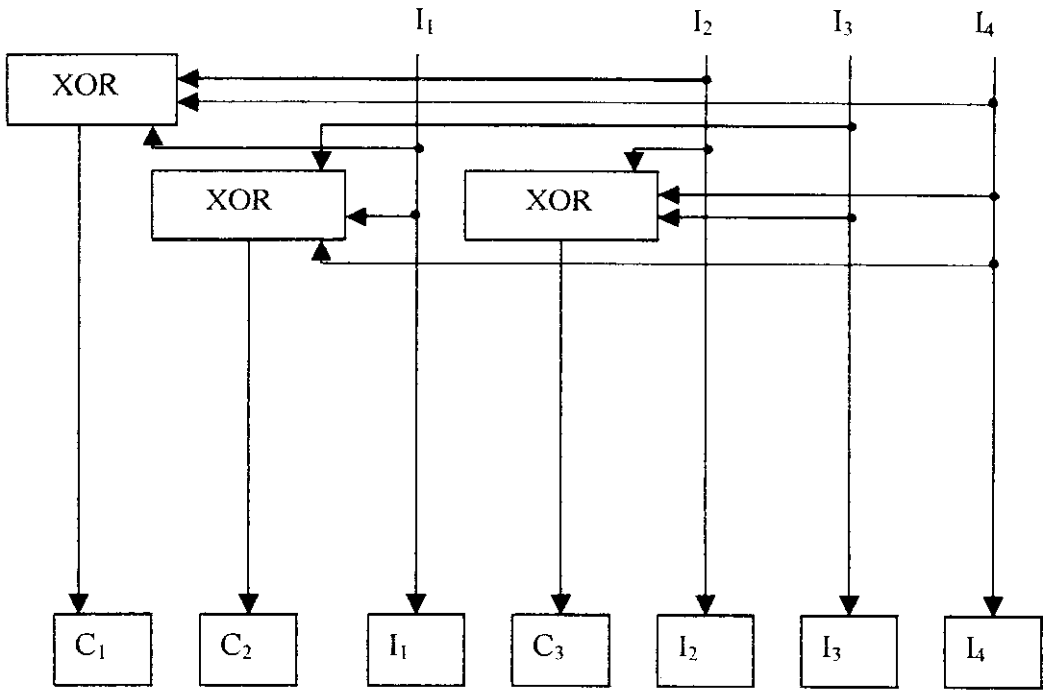


Figure B.1 : Schéma d'un codeur pour un code Hamming correcteur d'une erreur

A la réception, le décodage peut se faire au moyen du schéma de la figure B.2, tandis que le calcul des correcteurs, avec les relations :

$e_1 = c'_3 + i'_2 + i'_3 + i'_4$
$e_2 = c'_2 + i'_1 + i'_3 + i'_4$

$e_3 = c'_1 + i'_1 + i'_2 + i'_4$

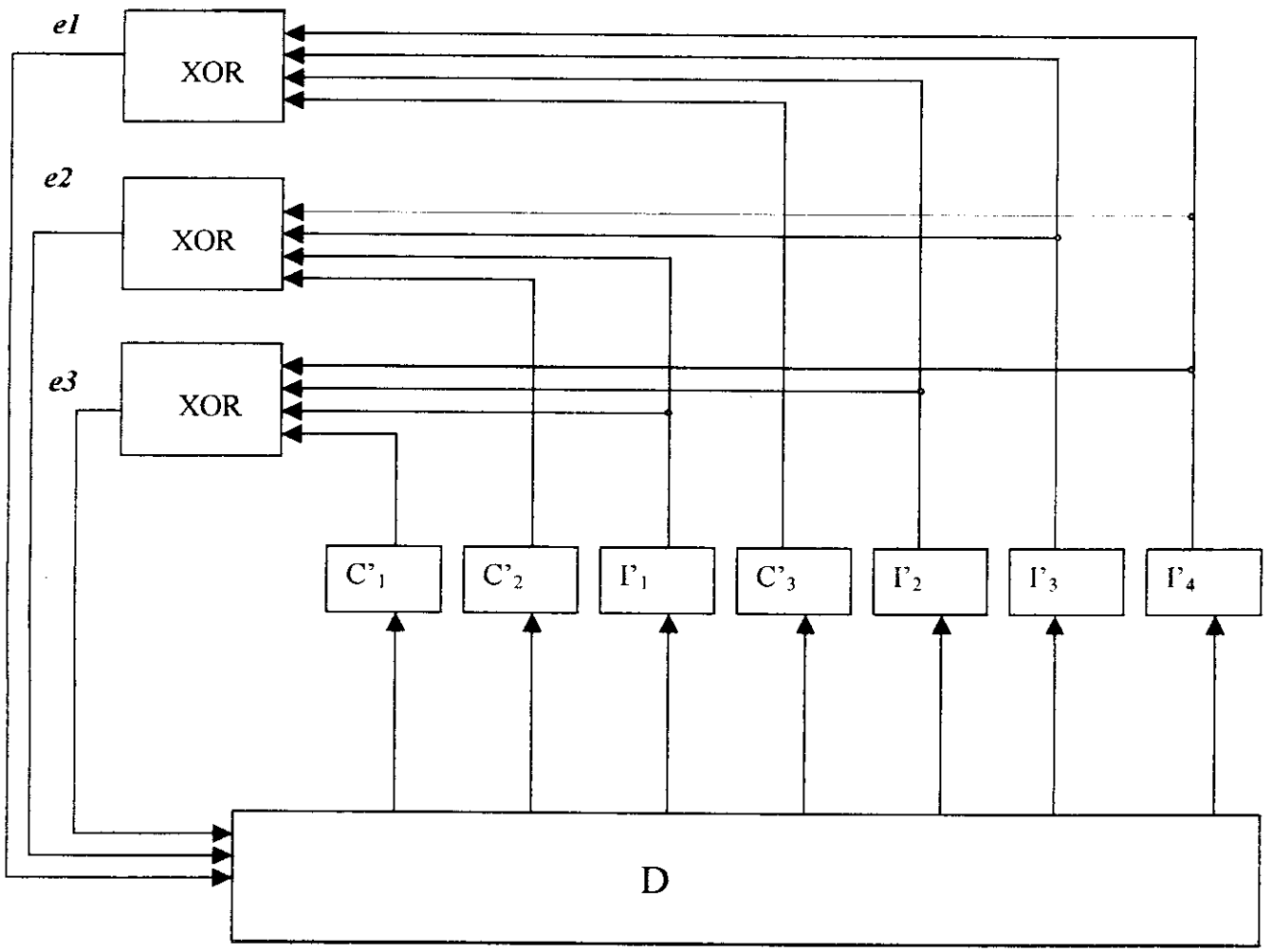


Figure B.2 : schéma d'un décodeur pour un code de Hamming correcteur d'une seule erreur dans un mot de quatre bits d'information (I1, I2, I3, I4)

Dans cet exemple on constate qu'on transmis et on reçoit quatre bits d'information, et trois bits de contrôle.

A partir de cet exemple on peut réaliser des systèmes à 8, 16, 24, 32,....2^n bits d'information en ajoutant 3 bits de contrôle à chaque mots de 4 bits, ainsi le système réalise une détection d'une seule erreur dans chaque mot de quatre bits, en faisant le traitement de chaque mot de quatre bits séparément.

BIBLIOGRAPHIE

[1] M. Robert, "ASIC et logiciels CAO associés ", *Techniques de l'Ingénieur, E2492.*

[2] A. Spataru, "Fondements de la théorie de la transmission et de l'information", *Première édition, Presses Polytechniques de Romandes*, 1987.

[3] D. Van Den Bout, "The practical XILINX designer lab book version 1.5", New *Jersey: Pentice Hall*, 1999.

[4] L. Duterieux. D. Demigny, "Logique programmable, Méthodologie de conception, Le langage VHDL", *édition Eyrolles*, 1997.

[5] D. Adrouche, "Conception d'un cryptoprocesseur à base de l'algorithme DES", *Thèse de Magistère, Ecole Nationale Polytechnique*, Alger.

[6] A. Nkesta, "circuits Programmables:mémoire, PLE, CPLD, FPGA", *édition Ellipses*, 1998.

[7] Data Sheet, "XC400E and XC4000X Series Field Programmable Gate Array", *XILINX*, 1999.
www.xilinx.com

[8] J. Weber, M, Meauder, "Le langage VHDL, cours et exercices", *Edition Dunod*, Paris, 2001.

[9] L. Abdeloual, "Conception des ASIC", *Rapport de Stage, Université de Rennes1/ENSTA/LASTI*, 2003.

[10] M. Aumiaux, "Initiation au langage VHDL", *Edition Masson.*

[11] D. Rabasté, " ASIC et composants à réseaux logiques programmables: PAL, PLD, CPLD, FPGA", *Stage de programmation des CPLD et FPGA en IUFM d'Aix-Marseille*, 2002

[12] "Architecture et Méthode de l'Electronique Numérique", *ENST/COMELEC*, 2002.

[13] N. Julien, "Circuits Logiques Programmables", *Université de Bretagne Sud - Lorient*, 1999.

[14] www.chez.com/jtag/description.html.

[15] F. Anceau, "Conception Algorithmiques des circuit VLSI", *Ecole Polytechnique, Majeur Electronique, Composant, et Système ECS*, 2004.

[16] www.xess.com

[17] G. Pelissier, "Présentation des outils de synthèse d'architectures",2000.

[18] "Development System Reference Guide", *XILINX5 Software Manuels XILINX*, 2002.