Proposition d'une méthodologie de conception de circuits intégrés de communication : réalisation d'un communicateur pour le réseau local FIP

Mario Diaz Nava

THESE

présentée par

Mario DIAZ NAVA

Ingénieur IPN - MEXICO

pour obtenir le titre de

DOCTEUR

DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(arrêté ministériel du 5 juillet 1984)

Spécialité : Informatique

==========

Proposition d'une méthodologie de conception de circuits intégrés de communication Réalisation d'un communicateur pour le réseau local FIP

==========

Soutenue le 1er Juillet 1986, devant la Commission d'Examen :

JURY

Président :	**M.**	**L. BOLLIET**
Examinateurs :	**MM.**	**M. DANG**
		R. GERBER
		G. MAZARE
		J. TAILLEBOIS
Invité :	**M.**	**G. MICHEL**

Laboratoire de Génie Informatique
Institut de Mathématiques Appliquées de Grenoble

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH
Vice-Présidents : B. BAUDELET
H. CHERADAME
R. CARRE
J.M. PIERRARD

Année universitaire 1984-1985

Professeurs des Universités

E.N.S.E.E.G.

BESSON	Jean	LOUCHET	François
BONNETAIN	Lucien	PARIAUD	Jean-Charles
BONNIER	Etienne	RAMEAU	Jean-Jacques
DURAND	Françis	SOHM	Jean-Claude
GUYOT	Pierre	SOUQUET	Jean-Louis

E.N.S.E.R.G.

BARIBAUD	Michel	GENTY	Pierre
BLIMAN	Samuel	GUERIN	Bernard
BUYLE BODIN	Maurice	POUPOT	Christian
CHENEVIER	Pierre	SERMET	Pierre
COHEN	Joseph	ZADWORNY	François
COUMES	André		

E.N.S.I.E.G.

BARRAUD	Alain	JOUBERT	Jean-Claude
BAUDELET	Bernard	JOURDAIN	Geneviève
BLOCH	Daniel	LACOUME	Jean-Louis
BRISSONNEAU	Pierre	LONGEQUEUE	Jean-Pierre
CAVAIGNAC	Jean-François	MASSELOT	Christian
CHARTIER	Germain	MORET	Roger
CHERUY	Arlette	PAUTHENET	René
DURAND	Jean-Louis	PERRET	René
FELICI	Noël	PERRET	Robert
FOULARD	Claude	POLOUJADOFF	Michel
GAUBERT	Claude	SABONNADIERE	Jean-Claude
IVANES	Marcel	SCHLENKER	Claire
JALINIER	Jean-Michel	SCHLENKER	Michel
JAUSSAUD	Pierre		

E.N.S.H.G.

BOIS	Philippe	LESPINARD	Georges
BOUVARD	Maurice	MOREAU	René
LESIEUR	Marcel	PIAU	Jean-Michel

E.N.S.I.M.A.G.

ANCEAU	François	MOSSIERE	Jacques
FONLUPT	Jean	ROBERT	François
LATOMBE	Jean-Claude	SAUCIER	Gabrielle
MAZARE	Guy	VEILLON	Gérard

U.E.R.M.C.P.P.

CHERADAME	Hervé	RENAUD	Maurice
CHIAVERINA	Jean	ROBERT	André
GANDINI	Alessandro	SILVY	Jacques

Professeurs Associés

BLACKWELDER	Ronald	ENSHG
HAYASHI	Hirashi	ENSIEG
PURDY	Gary	ENSEEG

Professeurs à l'Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis
CHATELIN	Françoise

Chercheurs du C.N.R.S.

Directeurs de recherche :

CARRE	René
FRUCHARD	Robert
JORRAND	Philippe
VACHAUD	Georges

Maître de recherche :

ALLIBERT	Michel	JOUD	Jean-Charles
ANSARA	Ibrahim	KAMARINOS	Georges
ARMAND	Michel	KLEITZ	Michel
BINDER	Gilbert	LANDAU	Ioan-Dore
BORNARD	Guy	LASJAUNIAS	Jean-Claude
DAVID	René	MERMET	Jean
DESPORTES	Jacques	MUNIER	Jacques
DRIOLE	Jean	PIAU	Monique
GIGNOUX	Damien	PORTESEIL	Jean-Louis
GIVORD	Dominique	THOLENCE	Jean-Louis
GUELIN	Pierre	VERDILLON	André
HOPFINGER	Emile	SUERY	Michel

Personnalités habilitées à diriger des travaux de recherche
(Décision du conseil scientifique)

E.N.S.E.E.G.

ALLIBERT	Colette	HAMMOU	Abdelkader
BERNARD	Claude	MALMEJAC	Yves (CENG)
BONNET	Roland	MARTIN GARIN	Régina
CAILLET	Marcel	NGUYEN TRUONG	Bernadette
CHATILLON	Catherine	RAVAINE	Denis
CHATILLON	Christian	SAINFORT	(CENG)
COULON	Michel	SARRAZIN	Pierre
DIARD	Jean-Paul	SIMON	Jean-Paul
EUSTATHOPOULOS	Nicolas	TOUZAIN	Philippe
FOSTER	Panayotis	URBAIN	Georges(ODEILLO)
GALERIE	Alain		

E.N.S.E.R.G.

BARIBAUD	Michel	DOLMAZON	Jean-Marc
BOREL	Joseph	HERAULT	Jeanny
CHOVET	Alain	MONLLOR	Christian
CHEHIKIAN	Alain		

E.N.S.I.E.G.

BORNARD	Guy	LEJEUNE	Gérard
DESCHIZEAUX	Pierre	MAZUER	Jean
GLANGEAUD	François	PERARD	Jacques
KOFMAN	Walter	REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine	OBLED	Charles
BOIS	Daniel	ROWE	Alain
DARVE	Félix	VAUCLIN	Michel
MICHEL	Jean-Marie	WACK	Bernard

E.N.S.I.M.A.G.

BERT	Didier	DELLA DORA	Jean
CALMET	Jacques	FONLUPT	Jean
COURTIN	Jacques	SIFAKIS	Joseph
COURTOIS	Bernard		

U.E.R.M.C.P.P.

CHARUEL	Robert

C.E.N.G.

CADET	Jean	NIFENECKER	Hervé
COEURE	Philippe (LETI)	PERROUD	Paul
DELHAYE	Jean-Marc (STT)	PEUZIN	Jean-Claude(LETI)
DUPUY	Michel (LETI)	TAIEB	Maurice
JOUVE	Hubert (LETI)	VINCENDON	Marc
NICOLAU	Yvan (LETI)		

Laboratoires extérieurs

C.N.E.T.

DEMOULIN	Eric	MERCKEL	Gérard
DEVINE	R.A.B.	PAULEAU	Yves
GERBER	Roland		

I.N.S.A. Lyon

GAUBERT	C.

****

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: M. M. MERMET
Directeur des Etudes et de la formation	: M. J. LEVASSEUR
Directeur des Recherches	: M. J. LEVY
Secrétaire Général	: Mle M. CLERGUE

Professeurs de la 1ère Catégorie

COINDE	Alexandre	Gestion
GOUX	Claude	Metallurgie
LEVY	Jacques	Métallurgie
LOWYS	Jean-Pierre	Physique
MATHON	Albert	Gestion
RIEU	Jean	Mécanique-Résistance des matériaux
SOUSTELLE	Michel	Chimie
FORMERY	Philippe	Mathématiques Appliquées

Professeurs de 2ème catégorie

HABIB	Michel	Informatique
PERRIN	Michel	Géologie
VERCHERY	Georges	Matériaux
TOUCHARD	Bernard	Physique industrielle

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

BISCONDI	Michel	Métallurgie
DAVOINE	Philippe	Géologie
FOURDEUX	Angeline	Métallurgie
KOBYLANSKI	André	Métallurgie
LALAUZE	René	Chimie
LANCELOT	Francis	Chimie
LE COZE	Jean	Métallurgie
THEVENOT	François	Chimie
TRAN MINH	Canh	Chimie

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

a Carmen

a mis Padres

a mis hermanos

REMERCIEMENTS

Je tiens à remercier d'abord les membres du jury:

Le Professeur Louis BOLLIET de l'Université de Grenoble II, qui m'a accueilli dans son équipe de recherche lors de mon arrivée en France, m'a aidé lors du démarrage de mes études de 3ème cycle, et a bien voulu accepter de présider ce jury;

Le Professeur Guy MAZARE de l'Institut National Polytechnique de Grenoble qui m'a accepté dans son équipe de recherche au Laboratoire de Génie Informatique;

Michel DANG, Maître de Conférences à l'Université de Grenoble II, qui m'a aidé à définir mon sujet de thèse, m'a conseillé et guidé dans mon travail de recherche et dans la préparation de mon mémoire de thèse;

Roland GERBER, Chef de la Division CCI au CNET-CNS qui m'a accueilli au CNET de Meylan, facilitant ainsi grandement mon travail de recherche;

M. J. TAILLEBOIS, Directeur des Services Techniques du Département Commande de la Société CROUZET qui s'est intéressé de près à mes travaux dans le cadre du projet "FIP capteurs";

et également :

Gérard MICHEL, Directeur du Département Etudes de la Société APTOR et ancien Chef de Département au CNET-CNS, qui n'a jamais hésité à me prodiguer son aide et ses conseils;

Jean-Louis LARDY, Chef du Département MCC au CNET-CNS qui a bien voulu lire mon manuscrit de thèse et me faire part de ses critiques et conseils;

Luc SPONGA, sans qui la partie simulation de circuits de mes travaux n'aurait pu être terminée,

tous les membres du CNET-CNS qui, de près ou de loin, m'ont aidé en mettant à ma disposition toute l'infrastructure matérielle et technique nécessaire à la réalisation de mes études;

Clotilde CHALAND qui a accepté de relire mon manuscrit et d'y apporter les retouches nécessaires à l'écriture du bon français;

Daniel IGLESIAS et les membres du service tirage de l'IMAG qui ont mis la dernière main à la réalisation typographique de ce travail;

Ces remerciements seraient incomplets sans citer le nom du CONACYT (Consejo Nacional de Ciencia y Technologia) qui dans le cadre d'aide à la formation de chercheurs a rendu possible mes études en France.

Que tous soient remerciés à la mesure de leur aide et de leur gentillesse à mon égard.

RESUME:

Dans le cadre de ce travail, il a été spécifié et réalisé un circuit intégré de communication pour le réseau local FIP. Le réseau FIP, projet national, est destiné à permettre la communication entre automates réflexes, capteurs et actionneurs. Le circuit intégré a été spécifié pour permettre soit la connexion de capteurs simples soit la connexion de capteurs intelligents ou des automates au réseau.

La conception de ce circuit intégré "à la demande" a été réalisée à partir d'une méthodologie originale. Cette méthodologie est un autre objectif important de ce travail. Elle est orientée vers la conception de circuits VLSI de communication à partir d'une bibliothèque d'opérateurs flexibles, d'une part pour réduire le temps de conception, et d'autre part pour donner la possibilité aux ingénieurs non-spécialistes en conception de concevoir eux-même leur circuit.

MOTS-CLES:

Réseaux locaux temps réel, circuits intégrés communicateurs, méthodologie de conception, opérateurs flexibles, bibliothèque de cellules.

ABSTRACT:

Within the framework of this work, a communicating integrated circuit has been specified and developed for the FIP local network. The purpose of this national project, the FIP network, is to allow for communication between sensors, actuators and programmable controllers. The integrated circuit has been specified such that it enables either simple sensors or more complex sensors or microcomputers to be connected together.

The design of this 'custom' integrated circuit has been developed from an original methodology. The latter constitutes another important objective of this paper. This methodology is geared towards the design of communicating VLSI circuits based on a flexible operator library in order, on the one hand, to reduce the design time and, on the other hand, to offer engineers who are not design specialists the possibility of designing their circuits themselves.

KEYWORDS:

Real time local networks, communicating integrated circuits, design methodology, flexible operators, cell library.

TABLE DES MATIERES

TABLE DES MATIERES

INTRODUCTION 1

PARTIE 1 GENERALITES

1 LES SYSTEMES DE COMMUNICATION 7
1.1 Principaux constituants d'un réseau de communication 7
1.1.1 Niveaux de communication 9
1.2 Principaux aspects de la réalisation d'un réseau local 11
1.3 L'intérêt d'utiliser les VLSI dans les circuits d'interface 16

2 DIFFERENTES METHODES DE CONCEPTION 21
2.1 Méthodes d'implantation de circuits intégrés 21
2.2 Méthodologies de conception 23
2.2.1 La démarche hiérarchique 23
2.2.2 La régularité 25
2.2.3 L'automatisation des différentes étapes de la conception 26
2.2.4 Les compilateurs de silicium 26
2.3 Conclusions 28

PARTIE 2 METHODOLOGIE

1 METHODOLOGIE PROPOSEE POUR LA CONCEPTION DE CIRCUITS VLSI DE COMMUNICATION 35
1.1 Les outils d'aide à la conception disponibles 35
1.2 Présentation de la méthodologie 36
1.2.1 Approche descendante 37
1.2.2 Construction ascendante 38
1.3 Critères et méthodologie à adopter pour l'implantation de la bibliothèque de cellules 40
1.3.1 Critères d'implantation 40
1.3.2 Implantation 42

2 ANALYSE FONCTIONNELLE DES CIRCUITS DE COMMUNICATION 47
2.1 Les circuits d'interface d'entrées/sorties . 47
2.1.1 Les circuits d'interface parallèle . . . 47
2.1.2 Les circuits d'interface série 52
2.2 Conclusions 57
2.3 Les circuits communicateurs 58

3 LES OPERATEURS FLEXIBLES 63
3.1 Définition des opérateurs flexibles 63
3.2 Fonctions rencontrées dans les circuits analysés 63

3.3 Classification des opérateurs flexibles . . 64
3.4 Présentation des opérateurs flexibles des différentes classes 64
3.4.1 Les opérateurs en tranches 64
3.4.2 Les opérateurs programmables 84
3.4.3 Divers 86
3.5 Outils à utiliser pour générer les opérateurs 87
3.5.1 L O F (Langage d'Opérateurs Flexibles) . 87
3.5.2 G A S P 89

4 PRESENTATION DES BIBLIOTHEQUES DE CELLULES ET D'OPERATEURS FLEXIBLES 95
4.1 La bibliothèque de cellules 95
4.1.1 Amplificateurs compilables 97
4.2 La bibliothèque d'opérateurs 98
4.2.1 Bibliothèque de fiches LOF réalisée . . 98
4.2.2 Les opérateurs programmables 99

PARTIE 3 REALISATION

1 INTRODUCTION 111

2 SPECIFICATIONS DU COMPOSANT FIP-VLSI 115
2.1 Caractéristiques générales du système de communication FIP 115
2.1.1 Le réseau FIP 115
2.1.2 Le protocole de communication 115
2.1.3 Caractéristiques générales du circuit FIP 117
2.2 Méthode d'adressage et structure des trames 118
2.2.1 Méthode d'adressage 118
2.2.2 Formats des trames 118
2.3 Définition des modes de fonctionnement du circuit FIP-VLSI 122
2.4 Interface côté station 123
2.4.1 Critères de choix du type d'interface . 123
2.4.2 Choix de l'Interface 123
2.4.3 Exemple d'utilisation de l'interface . . 125
2.5 Reconnaissance de l'adresse 126
2.5.1 Techniques proposées 126
2.5.2 Avantages et inconvénients de chaque technique 127
2.5.3 Programmation de l'adresse de la station 128
2.6 Interface côté medium 130
2.6.1 Type de Codage. 130
2.6.2 Indépendance vis-à-vis du médium. . . . 131
2.7 Initialisation du circuit 133
2.7.1 Mode de fonctionnement 133

2.7.2 Chargement de l'adresse physique de la station 133
2.7.3 Définition du sens de transfert du bus de données 133
2.8 Définition de la base de temps du circuit . 134
2.9 Définition de l'organisation externe du circuit 135

3 CONCEPTION DU CIRCUIT COMMUNICATEUR FIP-VLSI . . . 139
3.1 Présentation de la méthodologie de conception adoptée 139
3.2 Analyse descendante dans le cas du circuit FIP-VLSI 140
3.2.1 Analyse Fonctionnelle 140
3.2.2 Description du fonctionnement du FIP-VLSI 142
3.2.3 Architecture interne du circuit FIP-VLSI 143
3.2.4 Synthèse des modules 146
3.2.5 Plan de masse 150
3.2.6 Simulation logico-fonctionnelle 150
3.3 Mise en oeuvre de la construction ascendante du circuit 151
3.3.1 Implantation des opérateurs 151
3.3.2 Implantation des modules 153
3.3.3 La couronne de plots 153
3.3.4 Placement et routage des modules du FIP-VLSI 154

CONCLUSIONS 159

BIBLIOGRAPHIE 165

ANNEXES 175

A Les avantages de l'intégration 177

B Caractéristiques de plusieurs circuits d'entrée/sortie série 181

C Outils d'aide à la conception 187

D Bibliothèque de cellules 195

E Exemple d'opérateurs en tranches 213

F Définition des broches du circuit FIP-VLSI . . 229

G Algorithme de contrôle du circuit FIP-VLSI . . 223

INTRODUCTION

Les premières communications informatiques étaient réalisées entre un ordinateur et ses divers périphériques à travers des lignes directes. Aujourd'hui, cette communication est devenue beaucoup plus complexe. Les nouvelles techniques de communication et les possibilités immenses offertes par l'industrie des semiconducteurs ont permis le développement de réseaux de communication à grande distance. Ces réseaux sont utilisés pour la communication entre terminaux et calculateurs éloignés, ou pour l'interconnexion entre plusieurs calculateurs.

Les réseaux locaux constituent l'apport majeur de ces dernières années aux systèmes de communication. Si les premiers réseaux locaux commercialisés, comme ETHERNET, ont été conçus pour des applications bureautiques, on assiste maintenant au développement de réseaux industriels pour des applications de contrôle-commande de procédés. Un grand effort devra cependant être fait dans les prochaines années pour l'entière automatisation des usines. Plusieurs actions vont dans ce sens comme le projet international MAP (Manufacturing Automation Protocoles) [KAM-86] et le projet français FIP (Factory Instrumentation Protocol) [FIP-84]. FIP est une proposition présentée comme complémentaire de MAP.

Dans le cadre de ce travail, nous avons spécifié et réalisé un circuit intégré de communication pour réseau local de type FIP permettant la communication d'automates, de capteurs et d'actionneurs.

La conception de ce circuit intégré "à la demande" a été réalisée à partir d'une méthodologie originale. Cette méthodologie est un autre objectif important de ce travail. Elle est orientée vers la conception de circuits VLSI de communication à partir d'une bibliothèque d'opérateurs flexibles, d'une part pour réduire le temps de conception, et d'autre part pour donner la possibilité aux ingénieurs non-spécialistes en conception de concevoir eux-même leur circuit. Dans un premier temps, cette bibliothèque a été définie pour la réalisation de circuits de communication de faible complexité. Dans des travaux futurs, elle sera complétée pour permettre la conception de circuits plus complexes.

Ce travail est divisé en trois parties :

La première partie (chapitres 1 et 2) fait référence aux systèmes de communication et évoque les problèmes liés à la communication. Les principaux constituants d'un réseau local sont cités, afin de préciser le type de communication étudié. A partir de cette présentation, on souligne l'importance de l'utilisation des circuits intégrés VLSI dans les contrôleurs de communication ainsi que la nécessité de disposer d'outils appropriés pour leur réalisation. Dans un deuxième chapitre, on effectue un survol des différentes méthodologies de réalisation de circuits.

La deuxième partie présente la méthodologie proposée. A partir d'une étude fonctionnelle de quelques-uns des circuits de communication existants, les principales fonctions nécessaires sont dégagées. Une classification des différentes fonctions rencontrées qui peuvent faire l'objet d'une implantation automatique est présentée (proposition des opérateurs flexibles). Dans cette deuxième partie, la bibliothèque de cellules nécessaire pour la construction des opérateurs flexibles est précisée. Un exemple de bibliothèque de ce type est donné en annexe.

La troisième et dernière partie de cette thèse est consacrée aux spécifications du circuit FIP-VLSI déjà mentionné et à sa conception à partir de la bibliothèque d'opérateurs flexibles présentée dans la deuxième partie.

PARTIE 1

GENERALITES

1. LES SYSTEMES DE COMMUNICATION

1 LES SYSTEMES DE COMMUNICATION

Dans ce chapitre, nous évoquons les principaux constituants d'un système de communication et les niveaux d'abstraction utilisés pour sa réalisation. Ensuite, nous mentionnons les principaux constituants d'un réseau local. A la fin du chapitre, nous montrerons l'intérêt que présente l'utilisation des circuits VLSI pour le coût de réalisation d'un réseau local.

1.1 *Principaux constituants d'un réseau de communication*

"Le terme communication peut être défini comme l'opération par laquelle plusieurs entités échangent de l'information; plus précisément , l'une d'entre elles , l'émetteur , transmet à une ou plusieurs autres, les récepteurs , une information appelée message" [COR -81].

Dans cette étude le type de communication qui nous intéresse est la communication dans les systèmes informatiques.

Une communication peut être effectuée entre deux entités ou plus. La plus simple des communications est celle réalisée entre deux entités reliées à travers une ligne directe.

A partir du moment où l'on désire faire communiquer deux entités (ordinateurs, terminaux,...), émergent trois notions importantes qui sont la base de toute communication : voie de communication, interface et protocole.

. La voie de communication est un dispositif qui permet l'interaction entre deux entités qui communiquent entre elles. Les voies peuvent être réalisées sous des formes très diverses: supports métalliques, fibres optiques, ondes radio-électriques.

. L'interface est la partie de la voie accessible à chacune des entités et sur laquelle elles peuvent agir. Un exemple d'interface normalisée est l'avis V-24 du CCITT, ou bien, dans une communication téléphonique, c'est la prise téléphonique.

. Le protocole est l'ensemble des règles d'utilisation convenues par les entités pour réaliser une fonction particulière.

Dans la réalité, pour des raisons économiques (réduction du coût de connexion, augmentation de la disponibilité, partage de ressources), les communications s'effectuent à travers un réseau de communication; par exemple, dans le cas où l'on désire relier un terminal à un ordinateur distant ou faire communiquer plusieurs ordinateurs entre eux à travers un réseau local. La figure 1.1.1 donne un résumé des différents types de liaisons utilisées dans les systèmes informatiques et de leurs performances (distance, débit de

transmission).

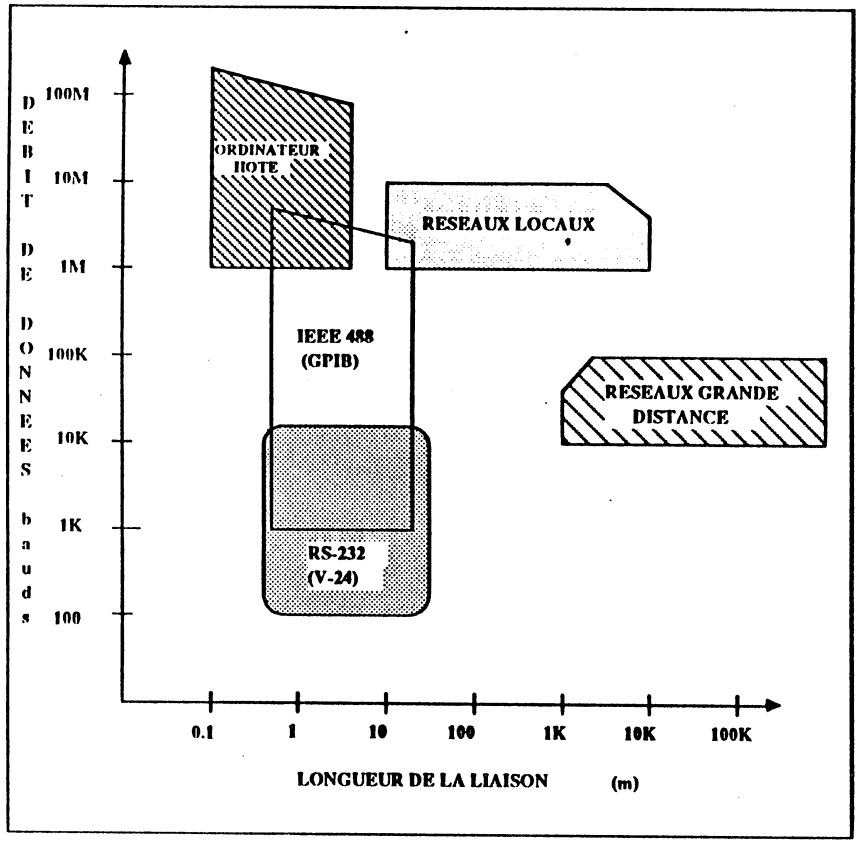


Figure 1.1.1

Par rapport à une communication directe entre deux entités, la communication à travers un réseau de communication pose les problèmes suivants:

. la désignation du correspondant,

. le partage des ressources d'un système entre plusieurs communications. Par exemple, comment sera partagée la voie de communication lorsque deux ou plus parmi les correspondants désirent "parler" en même temps ?,

. le type de réseau de communication à utiliser. Comment seront transmises les informations ?.

La réalisation de ces communications pose différents problèmes difficiles à appréhender sans faire des abstractions. Ainsi est-il possible de décomposer le réseau en plusieurs niveaux de communication.

1.1.1 Niveaux de communication

Un réseau peut être découpé en différents niveaux d'abstraction qui permettent de mieux structurer les échanges d'informations dans le réseau en question [DAY-83].

Chaque niveau réalise des services pour le niveau qui lui est supérieur et utilise les services offerts par le niveau inférieur. Les services offerts par un niveau sont réalisés à l'aide des fonctions et des services du niveau inférieur.

Pour transmettre des objets d'un niveau à l'autre, une interface est nécessaire.

Chaque niveau est composé d'entités. La coopération entre les entités du même niveau est définie par un protocole dans ce niveau.

Plusieurs avantages sont offerts par ce type de conception en niveaux:

. le service fourni par le niveau est défini indépendamment de la manière dont il est effectué,

. des changements peuvent donc être faits dans un niveau du moment que le service demandé par le niveau supérieur est toujours assuré.

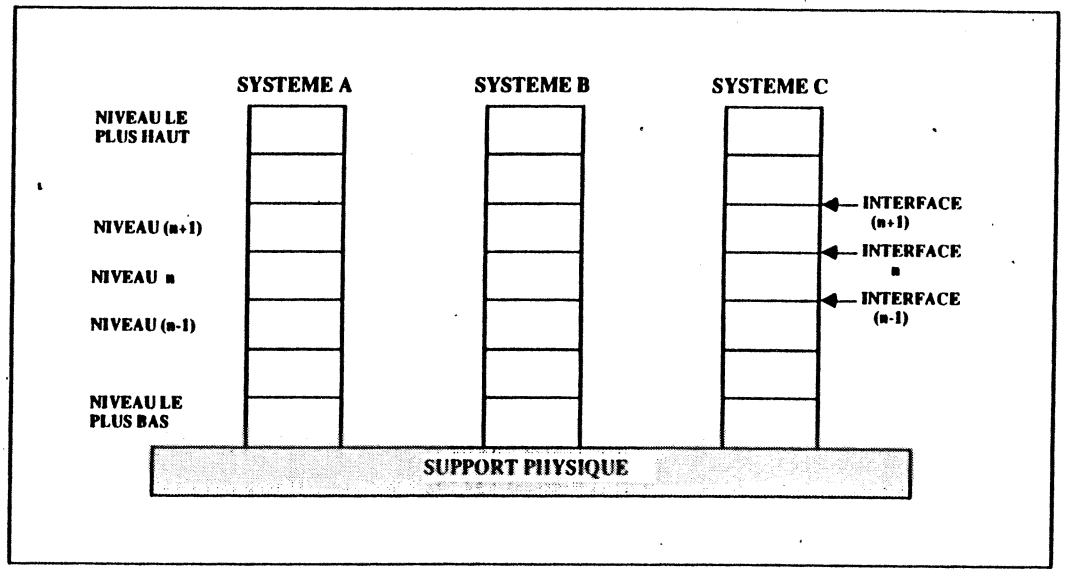


Figure 1.1.2 Niveaux d'abstraction et interfaces

Ainsi par exemple, le modèle de réference OSI (figure 1.1.3) est-il divisé en 7 niveaux. Les trois niveaux inférieurs (physique, liaison et transport) sont les niveaux du réseau utilisés pour transmettre les messages. Les trois niveaux supérieurs, cependant, reflètent les caractéristiques des systèmes communicants, leurs

fonctions sont établies sans prendre en compte le medium physique utilisé. Les utilisateurs (ou systèmes) concernés par la communication ne font appel qu'aux fonctions des niveaux session, présentation et application. Le niveau transport agit comme la liaison entre les systèmes et le réseau.

Il faut signaler que dans cette thèse, notre étude va se limiter aux deux premiers niveaux de communication du modèle OSI [OSI-79] : le niveau physique et le niveau liaison, surtout dans l'optique des réseaux locaux.

Le niveau physique

Le niveau physique détermine les normes : mécaniques, électriques et fonctionnelles pour l'accès à la voie de communication physique ou medium.

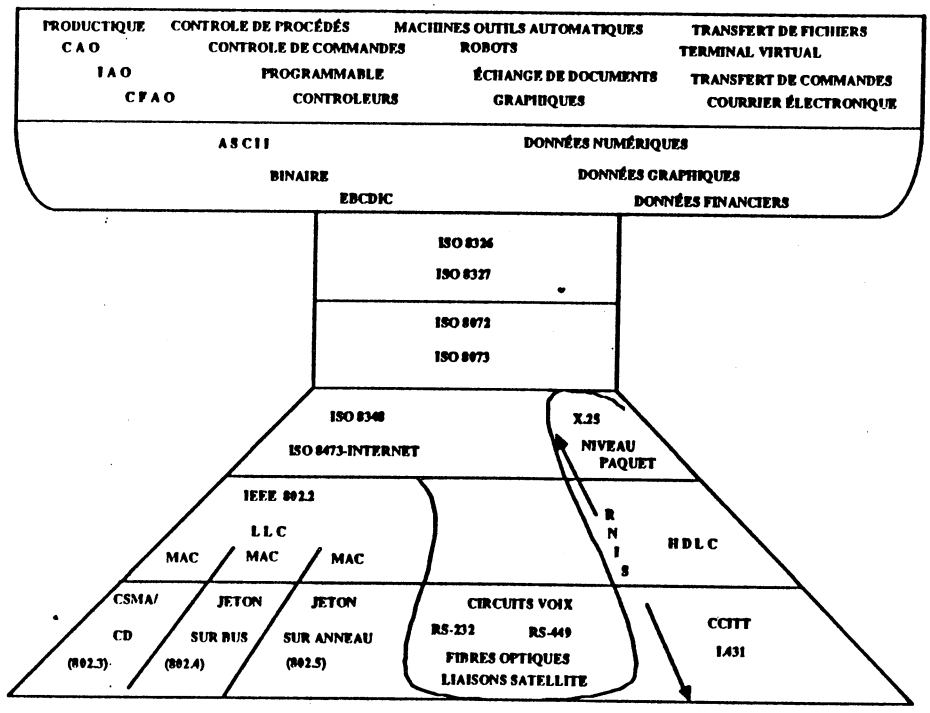


Figure 1.1.3 Modèle OSI et protocoles associés [VOE-86].

Le niveau liaison

Le but du niveau liaison est de fournir les moyens fonctionnels et les protocoles nécessaires pour les transferts de données entre les entités du réseau et détecter les erreurs possibles qui peuvent arriver sur le niveau physique.

Les protocoles de données typiques sont les protocoles type HDLC pour une connexion point à point dans un réseau à longue distance. Pour les réseaux locaux les standards les plus connus sont définis par le comité IEEE 802.

1.2 *Principaux aspects de la réalisation d'un réseau local*

Parmi les critères importants à prendre en compte dans la réalisation de tout réseau local, se trouvent le problème de la transmission, le problème des protocoles performants pour réaliser les échanges des informations et celui du faible coût de réalisation [CLA-78].

Aussi, la réalisation d'un réseau local est-elle fortement liée à son application. Par exemple : la sûreté de fonctionnement ou les problèmes de temps réel ne sont pas aussi importants dans un réseau bureautique que dans un réseau industriel.

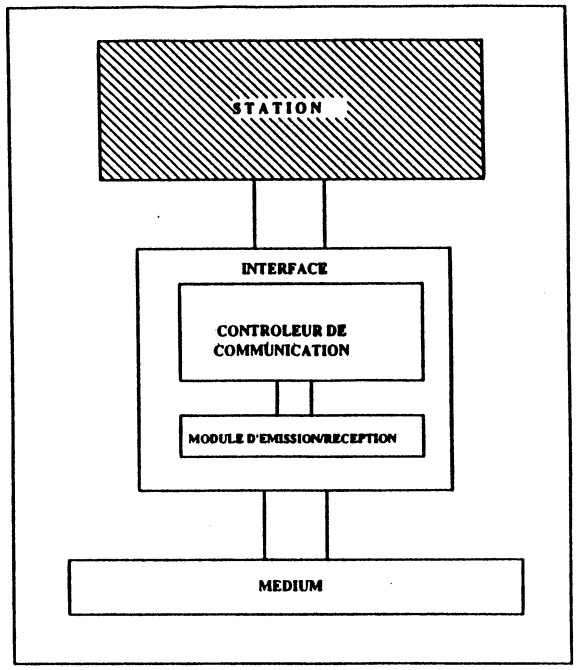


Figure 1.1.4 Réseau local (medium, interface, station)

S'il est vrai que le type d'application définit les performances d'un réseau, il faut aussi signaler que tous les éléments qui le constituent sont du point de vue fonctionnel les mêmes. Ces éléments sont les suivants :

- la voie de communication,

. le mécanisme d'accès au medium,

. l'interface qui permet la connexion entre la station et le medium de communication,

. les protocoles de communication chargés du contrôle des échanges d'information. Ils sont répartis en plusieurs niveaux : du niveau liaison au niveau application.

Avant de continuer, il faut indiquer que cette thèse se limite surtout à l'étude des circuits contrôleurs de communication, de par le rôle important qu'ils jouent dans la réalisation du réseau. Ils déterminent, dans une grande mesure, les performances du réseau. Par ailleurs, une forte réduction dans les coûts de connexion au réseau peut être obtenue via leur réalisation matérielle.

Nous allons donc mentionner ci-dessous les principales caractéristiques des éléments cités quelques lignes plus haut:

* La voie de communication

En général, la voie ou medium de communication doit être simple, fiable, bon marché, et physiquement robuste. Elle doit permettre la transmission à grande vitesse sans être sensible aux perturbations extérieures.

Les principales caractéristiques d'un réseau local liées au médium de communication sont [MAC-79] :

. type de transmission: en bande de base ou par modulation. La transmission en bande de base est surtout utilisée pour des raisons économiques : facilité de mise en oeuvre et bonne adaptation aux débits de transmission désirés,

. type de codage: en bande de base les plus utilisés sont : Manchester biphase, NRZ (non retour à zéro) et NRZI (non retour à zéro inversé) et en modulation : la modulation en fréquence (FSK).

. supports physiques : les fils métalliques, les câbles coaxiaux, les fibres optiques et les ondes-radioélectriques. Il faut signaler que le prix de revient du support physique de transmission joue aussi un rôle important dans les coûts de réalisation du réseau. Dans le projet MAP, conduit par General Motors, c'est le câble coaxial pour télévision (CATV) qui a été choisi, pour son faible prix et la quantité de câbles déjà installés. Mentionnons que c'est un réseau en large bande avec un protocole d'accès de type jeton sur bus (IEEE 802.4 [JBU-85]).

. topologie du réseau: étoile, bus ou anneau, sont les configurations les plus utilisées,

. type de connexion: en série ou en parallèle. Pour des raisons techniques et économiques la connexion série est la plus utilisée.

. utilisation de la voie: selon la topologie du réseau la voie peut être utilisée en point-à-point (relie directement deux et seulement deux stations communicantes) et voie partagée ou multipoint (permet de relier plus de deux stations communicantes). L'utilisation d'une voie partagée soulève deux problèmes :

. l'allocation de la voie, puisqu'une seule communication peut se dérouler à la fois,

. l'adressage des éléments connectés pour permettre à l'émetteur d'un message d'identifier le destinataire.

* Méthodes d'allocation

Plusieurs méthodes d'allocation de la voie de communication existent. Parmi les méthodes dynamiques d'allocation, nous pouvons citer:

. la méthode par compétition (contention), elle a besoin d'un algorithme pour traiter les collisions, par exemple le CSMA/CD [SHO-80] bien connu,

. la méthode par élection, elle peut se faire :

. par sélection. Par exemple un algorithme de ce type a été proposé par [ESW-81] et utilisé dans [DIA-82] [DAN-83] pour la réalisation d'un réseau local à bus parallèle d'une dizaine de mètres,

. par consultation (polling). Une variante de cette méthode sera utilisée dans le circuit communicateur FIP-VLSI qui sera présenté dans la dernière partie de ce travail.

Enfin pour terminer, il reste à signaler que les méthodes d'allocation les plus fréquemment utilisées sont celles définies par le comité 802 d'IEEE. Elles sont décrites dans les projets de norme : IEEE 802.3 CSMA/CD [CSM-85], IEEE 803.4 jeton sur bus et IEEE 802.5 jeton sur anneau [JAN-84]. Ces documents incluent les spécifications de la couche physique et les caractéristiques du medium.

* Méthodes d'adressage

L'autre problème présenté par l'utilisation des voies partagées ou multipoint est la manière de définir la station réceptrice.

D'une manière générale au niveau de la liaison, on donne à chaque station une adresse qui l'identifie. On peut aussi définir une adresse commune pour un groupe de stations (sous-diffusion) ou une adresse globale pour toutes les stations pour des besoins de diffusion de l'information. Plusieurs adresses peuvent être éventuellement reconnues par une seule station, ce qui correspond alors à un adressage logique et non plus seulement physique.

* Le circuit contrôleur de communication

Un élément important dans la construction d'un réseau local est le circuit contrôleur de communication. Cette interface généralement réalisée au niveau matériel permet de relier d'un côté la station hôte et de l'autre côté la circuiterie nécessaire pour la transmission de l'information sur le medium de communication (voir figure 1.1.4).

Une manière plus formelle de déterminer les frontières de cette interface et les fonctions qu'elle réalise est de faire référence à la norme **IEEE 802.**

La norme **IEEE 802** resulte d'un effort de normalisation pour les réseaux locaux des deux premiers niveaux, le niveau physique et le niveau liaison, du modèle **ISO.** Ces deux couches ont été sous-divisées de la manière suivante:

Le **LLC** est l'interface commune avec les couches supposées logicielles plus hautes. Toutes les méthodes d'accès pour le **LLC** autorisent la station à faire les transferts de données sans connaître exactement la méthode d'accès utilisée (définie par le niveau **MAC**).

Dans la couche physique, le **MAU** sert à connecter la station au medium de communication et la sous couche **PS** fourni les services nécessaires pour dérouler le protocole d'accès sur le réseau. La méthode exacte d'interconnexion est transparente pour le **MAC.** Lorsqu'un nouveau medium, tel que fibres optiques, sera disponible, de nouvelles normes pour la couche physique pourront être définies pour utiliser les méthodes d'accès déjà éprouvées.

La figure 1.1.5 (b) montre les fonctions que l'interface doit réaliser, lesquelles sont implantées par matériel.

La partie "matériel" est la clé pour obtenir de bonnes performances à un prix bas. Le grand progrès atteint par l'industrie des semiconducteurs a rendu possible dans un premier temps, la réalisation de ces interfaces à l'aide de circuits MSI et LSI. S'il est vrai que cette solution a permis la connexion aux réseaux locaux du point de vue technologique, les coûts de connexion sont restés trop élevés pour que les réseaux puissent atteindre les dimensions et le marché attendus. Ainsi une solution à ce problème de coût est-elle la possibilité d'intégrer ces interfaces dans un ou plusieurs circuits VLSI. Dans 1.3 nous citons les avantages d'une telle intégration, comme dans le cas des circuits réalisés pour le réseau Ethernet [INT-83].

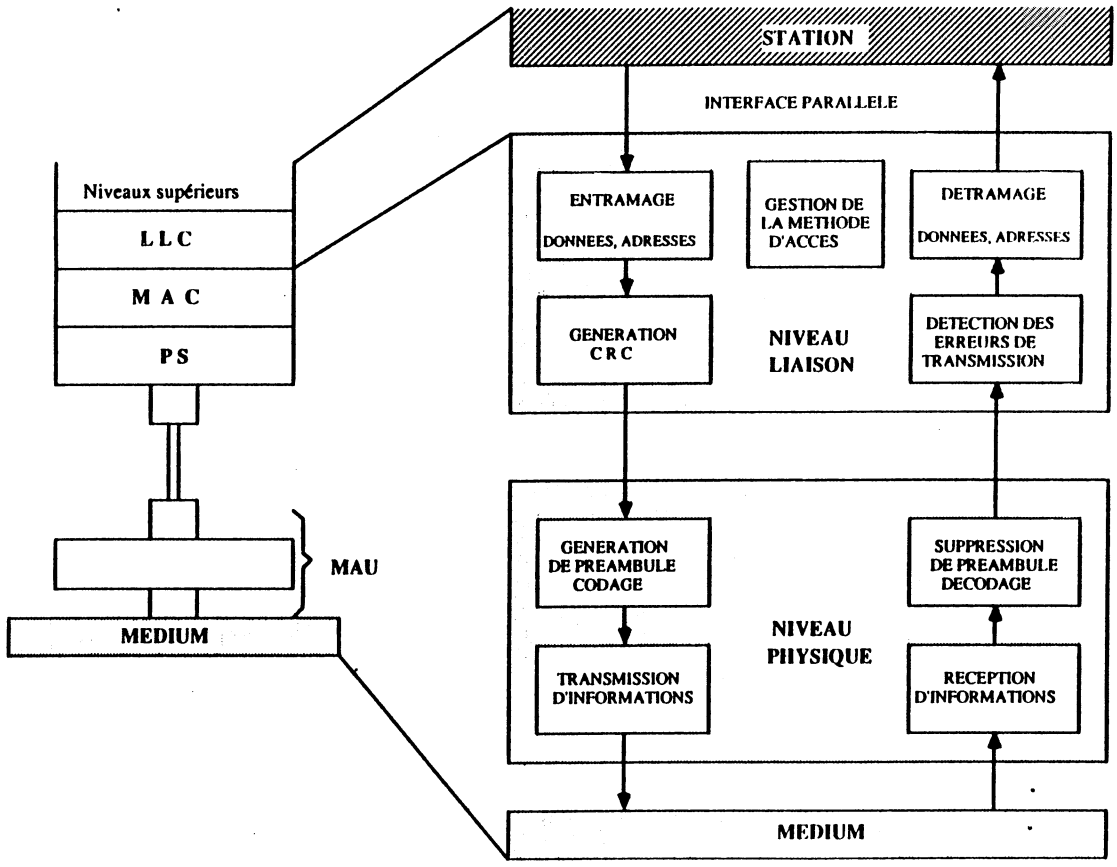


Figure 1.1.5 IEEE 802

* Les protocoles

Chaque étape d'une communication est réglée par un ou plusieurs protocoles particuliers qui permettent aux entités de se mettre d'accord sur les décisions à prendre et de se placer dans un état où la communication est possible. Ainsi par exemple, les éléments utilisés dans les trois premiers niveaux du modèle OSI sont ils:

. le bit ou le caractère pour les protocoles de gestion des ligne,
. les trames pour les protocoles du niveau liaison,
. les messages pour les protocoles de transport au niveau du réseau, ...

1.3 *L'intéret d'utiliser les VLSI dans les circuits d'interface*

Une des raisons importantes pour lesquelles les applications informatiques distribuées ont tant évolué, est la grande évolution technologique des circuits intégrés. Elle a provoqué une augmentation dans la puissance de traitement et une baisse de prix des composants, donc une réduction de prix dans les équipements (voir annexe A). Ci-dessous, nous allons citer divers avantages apportés par les circuits VLSI aux réseaux locaux, ainsi que les obstacles rencontrés pour aboutir à une réelle baisse de prix. Mais avant tout, l'utilisation des circuits VLSI représente un intérêt économique très important dans la réalisation d'un réseau local.

* Les principaux avantages

Le premier avantage des circuits VLSI est la possibilité de faire en sorte que le coût de la connexion puisse être plus faible que le prix de l'élément le moins cher connecté au réseau.

Le deuxième avantage est la possibilité d'augmenter les performances du réseau, d'une part par l'intégration d'un plus grande nombre de fonctions dans le même circuit et d'autre part grâce aux possibilités offertes par la technologie (augmentation de la vitesse de transmission).

Les avances obtenues dans le développement des outils automatiques pour la conception de circuits intégrés, offre la possibilité de réaliser des circuits VLSI pour des applications bien précises. Ceci permet de réduire l'encombrement des cartes utilisées dans la réalisation des connexions au réseau. Ceci amène comme conséquence une réduction dans les coûts de connexion, en plus de la réduction déjà obtenue dans la fabrication.

La sûreté de fonctionnement peut être augmentée. Ainsi que la facilité d'entretien.

Les contraintes propres aux solutions matérielles, par exemple manque de souplesse, peuvent être résolues par la conception des circuits spécifiques ou par des circuits microprogrammés. Un exemple de ce dernier type de circuits est un des 5 circuits réalisés par Texas Instruments [LIN-85] pour le réseau en anneau d'IBM (IEEE 802.5).

* Les inconvénients

Il faut que le circuit VLSI soit réalisé à un grand nombre d'exemplaires. Pour cela, d'une part il faut que le marché existe et d'autre part que les produits sur le marché respectent les normes internationales établies.

Un problème rencontré dans la réalisation d'un circuit VLSI est son temps de conception. Ce temps peut varier de plusieurs mois à plusieurs années, selon la complexité du circuit. Donc, cette étape de conception représente un facteur très important dans son prix de revient.

2. DIFFERENTES METHODES DE CONCEPTION DE CIRCUITS INTEGRES VLSI

2 DIFFERENTES METHODES DE CONCEPTION DE CIRCUITS INTEGRES VLSI

2.1 *Méthodes d'implantation de circuits intégrés*

Les trois méthodes de conception de circuits intégrés les plus répandues sont: les circuits réalisés "à la main" ("full custom"), les réseaux prédiffusés ("gate array") et les circuits à base de cellules précaractérisées ("standard cell"). Ces méthodes sont utilisées pour distinguer deux classes de circuits : les circuits à la demande ("custom") et les circuits personnalisables ("semi-custom").

Cette classification permet de se rendre compte des différences qui existent entre ces circuits. Les coûts de réalisation et les performances de chacun d'eux sont directement liés au degré de pré-traitement effectué.

Depuis un certain temps les circuits intégrés étaient conçu entièrement à la main, mais avec l'arrivée des systèmes de CAO, plusieurs étapes de la conception sont maintenant effectuées automatiquement.

Concevoir un circuit à la demande revient à intégrer les fonctions demandées sur un minimum de surface, le coût d'un circuit étant fortement fonction de celle-là.

La méthode de conception la plus répandue, assurant la meilleure utilisation du silicium, reste la méthode dite "au micron". Selon cette méthode les concepteurs dessinent tous les masques, dans les moindres détails. Lorsqu'il le peut, le concepteur utilise l'implantation symbolique qui consiste à représenter chaque transistor ou chaque porte, suivant le cas, par une figure géométrique (un symbole) relativement simple. L'implantation du circuit se ramène alors au placement des symboles et à leur interconnexion. Un programme informatique génère ensuite le dessin de tous les masques. Cette méthode est nettement plus rapide que la précédente et permet même à une personne ne connaissant pas parfaitement la technologie de concevoir un circuit à haute intégration. Le seul inconvénient de la méthode, par rapport à la précédente, est une certaine perte de place dans le silicium [BOU-85].

Il est courant que le cycle conception-fabrication de ce type de circuits comporte une à trois reprises selon les erreurs détectées.

Ces problèmes font que la réalisation d'un circuit à la demande devient très coûteuse et sa disponibilité prend beaucoup de temps. Elle varie bien entendu en fonction de la complexité du circuit.

* Les avancées de la technologie

Une conséquence importante du développement rapide de la technologie des semiconducteurs dans les dernières années est l'augmentation de la complexité des circuits VLSI; ce qui a obligé le développement de plusieurs approches de conception.

Les raisons principales de l'augmentation de cette complexité sont les suivantes:

.le développement de technologies microniques et sub-microniques (pour les années à venir) ont permis de réduire les dimensions des transistors. Ceci a donné par conséquent la possibilité d'en intégrer un plus grand nombre et donc de réaliser des circuits de plus en plus complexes. Une mesure possible de la complexité d'un circuit est donnée par le nombre de motifs qui doivent être dessinés; un nombre qui a plus que doublé tous les deux ans et il n'y a pas signe de ralentissement de cette croissance, si bien que les limites des procédés (photolithographie) les plus couramment utilisés aujourd'hui dans la fabrication des circuits intégrés seront bientôt atteintes. De nouvelles techniques sont abordées pour augmenter les possibilités d'integration, comme par exemple:

. la photorépétition sur tranche,
. l'écriture directe sur tranche,
. les techniques de circuits sur trois dimensions,

.l'augmentation du nombre de masques utilisés dans le processus de fabrication.

Cette augmentation de la complexité impose de fortes contraintes qu'il faut respecter :

.les dimensions du circuit. Evidemment, lorsque les dimensions de la puce augmentent, le nombre de dispositifs qui peuvent être placés sur la puce augmente aussi. Cependant, plus grande est la puce, plus la probabilité pour qu'un défaut de fabrication survienne est élevée; ce qui conduit à une diminution du rendement de fabrication,

. le nombre de motifs à intégrer. Dans la mesure où le nombre de dispositifs à intégrer augmente , le temps et les coûts de conception augmentent aussi, ainsi que les erreurs de conception,

. les règles de dessin et le nombre de masques à dessiner imposés par la technologie de fabrication.

Cette complexité entraîne celle des différentes étapes nécessaires pour la conception d'un circuit à haute densité d'intégration. Donc, seules l'adoption d'une méthodologie de conception systématisée et structurée, et l'utilisation d'outils automatiques peuvent permettre d'espérer que le système réalisé fonctionnera correctement sans avoir besoin de reprendre de nombreuses fois tout ou partie de la conception.

2.2 *Méthodologies de conception*

Il existe trois méthodes suggérées pour maîtriser la complexité des circuits intégrés: la décomposition hiérarchique du circuit à concevoir, l'utilisation de structures régulières et l'implantation automatique [MEA-80]. Ces méthodes ne sont pas exclusives les unes des autres mais représentent en fait le sens des efforts ou des recherches à entreprendre. Parmi ces trois méthodologies de conception la méthode hiérarchique est la plus importante.

2.2.1 La démarche hiérarchique

Le problème posé par la complexité de conception d'un circuit intégré peut être décomposable en plusieurs niveaux d'abstraction (figure 1.2.1) bien que leurs frontières et leurs noms ne soient pas bien établis.

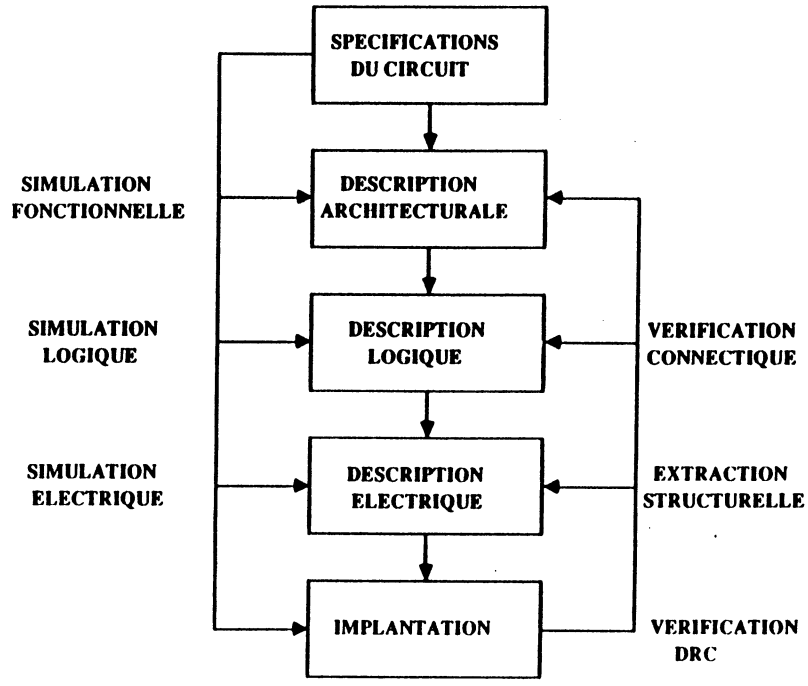


Figure 1.2.1 Niveaux d'abstraction

Le processus de conception d'un circuit peut être interprété comme la traduction des spécifications fonctionnelles du circuit, à travers plusieurs étapes de raffinement des spécifications jusqu'à l'implantation finale du circuit.

Avantages de l'abstraction

Les avantages donnés par l'abstraction, mentionnés au § 1.1.1, permettent de réduire la complexité par la diminution des données à manipuler à un moment donné. Cette réduction est réalisée par la décomposition du circuit en une série de blocs fonctionnels plus petits. La division successive des blocs produit un dessin hiérarchique descendant. Cette phase de décomposition descendante du dessin est réalisée jusqu'à ce que les éléments résultants soient suffisamment simples pour être facilement implantés (figure 1.2.2).

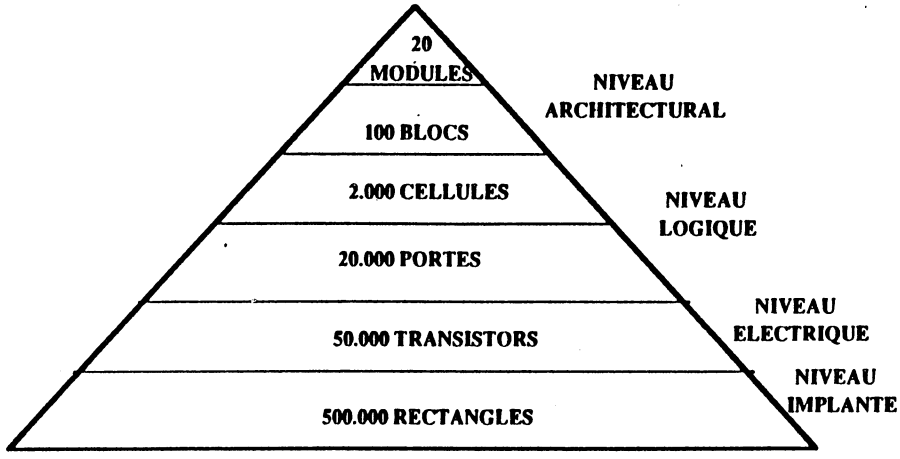


Figure 1.2.2

* **La démarche hiérarchique descendante**

Dans chaque niveau d'abstraction il est possible de réaliser certains traitements qui permettent de vérifier la conformité du circuit avant de passer au niveau inférieur. Cette démarche descendante est réalisée à l'aide de différents simulateurs fonctionnels, logiques, électriques,... Ainsi, chaque niveau doit-il remplir les fonctions qui lui sont propres en plus des contraintes imposées par les niveaux supérieurs. Par exemple:

. le fonctionnement du circuit, au niveau de la description logique, doit être conforme aux spécifications fonctionnelles avant de passer à la conception au niveau électrique,

. le dessin de masques du circuit ne sera pas effectué, avant que les contraintes électriques ne soient résolues.

* La démarche hiérarchique ascendante

Chaque élément est donc implanté indépendamment et assemblé avec les autres en respectant la même hiérarchie dans un processus de construction ascendante, mais en même temps il devra respecter à son tour une série de contraintes imposées, par exemple:

.le respect des règles de dessin, définies par la technologie utilisée,

.la connectique, pour respecter le comportement du circuit,

.la surface occupée du circuit, laquelle doit être la plus faible possible,

.la consommation autorisée,

.la vitesse de fonctionnement,...

2.2.2 La régularité

La deuxième démarche à suivre pour maîtriser la complexité est la recherche de régularité dans les motifs. La répétition des structures permet un gain de temps dans la conception et un moindre effort dans le dessin.

Tous les circuits VLSI courants exploitent la régularité des motifs. Les exemples les plus évidents sont les mémoires; mais les processeurs peuvent aussi avoir des structures régulières par l'utilisation d'une construction en tranches ("bit-slice") dans l'implantation de leur chemin de données [JOH-79] [SUZ-81].

Généralement dans les microprocesseurs une autre structure régulière très caractéristique est utilisée pour l'implantation des parties contrôle : les réseaux logiques programmables PLAs ("Programmable Logic Array") plutôt que la logique aléatoire.

Enfin, un niveau de régularité plus élevé sera proposé dans les nouvelles générations de machines : par exemple les réseaux systoliques [KUN-82] [QUI-83].

2.2.3 L'automatisation des différentes étapes de la conception

Une troisième considération à prendre en compte est l'automatisation. Les méthodes automatiques ou semi-automatiques d'implantation les plus couramment utilisées sont: l'implantation par cellules précaractérisées, par réseaux prédiffusés, par réseaux logiques programmables, ... Par ailleurs des outils de compilation de silicium peuvent directement générer, pour certains circuits, le dessin géométrique des masques à partir des spécifications fonctionnelles exprimées dans un langage de haut niveau.

L'utilisation d'outils d'implantation automatique permet à des ingénieurs de disciplines différentes de concevoir des circuits, mais aussi réduit le temps et les erreurs de conception.

Mais, l'automatisation du dessin présente encore des inconvénients car les circuits résultants occupent une surface plus grande, ont une consommation plus importante et sont aussi plus lents que les circuits dessinés à la "main". A noter que l'arrivée des nouvelles technologies, le développement des outils d'aide à la conception plus sophistiqués et l'évolution des différentes méthodes de conception ont permis de combler dans une certaine mesure ces inconvénients.

Les outils automatiques (par exemple, les générateurs automatiques de PLAs ou les logiciels d'assemblage de cellules élémentaires pour la construction d'opérateurs flexibles (RAMs, ROMs, chemin de données, ...)[ROU-86]) peuvent etre facilement intégrés dans un style de conception hiérarchique où le concepteur peut choisir la meilleure méthode d'implantation manuelle ou automatique, adaptée à l'étape en cours de conception.

L'utilisation de méthodes manuelles incorpore aussi une certaine automatisation. Par exemple la méthode symbolique de dessin MDMOS [LEB-80] [CON-82], garantit que l'implantation sera sans violation des règles de dessin.

2.2.4 Les compilateurs de silicium

S'il est vrai qu'il existe aujourd'hui des systèmes appelés "compilateurs de silicium", le développement d'un vrai compilateur de silicium est à venir.

Un compilateur de silicium "idéal" est un système de conception de circuits intégrés qui génère directement à partir d'une description fonctionnelle en langage de haut niveau, une implantation géométrique du circuit prête à être utilisée comme entrée du masqueur. Une condition importante est que ces systèmes doivent être capables

d'implanter toute classe de circuits.

Les compilateurs de silicium existants sont uniquement capables de réaliser l'implantation sur le silicium d'une classe spécifique de circuits: microprocesseurs, circuits de traitement de signal, et autres architectures bien connues [DEN-82] [SOU-83] [LOU-83] [CVT-86].

* Le fonctionnement des "compilateurs de silicium" existants

Bien qu'il n'existe pas un compilateur de silicium idéal, les compilateurs de silicium existants essaient de suivre les différentes étapes illustrées dans la figure 1.2.3.

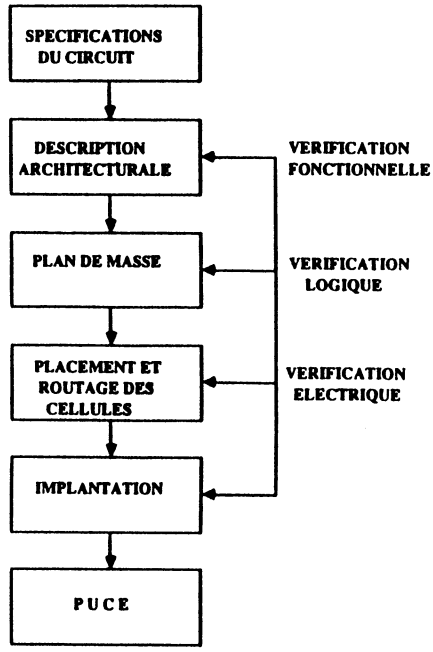


Figure 1.2.3 Etapes suivies par un compilateur de silicium

La première étape consiste, à partir des spécifications du circuit, à définir l'architecture du circuit à implanter; le schéma en termes de blocs comme les blocs fonctionnels tient compte de la vitesse, de la surface et d'autres considérations. Pour réaliser cette étape, plusieurs approches existent [GAJ-83]:

. Description fonctionnelle.-plusieurs compilateurs permettent la définition des spécifications fonctionnelles du circuit à partir d'un langage de haut niveau comme dans le cas de MacPitts qui utilise Lisp ou un langage formel dans le cas de First (un graphe de flux de données) ou une description graphique à l'aide d'opérateurs prédéfinis dans le cas d'IMHOTEP [REY-85]. Un inconvénient de cette solution est que quelques compilateurs seulement permettent au concepteur de spécifier, ou au moins forcer cette architecture. L'architecture la plus commune a été

bien entendu optimisée pour produire des microprocesseurs.

. Description structurelle.- certains algorithmes nécessitent des architectures spéciales, non considérées dans le compilateur. Ainsi existe-t-il des compilateurs où le choix de l'architecture peut être fait par un concepteur sachant que l'implantation sera laissée au compilateur; cette méthode est utilisée par Silicon Compiler Inc..

. Système expert.- Cette approche se situe en amont de l'approche précédente. Elle permet de générer plusieurs architectures à partir d'une description formelle des fonctions du circuit. C'est au concepteur d'en choisir une parmi celles qui lui sont proposées. Un exemple de cette approche est le système "Design Automation Assistance" (DAA) développé à l'université de Carnegie-Mellon [THO-83]. Il prend la description fonctionnelle d'un circuit et conçoit une description structurelle à l'aide d'une approche de système expert.

La deuxième étape concerne l'implantation et l'interconnexion des niveaux bas des blocs fonctionnels.

Les blocs construits sont généralement des éléments tels que: additionneur, registre, multiplexeur, ... tous disponibles pour le compilateur d'une façon paramétrable. Le paramètre le plus évident est le nombre de bits qui spécifie le nombre de fois qu'une tranche formée d'un bit, par exemple d'un additionneur, sera répétée pour effectuer une fonction sur un mot de donnée entier. Les autres paramètres qui peuvent être optimisés par le compilateur de silicium sont la vitesse et la consommation.

La troisième étape concerne l'implantation de chacune des cellules paramétrées qui est générée à partir d'un réseau de transistors de base pour produire le résultat correct ou par l'utilisation de cellules précaractérisées. Idéalement, les implantations produites par ces procédures sont considérées comme correctes par construction.

2.3 *Conclusions*

Les buts de la recherche de méthodes et le développement d'outils pour l'implantation automatique de circuits sont de produire des outils qui puissent implanter une fonction quelconque rapidement et l'implanter aussi bien qu'un concepteur expérimenté puisse le faire et avoir le circuit disponible le plus vite possible. Tendant vers ces objectifs, beaucoup de travaux ont été déjà effectués pour avoir de meilleures techniques de placement et routage, principalement pour les réseaux prédiffusés, qui ont l'avantage d'un faible coût de fabrication. D'autres travaux ont été développés pour les systèmes de cellules précaractérisées et les cellules paramétrables telles que

celles utilisées dans Bristle Blocks (cellules déformables). La définition de nouveaux outils de génération automatique de plans de masse permettra l'implantation plus efficace des circuits.

Les circuits VLSI sont de plus en plus influencés par la conception hiérarchique et régulière. L'automatisation sera utilisée dans la mesure du possible; seules les parties les plus critiques seront faites à la main. Dans ce sens, nous allons proposer une méthodologie de conception à base d'outils tels que les générateurs de PLAs et des systèmes d'assemblage de cellules pour générer des macro-fonctions à partir de cellules précaractérisées. Dans l'approche proposée, qui tient de l'approche de description structurelle, il faudra définir auparavant l'architecture à implanter.

PARTIE 2

METHODOLOGIE

1. METHODOLOGIE PROPOSEE

1 METHODOLOGIE PROPOSEE POUR LA CONCEPTION DE CIRCUITS VLSI DE COMMUNICATION

Dans l'optique d'une diminution du temps de conception d'un circuit intégré à la demande et d'une augmentation de la sûreté de conception, nous allons proposer une méthodologie de conception basée sur l'assemblage d'opérateurs déjà prédéfinis et contenus dans une bibliothèque.

Cette méthodologie sera orientée principalement vers les circuits de communication. La méthodologie proposée est basée sur les approches d'une conception hiérarchique; elle prend en compte l'utilisation de structures régulières et l'utilisation d'outils d'implantation automatique.

1.1 *Les outils d'aide à la conception disponibles*

Avant de présenter la méthodologie, il est important de faire quelques remarques sur les outils d'aide à la conception disponibles. Ils jouent un rôle important dans la définition d'une méthodologie de conception [ANC-83].

Ici, nous allons uniquement mentionner les outils disponibles, la présentation sera détaillée dans les chapitres suivants. Parmi les outils disponibles ont peut mentionner :

.CASSIOPEE [BEY-82] [HEN-80].- c'est un système intégré d'aide à la conception qui regroupe dans une base de données unique les représentations des différents niveaux d'abstraction du circuit (implanté, électrique, logique, ..).

.LOF (Langage d'Opérateurs Flexibles) [BER-84].- c'est un assembleur de silicium,

.GASP (Générateur Automatique de Séquenceurs) [FLA-84].- c'est un système qui permet, à partir d'une description fonctionnelle (graphe d'états, équations logiques,...), la génération automatique de fonctions combinatoires, séquenceurs, automates, ..., réalisées à base de PLAs et de ROMs.

Les systèmes LOF et GASP travaillent directement avec les informations contenues dans la base de données de CASSIOPEE.

Il faut signaler qu'un des buts de la méthodologie présentée est la validation de ces outils dans la conception de circuits à base de blocs flexibles ou précaractérisés. Elle servira aussi comme étude préliminaire du système SCHUSS [CNE-85] en développement au CNET-Grenoble.

1.2 *Présentation de la méthodologie*

A la différence des méthodologies de conception présentées dans la première partie de cette thèse pour la conception d'un circuit à la demande, la méthodologie proposée peut être considérée comme intermédiaire entre une conception à base de cellules précaractérisées et une conception dite "de compilation de silicium" partant d'une description structurelle.

Les éléments du circuit à implanter seront constitués de blocs compilés et stockés dans une bibliothèque dite d'opérateurs flexibles (ces blocs seront construits à base de cellules précaractérisées). Par ailleurs, cette méthodologie prévoit l'utilisation d'outils qui permettent de générer des modules à partir d'une description formelle; par exemple, la partie contrôle d'un circuit peut être synthétisée par un seul séquenceur.

Dans cette méthodologie, il sera nécessaire que le concepteur définisse l'architecture du circuit et sa décomposition en blocs fonctionnels.

La figure 2.1.1 illustre les différentes étapes de la méthodologie proposée, elle suit une double démarche : une conception descendante et une construction ascendante. Il faut remarquer que les blocs hachurés correspondent aux étapes qui seront réalisées une seule fois et effectuées lors de la première mise en oeuvre de la méthodologie ou dans le cas d'un changement de technologie, par exemple d'une technologie NMOS à CMOS. Plusieurs critères et la méthode à adopter (dans ce travail) pour l'implantation de la bibliothèque de cellules sont donnés ci-dessous (cf. 1.3).

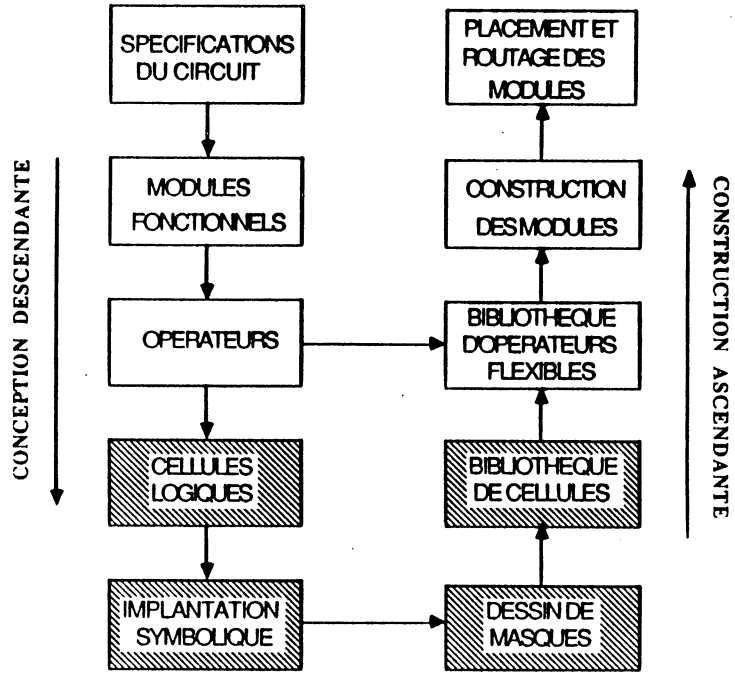


Figure 2.1.1

Cette méthodologie sera d'autant plus valable que bon nombre de circuits seront implantés avec, car ainsi la conception d'un circuit sera limitée uniquement aux niveaux d'abstraction plus élevés. Ceci permettra d'éliminer les étapes plus fastidieuses qui demandent un fort investissement en temps et qui sont source possible d'erreurs.

1.2.1 Approche descendante

* **Spécifications du circuit**

La première étape consiste à définir une architecture cible du circuit à concevoir à partir de l'analyse de leurs spécifications fonctionnelles. Dans cette phase le concepteur doit prendre en compte en premier les solutions qui permettent la mise en oeuvre des outils de génération automatique; par exemple, pour l'implantation de la partie contrôle, utiliser un ou plusieurs PLAs.

* **Modules Fonctionnels**

Découper l'architecture proposée dans divers modules fonctionnels. A partir de ces modules fonctionnels, définir le plan de masse du circuit à implanter (figure 2.1.2). Un module fonctionnel peut être constitué d'un ou plusieurs opérateurs selon les fonctions réalisées.

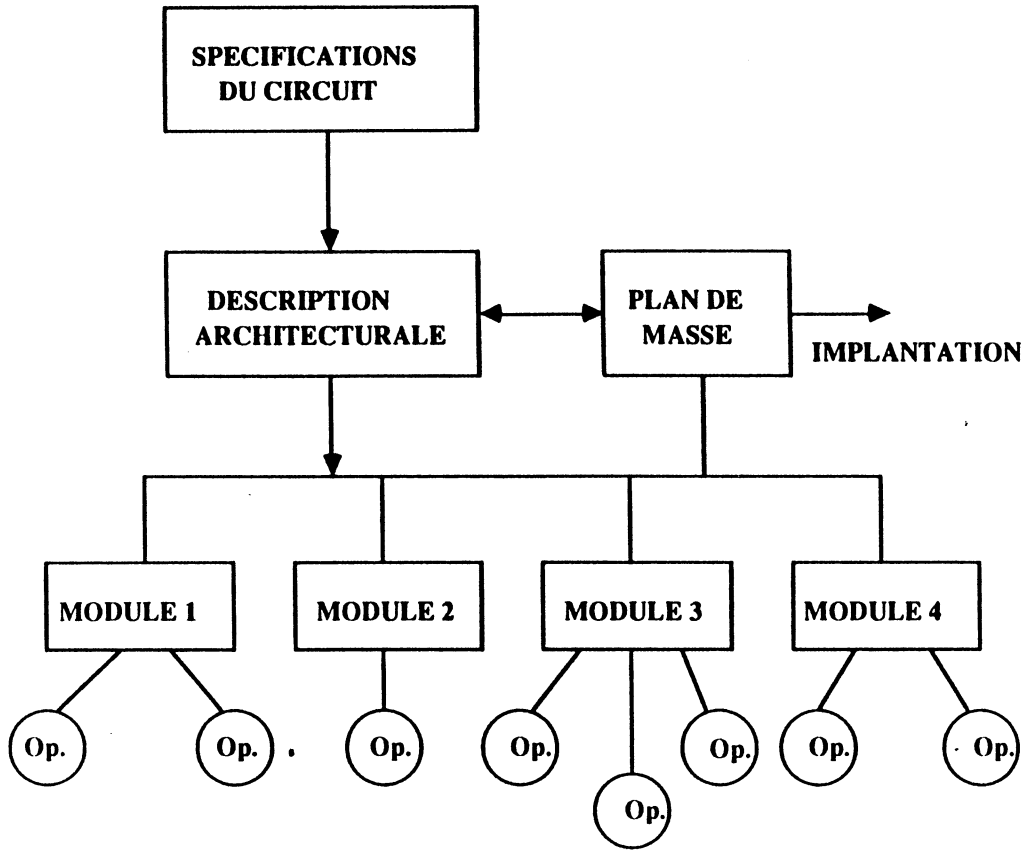


Figure 2.1.2

* Opérateurs

Les opérateurs sont des éléments du circuit qui réalisent des fonctions très spécifiques (registres à décalages, compteurs, mémoires, séquenceurs...).

Une idée fondamentale, à cette étape de la méthodologie, est de pouvoir générer des opérateurs qui puissent être réutilisés en différentes circonstances et dans l'implantation de différents circuits; il faut donc que ces opérateurs soient flexibles et qu'ils puissent être générés automatiquement et taillés en fonction de l'application envisagée.

Un opérateur est défini par sa structure topologique et le type de fonction qu'il réalise. Ils peuvent être classés en opérateurs en tranches et opérateurs programmables.

Les opérateurs en tranches

Ces opérateurs seront construits à partir des cellules élémentaires (portes logiques, bascules, amplificateurs,...), et générés automatiquement par la définition d'une série de paramètres (nombre de bits, consommation, vitesse de fonctionnement, ...).

L'implantation de ces opérateurs en tranches demande la définition d'une structure topologique spécifique pour déterminer comment seront assemblées les cellules pour réaliser la fonction désirée. Dans ce cas l'implantation de structures modulaires et régulières faciliteront la tâche.

Les opérateurs programmables

Ces opérateurs seront générés automatiquement à partir d'une description formelle de la fonction ou module à implanter. Par exemple, un graphe d'état peut décrire le fonctionnement d'un séquenceur ou les équations logiques peuvent définir le comportement d'un PLA.

1.2.2 Construction ascendante

Cette partie de la méthodologie est dédiée à la construction du circuit à partir de chacun des éléments déjà implantés; mais pour réaliser cela une construction hiérarchique sera nécessaire. Ainsi par exemple, la construction d'un opérateur en tranche fera appel à la bibliothèque de cellules; la construction d'un module à son tour aura besoin de chacun des opérateurs qui le constitue, et ainsi de suite jusqu'au dessin complet des masques du circuit.

* **Bibliothèque d'opérateurs flexibles**

Cette bibliothèque sera constituée surtout des opérateurs en tranches, car les opérateurs programmables sont construits directement à partir de leur description fonctionnelle et généralement implantés à base d'une structure PLA ou ROM.

Par contre, la génération automatique et paramétrable des opérateurs en tranches requiert des programmes informatiques permettant le calcul de la position exacte de chacune des cellules qui les constituent, selon la structure topologique correspondante.

Donc, chaque opérateur en tranches sera défini par un programme spécifique. Il permettra de décrire l'assemblage des différentes cellules qui constituent l'opérateur en question.

Le programme exécuté par l'assembleur de silicium permettra de générer le dessin de masques de l'opérateur correspondant selon une série de paramètres définis par le niveau supérieur.

L'ensemble de ces programmes et les descriptions fonctionnelles des opérateurs programmables constituent ce que nous allons appeler : bibliothèque d'opérateurs flexibles.

* **Construction de modules**

Les modules seront construits soit à partir de l'assemblage de plusieurs opérateurs soit par la génération directe du module. Dans ce cas, un module est formé d'un seul opérateur (par exemple, le module de contrôle du circuit qui peut être défini par un simple séquenceur).

* **Placement et routage de modules**

Cette étape constitue la dernière phase de la construction du circuit avant d'obtenir le dessin complet des masques.

Chacun des modules fonctionnels sera placé selon le plan de masse général prédéfini et ensuite un routage manuel ou automatique devra être effectué pour interconnecter les divers modules.

> Remarque: Il faut signaler que chacun des niveaux de description, dans l'approche de conception descendante, devra être validé à l'aide des simulateurs selon le niveau d'abstraction concerné : simulation fonctionnelle, logique, et électrique. La simulation électrique doit être utilisée chaque fois qu'un nouvel opérateur est défini pour garantir son bon fonctionnement et fournir les caractéristiques électriques et dynamiques utilisées par les niveaux supérieurs.

La vérification faite lors de la construction du circuit concerne les règles de dessin DRC ("Design Rulers Checking") sur les nouvelles structures générées.

1.3 *Critères et méthodologie à adopter pour l'implantation de la bibliothèque de cellules*

1.3.1 Critères d'implantation

La bibliothèque de cellules doit être réalisée dans l'optique d'utiliser le moins de cellules possibles et pour le plus grand nombre d'opérateurs.

La technnologie considérée est une technologie N MOS à deux niveaux de connexion: l'aluminum et le silicium polycristallin.

L'implantation de ces cellules doit en faît respecter des règles dans le but de simplifier leur utilisation, d'optimiser les interconnexions entre cellules, d'aider l'implantation en tranche et de faciliter de cette manière la réalisation des différents opérateurs. Plusieurs de ces règles correspondent aux propositions faites dans [SUZ-81] et [REI-83].

Ces règles sont les suivantes:

1.- Il faut respecter pour toutes les cellules une largeur déterminée.

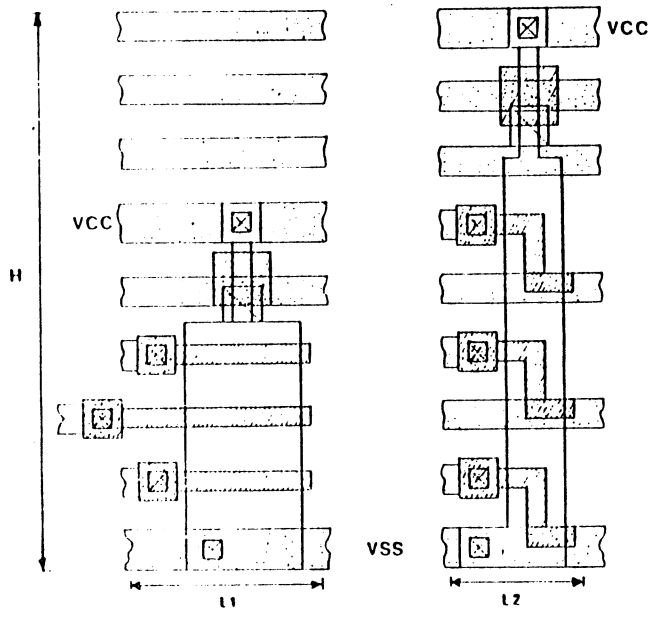


Figure 2.1.3

Il faut signaler qu'une réduction dans la largeur de la cellule provoque son allongement et donc une augmentation de son degré de transparence (i.e. nombre de fils d'interconnexion qui peuvent la traverser). La figure 2.1.3 illustre un exemple de cette considération.

2.- La cellule doit être construite avec des transparences parallèles aux fils d'aluminium (VDD, VSS, horloges et fils de contrôle). Ceci correspond à la structure en bandes définie en [BIA-81] [FIL-81].

Cette structure permet, lors de la construction de l'opérateur, de relier les lignes d'alimentation en formant une structure en peignes imbriqués comme le montre la figure 2.1.4.

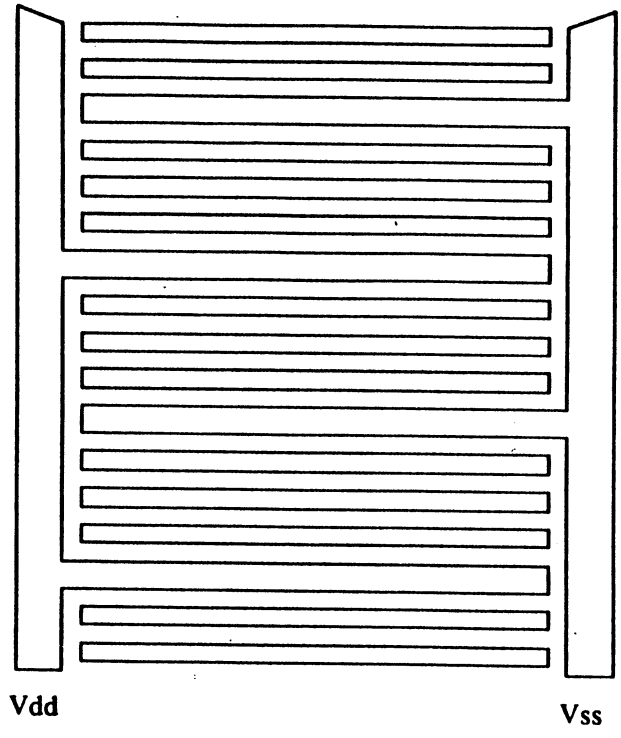


Figure 2.1.4

L'utilisation des transparences permet les interconnexions entre les différents blocs et évite de former des canaux d'interconnexion qui provoquent une augmentation de la surface du circuit.

3.- Il faut essayer de placer les fils de contrôle, d'alimentation ou d'horloges ainsi que les transparences à la même hauteur afin de faciliter l'interconnexion horizontale des cellules.

4.- Les entrées et les sorties de deux cellules adjacentes doivent être du même côté. Ceci oblige à réaliser les cellules avec les entrées du côté inférieur par exemple et les sorties du côté supérieur. Il arrive que la seule juxtaposition de deux cellules permette de les relier.

5.- Les entrées et les sorties de chaque cellule doivent être en silicium-polycristallin (poly).

1.3.2 Implantation

Une fois que le schéma logico-électrique est connu, les cellules seront normalement dessinées à la main et conçues pour s'assembler avec leurs voisines.

* Le dessin des cellules

Plusieurs méthodes existent pour réaliser le dessin des masques, parmi les plus utilisées on peut citer: le "dessin au micron" (la plus classique) et le dessin symbolique bâton (stick diagram).

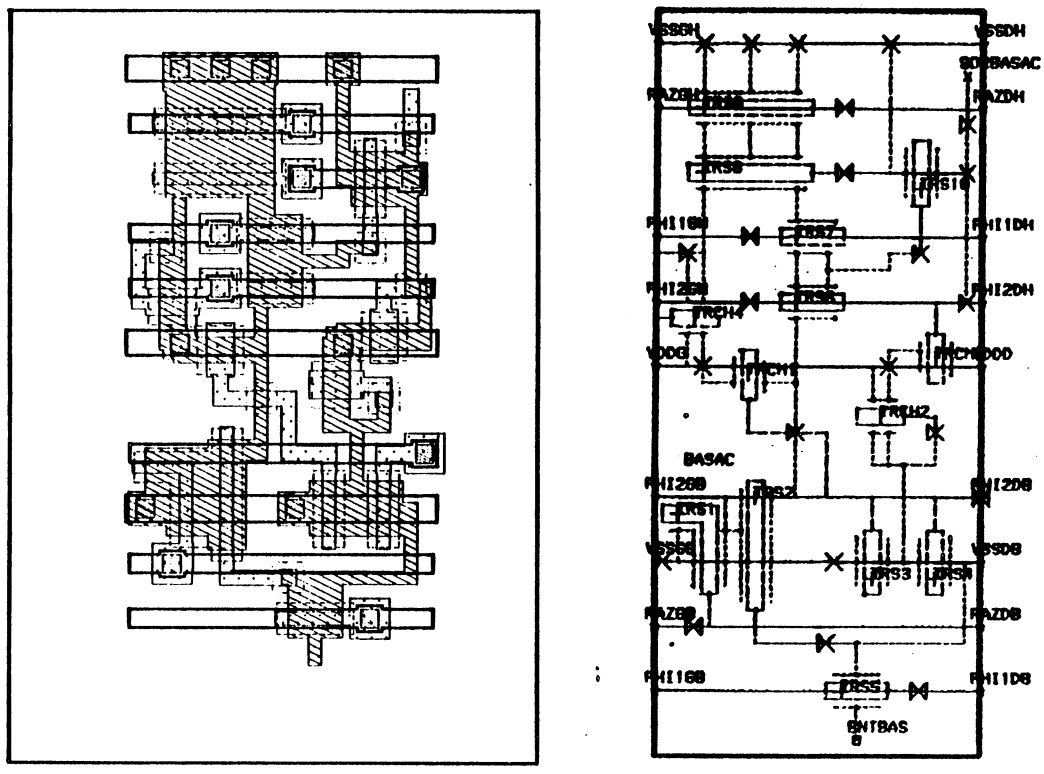


Figure 2.1.5 (a) dessin "au micron", (b) dessin symbolique bâton

. La première méthode est la représentation à base de rectangles et de polygones de chacun des éléments qui forment la cellule (connexions, contacts, transistors). La figure 2.1.5(a) montre un exemple du dessin des masques d'une cellule (bascule maître-esclave).

. Le dessin symbolique bâton permet une représentation squelettisée du dessin rectangle; il peut être réalisé sur grille à pas fixe ou grille à pas variable et tassé par programme. La figure 2.1.5(b) illustre la bascule maître-esclave sous la forme d'un symbolisme bâton sur grille à pas variable.

Dans notre application, nous utiliserons pour le dessin des cellules le symbolique bâton sur grille fixe CRASH [CRA-82] pour NMOS. Ce symbolisme n'utilise pas de mécanismes de compactage mais permet d'obtenir un dessin "au micron" avec une perte de place de 10% par rapport à une solution classique. Il est souhaitable que de futures implantations soient faites "au micron" pour optimiser au maximum la surface occupée.

* Vérification du schéma implanté

L'utilisation du symbolisme bâton demande la connaissance d'un certain nombre de règles technologiques (règles de dessin). Donc, pour garantir la bonne implantation des circuits (dans notre cas les cellules) plusieurs types de vérifications doivent être faites:

. une simulation électrique fine; à partir du dessin de la cellule, elle permet de prendre en compte toutes les données réelles de l'implantation (capacités, résistances, géométrie des transistors). Cette simulation, a posteriori, valide les estimations de départ et permet de caractériser la cellule.

. la vérification des règles de dessin (DRC); la vérification de chacune des cellules est nécessaire pour être sûr que toutes les règles de dessin imposées par la technologie ont bien été respectées. Le respect de ces règles est impératif pour espérer un rendement de fabrication correct.

. vérification de continuité "Electrical Rule Checking" (ERC) et et de cohérence "Layout versus Schematic" (LVS); il est possible d'effectuer une comparaison entre le schéma logique de la cellule et son schéma implanté, ceci peut être fait par programme ou manuellement

Dans la suite de cette partie 2 de la thèse, nous allons étudier un certain nombre de circuits de communication. Ceci aura pour but d'analyser leurs diverses architectures et de pouvoir déterminer les différents modules fonctionnels qui les constituent. Ensuite, nous allons essayer de définir les fonctions réalisées pour

chacun de ces modules et les opérateurs qui leur sont associés.

Après avoir défini les différents opérateurs, nous allons mettre en oeuvre quelques étapes de la méthodologie : classification des opérateurs (en tranches ou programmables) et dans le cas des opérateurs en tranches, la définition de sa structure topologique et la décomposition des cellules qui le constituent.

Avant de terminer la partie 2, les outils de conception utilisés pour la génération des différents opérateurs seront présentés, ainsi qu'un exemple de mise en oeuvre d'une bibliothèque de cellules et d'une bibliothèque d'opérateurs flexibles : opérateurs en tranches plus quelques exemples de la réalisation des opérateurs fonctionnels.

2. ANALYSE FONCTIONNELLE

DES CIRCUITS DE COMMUNICATION

2 ANALYSE FONCTIONNELLE DES CIRCUITS DE COMMUNICATION

Dans ce chapitre sera présentée l'analyse réalisée sur différents circuits de communication d'interface d'entrée/sortie pour déterminer les principaux blocs fonctionnels qui les constituent. L'objectif recherché est la synthèse de ces blocs en termes d'opérateurs qui puissent être répertoriés dans une bibliothèque et générés plus tard automatiquement pour la réalisation de futurs circuits VLSI de communication.

Cette analyse sera orientée principalement vers les circuits d'interface d'entrée/sortie et vue comme une extension des circuits communicateurs ou coupleurs utilisés dans les couches basses du modèle OSI concernant les réseaux locaux.

2.1 *Les circuits d'interface d'entrées/sorties*

Ces circuits permettent la communication, en l'occurrence, entre un système à microprocesseur et le monde extérieur (périphériques).

Plusieurs types de circuits permettent de réaliser cette interface. Ce sont:

. les circuits d'interface parallèle,

. les circuits d'interface série.

Dans les circuits série, il faut considérer les circuits série asynchrone, série synchrone et les circuits multiprotocoles.

Avant de commencer,il est prudent d'avertir le lecteur que l'analyse faite ci-dessous est une analyse fonctionnelle d'un groupe de circuits; il ne s'agit pas d'analyser tous les circuits d'entrées/sorties existants car les mêmes fonctions peuvent être réalisées dans un seul circuit ou dans plusieurs selon le constructeur. La seule considération faite dans le choix des circuits analysés a été de suivre une certaine continuité et évolution dans la complexité du protocole de communication à implanter, ce qui est étroitement lié à la complexité du circuit.

2.1.1 Circuits d'interface parallèle

Ces circuits permettent l'interface entre un système à microprocesseur et les périphériques à travers une liaison parallèle.

fonctions de communication

Les principales fonctions réalisées par ces circuits, (vues du côté microprocesseur) sont les suivantes:

. en réception, la mémorisation des informations en provenance d'un périphérique et le maintien de celles-ci d'une manière stable, jusqu'à ce que le microprocesseur demande cette information.

. en émission, la mémorisation des informations qui viennent du microprocesseur et leur conservation aussi longtemps que le périphérique en a besoin.

. en outre, il y a besoin d'un mécanisme de sélection et d'une commande de lecture/écriture. Le schéma de la figure 2.2.1 montre ces fonctions.

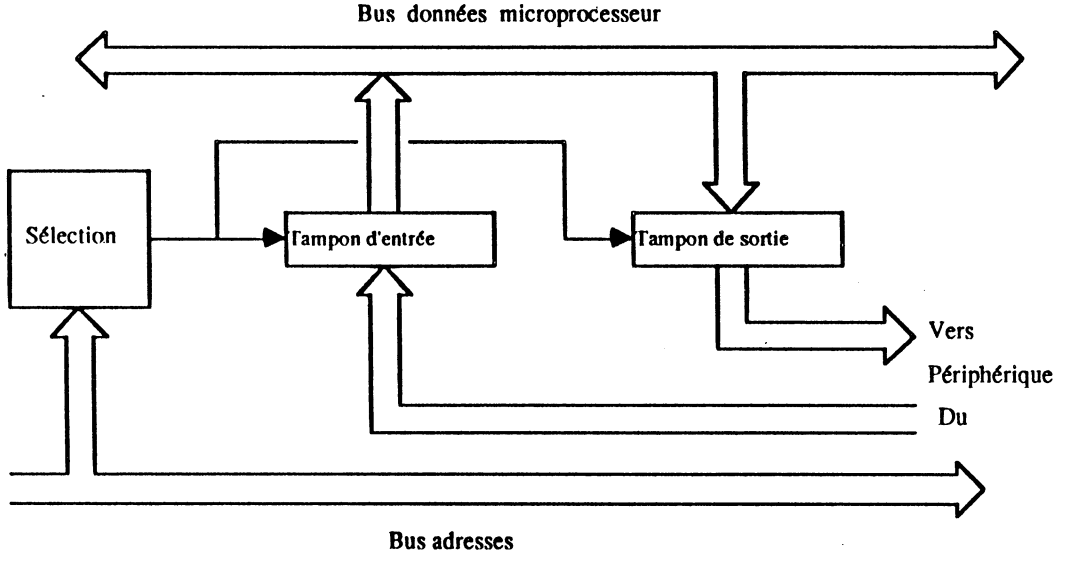


figure 2.2.1

L'arrivée des composants LSI "d'entrées/sorties programmables" (PIO) est venue enrichir les fonctions des circuits d'entrées/sorties en ajoutant de nouvelles possibilités. Ces fonctions sont réalisées par programmation du circuit.

Plusieurs circuits de ce type existent sur le marché et ils sont normalement liés à une famille de microprocesseurs. Parmi les plus répandus, on peut citer; le **MC6821 PIA** ("Peripheral Interface Adapter"), le **8255 PPI** ("Programmable Peripheral Interface") et le **Z80 PIO** ("Parallel input output").

Le PIA est un circuit d'entrées/sorties parallèles qui fournit le moyen d'interfacer des périphériques avec les microprocesseurs de la famille Motorola **MC68xx**.

Le PPI a été conçu pour etre utilisé avec les différentes familles de processeurs 8 bits d'INTEL le 8080, le 8085, ...

Le Z80 PIO, comme les deux circuits précédents, est un circuit d'interface parallèle dédié principalement aux systèmes à microprocesseur Z80.

Le Z80 PIO est compatible dans certains modes de fonctionnement avec le PIA et avec le PPI. Son organisation interne et externe rappelle sur certains points plus le PIA que le PPI.

Ces circuits réalisent une interface parallèle entre les dispositifs périphériques (clavier, lecteur de bande perforée, contrôleur de CRT, contrôleur de lecture de disquettes, imprimantes, ...) et le bus du microprocesseur en question. Ces trois circuits sont fabriqués en technologie MOS canal N et montés chacun dans un boîtier 40 broches.

* Architecture externe

Les trois circuits présentent une architecture externe constituée de deux groupes de broches, un groupe lié au bus local du microprocesseur et l'autre groupe aux périphériques, figure 2.2.2.

Le groupe de lignes connecté au bus local varie selon le constructeur. Mais globalement, on peut dire que ces lignes peuvent être classées de la manière suivante:

. Un bus de données bidirectionnel 8 bits qui permet les échanges de données entre le microprocesseur et le PIO,

. un groupe de lignes de contrôle en provenance du microprocesseur pour la sélection du boîtier, le contrôle de la lecture/écriture des données et des commandes, la sélection des registres internes et de validation.

. une ligne d'initialisation du circuit. Cette ligne n'est pas explicitement définie dans le cas du Z80 PIO, mais la fonction d'initialisation est réalisée par la combinaison de certaines lignes de contrôle,

. plusieurs lignes de demande d'interruption. Dans le cas du PIA chaque port du circuit est associée à une ligne d'interruption, la sortie de chaque ligne est en drain ouvert pour permettre un OU câblé entre toutes les lignes d'interruption.

fonctions de communication

Dans le cas du PPI, les interruptions ne sont pas faites d'une manière câblée sinon par la lecture d'un registre d'état.

Une caractéristique particulière du Z80 PIO est la façon dont il gère les interruptions. Tous les échanges d'informations avec le microprocesseur se font sous interruption. Le Z80 PIO a toute la logique nécessaire pour réaliser la gestion d'une structure hiérarchisée d'interruptions. Il donne la possibilité d'interrompre le microprocesseur sur occurrence d'une configuration spécifique dans les périphériques.

Ces différents aspects dans la gestion et génération des interruptions font du Z80 PIO un circuit plus complexe que le PIA et que le PPI. Ces circuits donnent au microprocesseur une plus grande souplesse pour le traitement des interruptions venant des dispositifs périphériques et permettent aussi d'augmenter la capacité de traitement.

Le groupe de lignes côté périphériques est constitué, dans le cas du PIA et du Z80 PIO de deux ports bidirectionnels 8 bits et quatre lignes de contrôle (deux lignes associées par port). Ces lignes de contrôle permettent d'effectuer le protocole de communication ("handshake", impulsionnel, ou programme).

Quant au PPI, il est formé de trois ports 8 bits bidirectionnels A, B et C.

La figure 2.2.2 illustre en termes des modules fonctionnels l'architecture interne des circuits d'interface analysés.

* Architecture interne

L'analyse des architectures internes des circuits PIA, PPI et PIO, a permis de dégager les principaux blocs fonctionnels qui les constituent. Ces blocs sont une interface avec le microprocesseur, un bloc de contrôle et les ports.

L'interface avec le microprocesseur est constituée par les deux modules suivants:

. le module d'interface de données. Il est construit par des amplificateurs trois états bidirectionnels qui permettent le transfert des données entre les circuits d'interface et le microprocesseur.

. Un module de sélection constitué par un décodeur, génère les commandes de lecture/écriture pour effectuer les échanges d'information entre le microprocesseur et le "PIO". Ce module génère aussi les commandes qui permettent la programmation des différents registres (de contrôle, de données et d'état) qui

constituent les modules de contrôle des ports.

. Des modules de contrôle logique des ports. Chacun des modules est associé à un port du circuit. Ces modules constitués normalement par des registres à lecture seule ou à écriture seule, permettent de définir d'une part le mode de fonctionnement (configuration et protocole de communication) de leur port associé et d'autre part le contrôle des interruptions.

. Les ports permettent le dialogue avec les périphériques. Ils sont constitués par des registres de mémorisation, accessibles par le microprocesseur pour l'écriture des données en sortie (émission) et pour la lecture des données en entrée (réception). Les sorties de ces registres sont normalement commandées par des amplificateurs trois états.

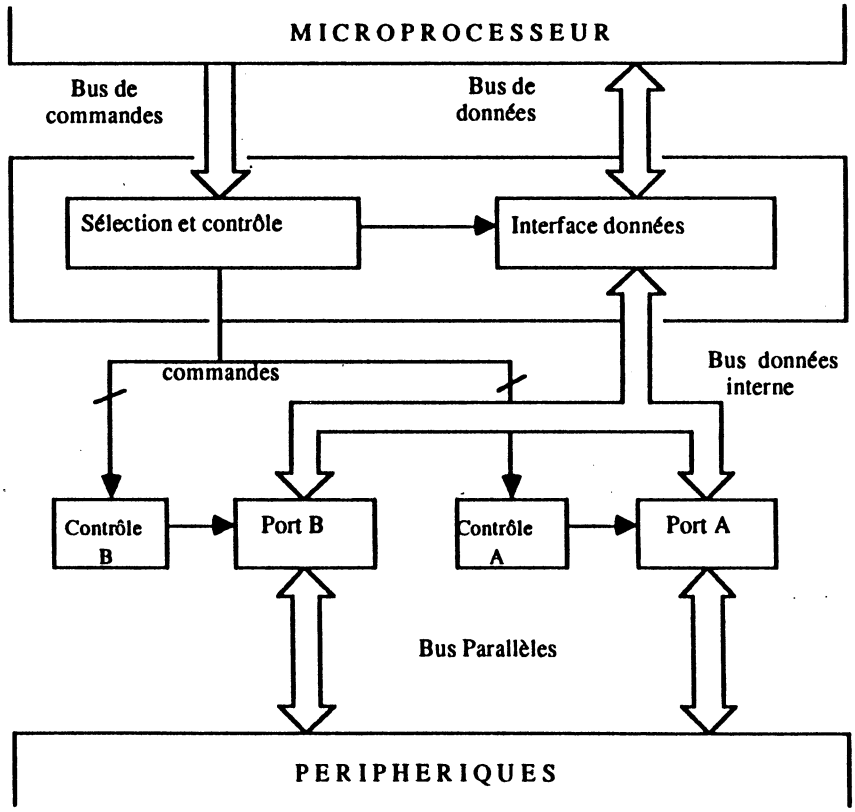


figure 2.2.2 Architecture interne d'un PIO

Dans le cas du **PIA** et du **Z80 PIO**, tous les bits de chacun des ports peuvent être programmés individuellement en entrée ou en sortie.

Les trois ports du **PPI** sont constitués de 24 lignes d'entrées/sorties. Elles peuvent être programmées dans deux groupes de 8 bits (ports **A** et **B**) et deux groupes de 4 lignes qui forment le port C. Les ports peuvent opérer en trois modes différents; en entrée/sortie, entrée/sortie avec "handshake" (exclusif pour les ports

A et B), et le dernier mode correspond à une transmission bidirectionnelle avec "handshake" exclusif du port A. Ces modes de fonctionnement sont aussi proposés par le Z80 PIO. Dans ces modules, les lignes du port C sont utilisées pour réaliser les commandes de "handshake" et les bits du port C qui ne sont pas utilisées comme lignes de contrôle seront utilisées comme lignes d'entrée/sortie.

2.1.2 Les circuits d'interface série

Divers circuits intégrés monolithiques LSI ont été commercialisés, par différents constructeurs de circuits microprocesseur et utilisés comme circuits d'interface spécialisés pour liaisons série synchrone et asynchrone.

Une dizaine de ces circuits, de constructeurs différents, ont été analysés. Un résumé des résultats obtenus est donné dans les tableaux I, II, III et IV de l'annexe B.

Parmi ces circuits, on a choisi pour cette présentation trois circuits de communication compatibles avec la famille de microprocesseurs MC68xx de Motorola. Le choix de ce constructeur a été fait d'une part car ses composants sont parmi les plus connus et répandus sur le marché et d'autre part pour l'implantation séparée des protocoles: synchrone (orienté octet et orienté bit) et asynchrone "START-STOP".

Les caractéristiques générales des circuits analysés sont les suivantes:

1.-L'ACIA MC6850 "Asynchronous Communications Interface Adapter". Il fournit un moyen de communication pour réaliser l'interface entre un système à microprocesseur et des périphériques à accès série asynchrone. L'ACIA est un circuit programmable permettant la communication série asynchrone selon la procédure "START-STOP".

2.- Le MC6852 SSDA "Synchronous Serial Data Adapter" est un circuit programmable qui permet l'émission et la réception en série avec synchronisation des données parallèles présentes sur le bus de données du système à microprocesseur. Le circuit permet l'insertion/suppression des caractères de contrôle, le calcul de parité sur les données, l'émission/réception de blocs d'information (suite de caractères).

3.- Le MC6854 ADLC "Advanced Data Link Controller", est un circuit contrôleur de communication série qui fournit les fonctions requises pour l'implantation des protocoles de haut niveau: ADCCP "Advanced Data Communication Control Procedure", HDLC "High Level Data Link Control", et SDLC "Synchronous Data Link Control".

L'ADLC permet différents modes de communication: point à point ou dans une configuration en boucle. Dans ces deux modes les données peuvent être codées soit en code NRZ "Non Retour à Zéro" ou en code NRZI "Non Retour à Zéro Inversé". En outre, le circuit offre la possibilité de travailler en mode auto-test (test-miroir); dans ce mode un bouclage interne se produit entre la sortie de l'émetteur et l'entrée du récepteur, pour vérifier le bon fonctionnement du circuit. L'ADLC est considéré comme un circuit multiprotocoles.

Ces trois circuits sont fabriqués en technologie MOS canal N et livrés dans des boîtiers 24, 24, et 28 broches respectivement.

* Architecture externe

De la même manière que dans les circuits parallèles, l'architecture externe des circuits série est constituée de 2 groupes de broches. Un groupe qui forme l'interface côté bus microprocesseur et l'autre groupe connecté directement à un périphérique ou à travers un modem.

- Interface côté bus local. Les lignes qui constituent cette interface sont pratiquement les mêmes que celles déjà définies dans le cas des circuits d'interface parallèle; un bus bidirectionnel 8 bits; un groupe de lignes pour la sélection du boîtier, le contrôle de la lecture/écriture des données, la sélection des registres de contrôle et les registres d'état.....

L'ADLC a la particularité de fournir les signaux de contrôle pour un circuit d'accès direct à mémoire (DMA).

Côté périphérique, les interfaces des trois circuits sont semblables. Elles sont constituées de lignes pour la transmission, la réception, les entrées d'horloges externes pour contrôler la vitesse d'émission et de réception, de lignes de contrôle d'entrée et de sortie selon les normes RS-232C(V-24) et RS-422, pour l'interface avec un modem.

D'autres lignes spéciales sont incluses, ainsi par exemple le SSDA fournit-il un signal pour indiquer la détection du (ou des) caractère(s) de synchronisation. Dans le cas de l'ADLC, il fournit les signaux nécessaires pour travailler soit comme une station primaire, soit comme station secondaire dans une configuration en boucle.

* Architecture interne

Les architectures internes des trois circuits sont similaires. Elles peuvent être représentées, figure (2.2.3) par quatre blocs fonctionnels: un bloc d'interface avec le microprocesseur, un bloc d'émission, un bloc de réception, et les blocs de contrôle. Il faut signaler que la complexité des 3 derniers blocs est déterminée par la complexité du protocole réalisé.

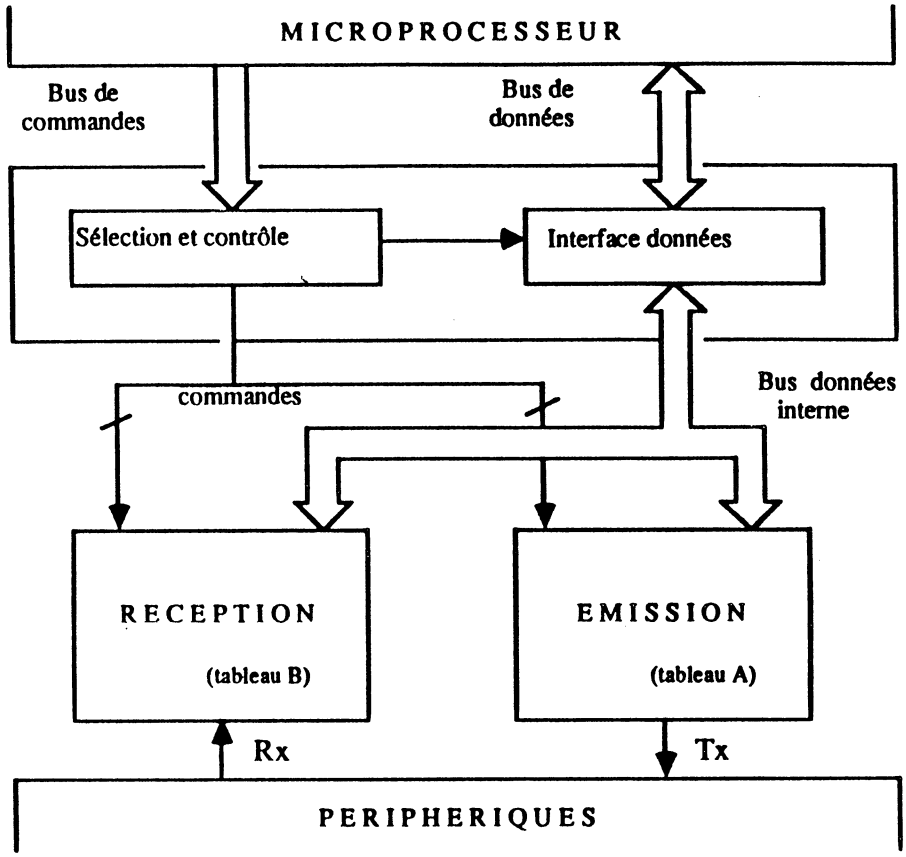


figure 2.2.3 Blocs fonctionnels des circuits de communication série.

Interface microprocesseur.- ce bloc est constitué de deux modules:

. une interface de données bidirectionnelle 8 bits, pour les échanges des informations,

. un module de sélection et de contrôle, qui assure la sélection du boîtier, la sélection des registres internes (de contrôle et d'état) et permet de contrôler la lecture/écriture de données provenant du bus local. Il permet aussi le contrôle des demandes d'interruption.

Ce bloc d'interface a les mêmes caractéristiques que celles définies dans le cas des circuits d'interface parallèle.

Bloc de transmission.- ce bloc réalise, globalement et selon le circuit, les fonctions suivantes:

. l'entramage de données (et d'adresses),

. le contrôle d'erreur sur les données,

. la génération des délimiteurs (de caractère ou de trame),

. la génération de la transparence des données,

. le codage des données, dans le cas de l'ADLC.

A partir de ces fonctions, les modules qui permettent leur réalisation sont résumés dans le tableau A suivant pour chacun des circuits analysés.

FONCTIONS REALISEES EN EMISSION

FONCTION	ACIA	SSDA	ADLC
Génération délimiteurs	START-STOP	caractère de SYNCHRO	fanion, annulation, repos
Contrôle d'erreur	calcul parité	génération CRC	génération CRC
Transparence	-	insertion caractère DLE	insertion de zéros
Entramage	stockage (reg. tampon) sérialisation des données	stockage (FIFO) sérialisation des données multiplexage	stockage (FIFO) sérialisation des données multiplexage

tableau A

Bloc de réception.- il réalise, selon le circuit en question, les fonctions suivantes;

. le détramage de données, et la reconnaissance d'adresses,

. la vérification de la parité ou du champ de contrôle d'erreur sur les données,

. la détection des délimiteurs (de caractère ou de trame),

. le traitement de la transparence sur les données,

. le décodage des données, dans le cas de l'ADLC.

Le tableau B ci-dessous résume les modules réalisant les fonctions de réception de données dans chacun des circuits présentés.

Il faut signaler que dans les trois circuits les blocs de transmission et de réception sont associés à une mémoire tampon d'un seul registre (dans le cas de l'ACIA) ou une mémoire tampon FIFO de 3 octets qui permettent de stocker les mots jusqu'à leur transmission ou jusqu'à ce que le microprocesseur effectue la lecture.

Un circuit générateur d'horloge programmable est aussi associé à chacun de ces blocs pour la génération interne de la fréquence de transmission et de réception. Ces fréquences peuvent être aussi fournies de l'extérieur du circuit.

Le bloc de contrôle. Il est évident que le bloc d'émission comme le bloc de réception sont associés aux circuits de contrôle qui permettent: le contrôle du déroulement de la transmission et de la réception des données, la gestion des signaux de contrôle pour les circuits MODEM et pour les DMA's liés à la réception ou émission de données, et le contrôle des interruptions.

FONCTIONS REALISEES EN RECEPTION

FONCTION	ACIA	SSDA	ADLC
Détection des délimiteurs	logique de synchronisation	dét: caractère de SYNCHRO	dét: fanion, annulation, repos
Vér. contrôle d'erreur	vérification calcul parité	vérification CRC	vérification CRC
Suppression Transparence	-	suppression caractère DLE	suppression de zéros
Détramage	stockage (reg. tampon) désérialisation des données	stockage (FIFO) désérialisation des données	stockage (FIFO) désérialisation des données
Identification adresse	-	comparateur	comparateur

Tableau B

Le fonctionnement de ces blocs de contrôle peut être programmé à travers différents registres de contrôle (à écriture seule). Parmi les divers paramètres de contrôle, on peut citer:

. ACIA: longueur du caractère, parité, nombre de bits d'"arrêt", fréquence de transmission.....

. SSDA: motif de synchronisation (un ou deux caractères), type de synchronisation (interne ou externe), ...

. ADLC: l'adresse du destinataire, type de protocole, mode de fonctionnement (point à point ou en boucle)....

Il est possible aussi de fournir le microprocesseur de l'état du déroulement de la transmission à travers des mots d'état contenus dans des registres à lecture seule (registres d'état). Parmi ces informations, on peut toujours citer: erreur de parité ou erreur de CRC, bourrage, sur-cadencement, demande d'interruption ...

2.2 *Conclusions*

L'analyse des circuits ACIA, SSDA, et ADLC nous a permis de déterminer d'une part plusieurs blocs fonctionnels et d'autre part un certain nombre de modules et de fonctions caractéristiques de ces circuits de communication série. Cette analyse a permis aussi de constater que la complexité des circuits augmente en fonction de la complexité du protocole, mais toujours en gardant la même architecture de base.

Comme il a déjà été dit, cette étude n'a pas été limitée aux seuls circuits présentés. Le tableau I (voire annexe B) montre les caractéristiques générales de tous les circuits étudiés et les tableaux II, III et IV montrent les fonctions réalisées selon les différents protocoles de communication série.

Ces tableaux donnent une idée du degré de complexité de chacun des circuits selon les fonctions réalisées. Cette complexité a été étroitement liée à l'évolution de la technologie, qui a permis:

. d'une part d'augmenter la densité d'intégration et par conséquente de pouvoir intégrer un plus grand nombre de fonctions dans un même composant,

. d'autre part d'augmenter la vitesse de transmission.

Ceci a donné comme résultat la naissance des circuits VLSI de commmunication. Par exemple, les circuits Z8030, Z8530 de Zilog [ZIL-84] et le SNC 68562 de Signetics [MAG-85] [ROL-85]. Les circuits Z8030 et le Z8530 ont été conçus, le premier pour travailler avec les microprocesseurs à bus multiplexés et le second destiné aux microprocesseurs à bus séparés. Quant au SNC 68562, il a été conçu pour travailler avec la famille de microprocesseurs 68000.

L'arrivée de ce type de circuits VLSI a permis l'intégration, dans un seul circuit (au lieu des trois circuits Motorola présentés), des fonctions nécessaires pour implanter entièrement les protocoles:

fonctions de communication

. asynchrone: "START-STOP"

. synchrone

. orienté caractère: MONOSYNC, BISYNC,
. orienté octet: DDCMP,
. orienté bit: HDLC, SDLC, ADCCP.

En plus des fonctions intégrées dans les circuits analysés, ces circuits réalisent, entre autres, les fonctions supplémentaires suivantes:

. deux canaux de communication,

. un générateur de baud programmable par canal,

. un oscillateur à asservissement de phase (PLL digitale) par canal,

. le calcul et la vérification des caractères de contrôle de bloc (BCC)

. les codeurs/décodeurs FM0,FM1, Manchester,

. les circuits de contrôle d'interruptions programmable,.....

Enfin, ces circuits VLSI intégrent la circuiterie pour toutes les fonctions requises dans les systèmes avancés de communication de données.

2.3 *Les circuits communicateurs*

Les autres types de circuits de communication à considérer sont les circuits intégrés, dits communicateurs, utilisés dans les couches 1 (physique) et 2 (liaison) du modèle OSI. Le standard IEEE 802, concernant les réseaux locaux, propose le découpage de ces deux couches selon la figure 2.2.4.

Ce découpage a été suivi par les constructeurs de circuits intégrés pour implanter les niveaux MAC, PS et MAU dans plusieurs composants.

Selon les standards : IEEE 802.3 - CSMA/CD, IEEE 802.4 - Jeton sur bus et IEEE 802.5 - Jeton sur anneau, divers composants existent déjà sur le marché ou sont en train d'être développés.

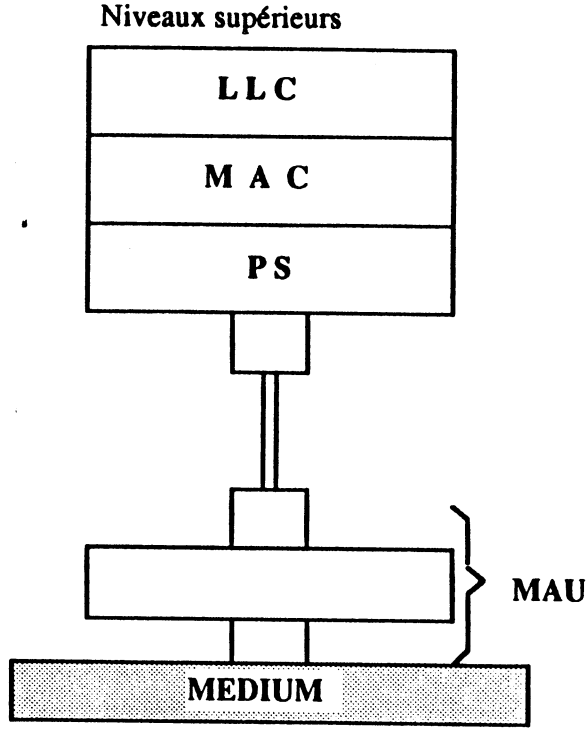


Figure 2.2.4 IEEE 802

Dans [DAN-85] ont été présentés, d'une manière générale, les circuits existants, et une synthèse des différentes fonctions réalisées par de tels circuits intégrés au niveau MAC et PS.

Il faut signaler qu'ici, nous allons nous limiter à faire uniquement certaines constatations par rapport aux circuits déjà présentés car l'analyse plus approfondie des circuits communicateurs est et sera l'objet d'études ultérieures au sein de notre équipe de recherche (PPCI-LGI).

Parmi les constatations faites, nous allons citer les suivantes:

1.- On peut dire que les circuits intégrés de communication série dits multiprotocoles effectuent un sous-ensemble de fonctions du niveau MAC, i.e. ils réalisent les fonctions :

. d'entramage/détramage,

. d'adressage,

. de contrôle d'erreurs,

mais ils n'effectuent pas la gestion de la méthode d'accès propre aux systèmes de communication répartis à accès décentralisé. Celle-ci constitue une différence fondamentale entre les deux types de circuits.

2.- Le débit de transmission constitue un critère de partage entre les deux classes de circuits analysés.

Les circuits communicateurs, par le haut débit de transmission de données requis dans certains réseaux locaux (par exemple; 10 Mbits/s pour ETHERNET, ou 4,5 Mbits/s dans le cas du réseau en anneau d'IBM) , ont besoin d'une interface complexe côté station.

Les constructeurs proposent des circuits communicateurs intégrant plusieurs fonctions qui auparavant se réalisaient par des circuits intégrés externes. Parmi ces circuits, on peut citer les contrôleurs de DMA, la gestion du bus local, la gestion des tampons alloués en mémoire externe, ...

Dans le cas des circuits série analysés (cf 2.1.2), ces fonctions ne sont pas intégrées dans le circuit, si bien qu'il existe une certaine tendance à intégrer tout ce qui est possible pour libérer au maximum le CPU des tâches d'entrée/sortie d'information.

3.- Une troisième différence est sur les fonctions de gestion des erreurs et des statistiques propres aux méthodes d'accès utilisées dans les réseaux (nombre de collisions, fréquence de retransmissions,...).

4.- Enfin pour terminer, il faut signaler que les circuits communicateurs doivent aussi réaliser différentes fonctions nécessaires pour vérifier le bon fonctionnement du réseau: des diagnostics (vérification logique du CRC, test de réflectométrie, vidage, ...) et les tests déjà rencontrés dans les circuits série: bouclages interne et externe, autotest.

3. LES OPERATEURS FLEXIBLES

3 LES OPERATEURS FLEXIBLES

3.1 Définition des opérateurs flexibles

L'étude des différents circuits de communication présentés dans le paragraphe précédent et la définition d'un certain nombre de fonctions caractéristiques réalisées par ce type de circuits, nous ont amenés à proposer une bibliothèque d'opérateurs flexibles. L'objectif recherché, comme il a été déjà dit dans 2.1, est de pouvoir utiliser cette bibliothèque dans la construction plus rapide des circuits VLSI dédiés à la communication.

Il faut rappeler qu'une bibliothèque d'opérateurs flexibles est un ensemble de macrofonctions standards (opérateurs flexibles), parfaitement définies au niveau fonctionnel, logique, électrique et implanté qui peuvent être générées automatiquement.

3.2 *Fonctions rencontrées dans les circuits analysés*

Les fonctions rencontrées dans les circuits de communication analysés peuvent être répertoriées de la manière suivante:

. fonctions combinatoires (encodeurs, décodeurs, comparateurs)

. fonctions séquentielles simples (registres à décalage, compteurs, ..),

. fonctions séquentielles complexes (automates, séquenceurs),

. mémoires vives pour la conservation des données temporaires (FIFO),

. mémoires mortes pour la conservation des données permanentes (programmes, réseaux logiques programmables PLAs, ...),

. des fonctions non encore standard (décodeurs et encodeurs biphase,...).

Remarque: Les circuits de communication analysés n'ont pas besoin de réaliser des fonctions arithmétiques du type addition, soustraction, multiplication....., comme dans le cas des circuits microprocesseur ou des circuits utilisés dans le traitement du signal.

3.3 *Classification des opérateurs flexibles*

Une analyse des fonctions précédemment citées et de la manière de pouvoir les implanter automatiquement, nous a permis de dégager, selon la méthode utilisée pour leur génération, deux classes d'opérateurs:

. les opérateurs en tranches ou "bit slice" sont les opérateurs qui grâce à leur modularité et à leur régularité peuvent être générés à partir d'une certaine paramétrisation (nombre de bits, caractéristiques électriques de l'opérateur, position des entrées/sorties, ...). Ces opérateurs ont la caractéristique d'utiliser un jeu de cellules de base (bascules, portes, ...) pour leur construction.

. les opérateurs programmables, ont la caractéristique de pouvoir être générés automatiquement à partir de leur description formelle. Par exemple, l'implantation automatique d'une fonction combinatoire à partir de sa (ou ses) équation(s) logique(s) associée(s) ou l'implantation d'une fonction séquentielle complexe à partir d'un graphe d'état qui décrit son comportement.

3.4 *Présentation des opérateurs flexibles des différentes classes*

Dans ce paragraphe, nous allons présenter plus en détail quels sont les opérateurs constituant les opérateurs en tranches et les opérateurs programmables, ainsi que les fonctions qui leur sont associées.

3.4.1 Les opérateurs en tranches

La structure régulière et modulaire caractéristique de ce type d'opérateurs permet leur réalisation en forme de tranches ou en bandes. Ces tranches, construites par éléments ou cellules juxtaposés les uns aux autres, peuvent être répétées n fois selon les caractéristiques de l'opérateur.

L'utilisation des outils de CAO du type "assembleur de silicium" offre la possibilité de construire ces opérateurs d'une manière programmée et par conséquent de pouvoir définir leurs caractéristiques à partir d'un ensemble de paramètres.

Les fonctions proposées pour être générées par cette méthode d'implantation et les plus couramment rencontrées dans les circuits de communication analysés, se trouvent être les suivantes:

. la fonction combinatoire de comparaison,

. les fonctions séquentielles:

  . de décalage,
  . de mémorisation,
  . de comptage,
  . de génération de CRC.

Ensuite, nous allons présenter chacune de ces fonctions et les opérateurs utilisés pour leur implantation. Nous allons proposer pour chaque opérateur une structure modulaire de construction et la définition de quelques paramètres qu'il est possible de prendre en compte lors de sa génération.

3.4.1.1 La fonction de comparaison

Cette fonction permet de comparer deux mots (sur deux bus) par un OU exclusif distribué, comme il est montré dans la figure 2.3.1:

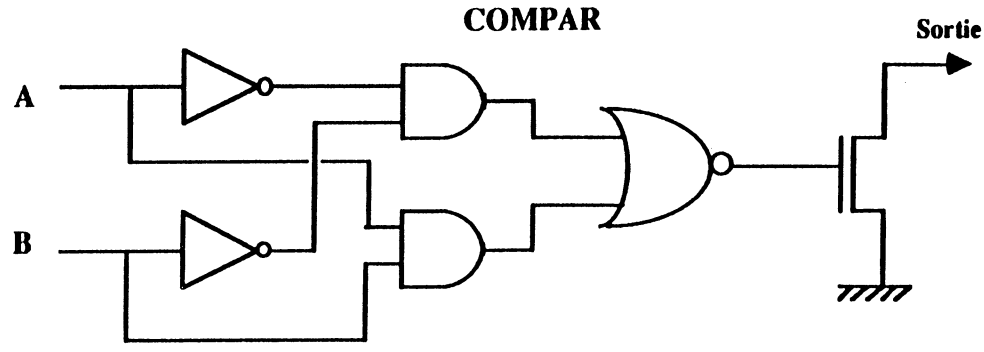


Figure 2.3.1 Schéma logique comparateur

La fonction de comparaison est souvent utilisée pour la détection de l'adresse de la station ou dans l'identification des motifs de synchronisation: caractère, fanion, ... Dans ces deux cas la comparaison peut être faite entre une donnée et un motif déterminé ou un mot stocké en mémoire.

Cette fonction sera implantée par un opérateur dit de comparaison et structuré de la manière suivante:

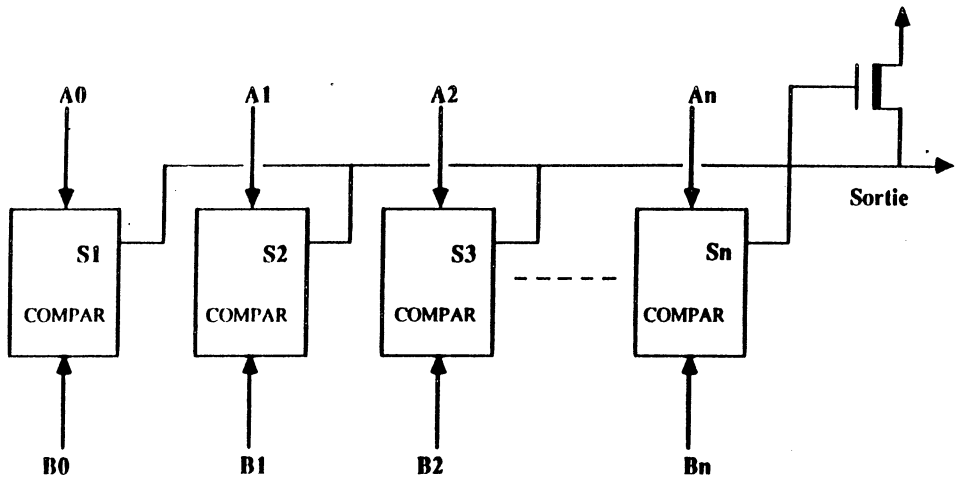


Figure 2.3.2 Structure comparateur

L'opérateur est constitué d'une cellule OU exclusif à drain ouvert (ou par la fonction d'égalité et d'une porte non-OU à drain ouvert) et d'un élément de charge (par exemple, un transistor MOS à dépletion). Il faut signaler que les caractéristiques de l'élément de charge seront définies en fonction du nombre d'entrées ou bits constituant les mots à comparer.

La génération de cet opérateur peut être faite d'une manière flexible par la définition des paramètres suivants: longueur des mots à comparer (nombre de bits), motif ou valeur à comparer, position des entrées et de la sortie, écartement entre chaque bit constituant le mot.

3.4.1.2 La fonction de décalage

La fonction de décalage consiste à faire glisser, A1-An informations binaires contenues dans un registre de n cellules mémoire (bascule D, J-K, R-S maître-esclave), d'une cellule à la suivante par un signal d'horloge (figure 2.3.3).

Cette fonction est très utile pour la réalisation de retards (n coups d'horloges) ou la transformation d'informations série-parallèle, parallèle-série. Ces fonctions peuvent être implantées respectivement par trois types d'opérateurs: registre à décalage, registre désérialisateur et registre sérialisateur.

> Remarque.- la cellule mémoire considérée sera une bascule maître-esclave constituée de deux points mémoires "Latch".

* **Opérateur registre à décalage**

Il réalise la fonction même de décalage. Un registre à décalage est constitué de n cellules mémoires reliées entre elles, comme il est montré dans la figure 2.3.3. L'entrée du registre est l'entrée de la première bascule et la sortie de la nième bascule la sortie du registre. Donc, les informations qui rentrent dans le registre à décalage seront retrouvées sur sa sortie après n signaux de décalage. Les étages intermédiaires ne sont pas accessibles.

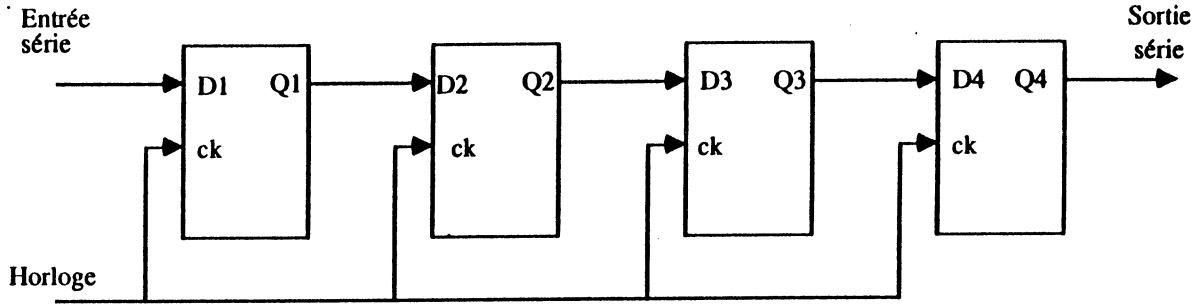


Figure 2.3.3 Schéma logique registre à décalage

La structure modulaire de cet opérateur (figure 2.3.4), par sa simplicité, correspond pratiquement à celle décrite par son schéma logique; elle est constituée d'une simple bascule répétée n fois.

Il est possible de proposer, pour la construction de cet opérateur d'une manière flexible, quelques paramètres: nombre de bits (bascules), écartement entre bits, position de l'entrée et position de la sortie.

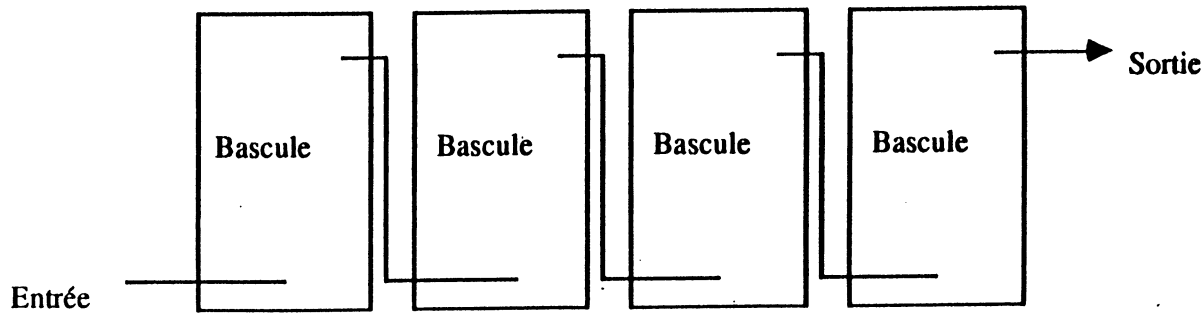


Figure 2.3.4 Structure registre à décalage

* Opérateur registre désérialisateur

La fonction réalisée par cette opérateur est la transformation d'une séquence de données série en une représentation parallèle, par exemple, lorsqu'il s'agit de recevoir des mots d'information d'une ligne série.

Le schéma logique de base du registre désérialisateur correspond à celle du registre à décalage, mais avec les sorties de tous les étages accessibles (figure 2.3.5).

La structure modulaire est donc la même que celle définie pour le registre à décalage plus les sorties de chaque étage du registre.

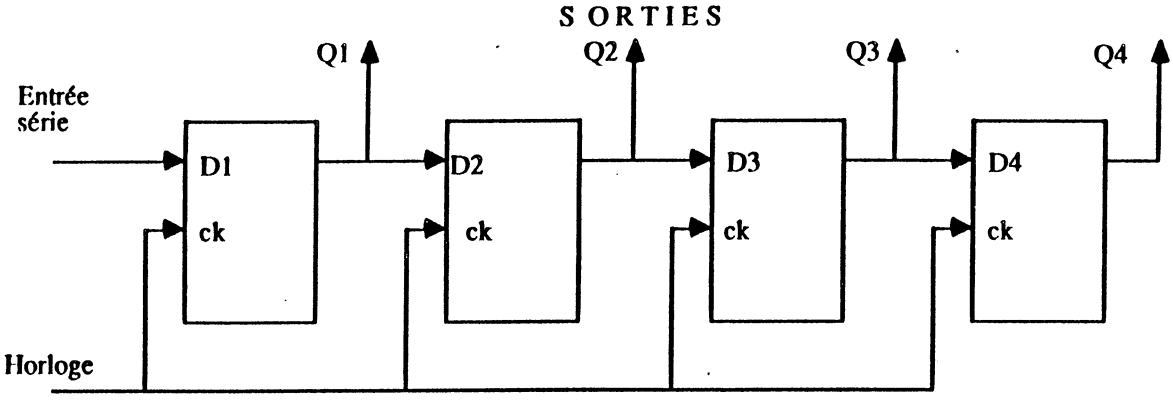


Figure 2.3.5 Schéma désérialisateur.

Plusieurs caractéristiques peuvent être ajoutées à cet opérateur, telles que la remise à zéro du registre par un signal de commande (par exemple le signal de RESET du système). Cette caractéristique est nécessaire dans certaines applications, pour éviter d'avoir des configurations erronées à l'initialisation du système.

Dans ces conditions, la fonction demandée peut être effectuée par le simple changement de la bascule par une bascule avec mise à zéro (asynchrone ou synchrone), ou par l'addition d'une cellule (par exemple une porte ET) connectée à l'entrée de chaque étage. Cette porte met à zéro l'entrée de la bascule sous la présence du signal de contrôle (dans ce cas la remise à zéro est synchrone). La figure 2.3.6 illustre la deuxième solution.

La structure modulaire de chaque bit du désérialisateur sera constituée, comme il est illustré dans la figure 2.3.7, par une bascule maître-esclave et une porte ET. Le signal de remise à zéro peut éventuellement être amplifié par une cellule d'amplification (amplificateur-inverseur) liée à l'opérateur ou placée ailleurs.

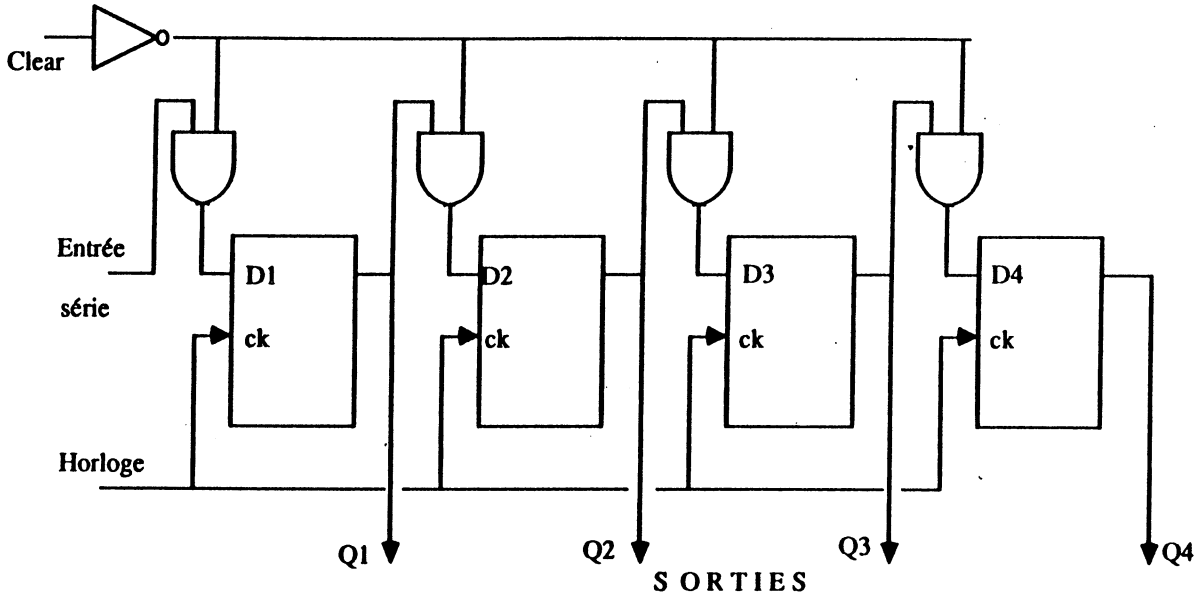


Figure 2.3.6 Schéma logique désérialisateur
avec remise à zéro synchrone

Entre les différentes caractéristiques qu'un registre désérialisateur peut comporter, il est possible de définir pour sa génération automatique les paramètres suivants: nombre de bits, écartement entre bits, position de l'entrée et des sorties, avec ou sans r.a.z, r.a.z synchrone ou asynchrone...

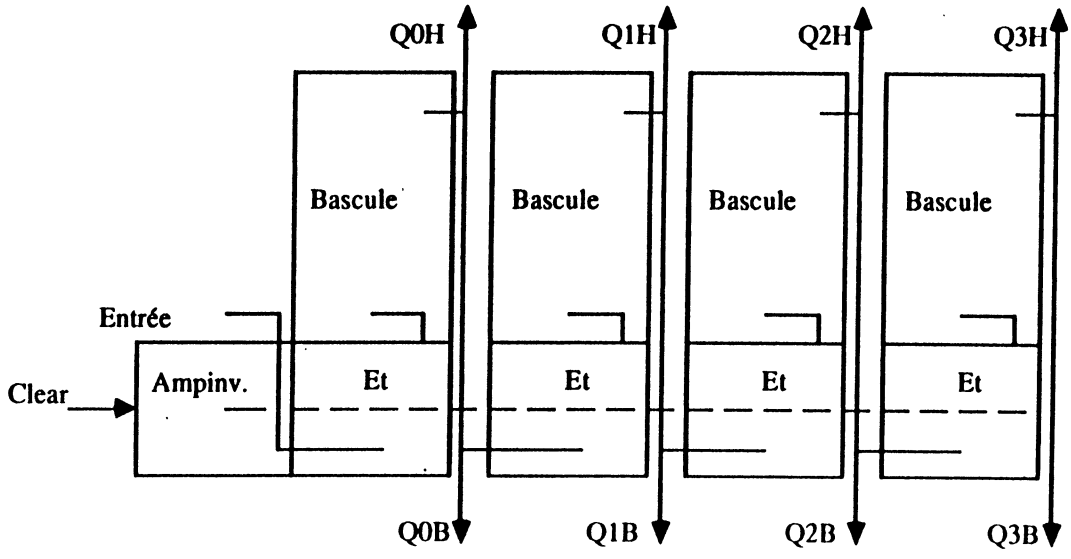


Figure 2.3.7 Structure registre désérialisateur avec remise à zéro

* Opérateur registre sérialisateur

La fonction réalisée par cet opérateur est la transformation des informations du mode parallèle au mode série. Cette fonction est utilisée lorsqu'il s'agit d'émettre des mots d'information sur une ligne de transmission série.

Le schéma logique (figure 2.3.8) de cet opérateur correspond à celui d'un registre à décalage avec une logique combinatoire à l'entrée de chaque étage du registre. La logique est un multiplexeur 2 vers 1 (ou sélecteur 1 parmi 2), qui permet soit le stockage des mots d'information s'il y a présence d'un signal de chargement, soit le décalage des informations dans le cas contraire.

La structure modulaire de cet opérateur (figure 2.3.9) correspond à une tranche constituée d'une bascule et d'un sélecteur, répétés n fois. La logique de chargement, comme dans le cas précédent, peut être considérée dans l'opérateur lui même ou placée ailleurs.

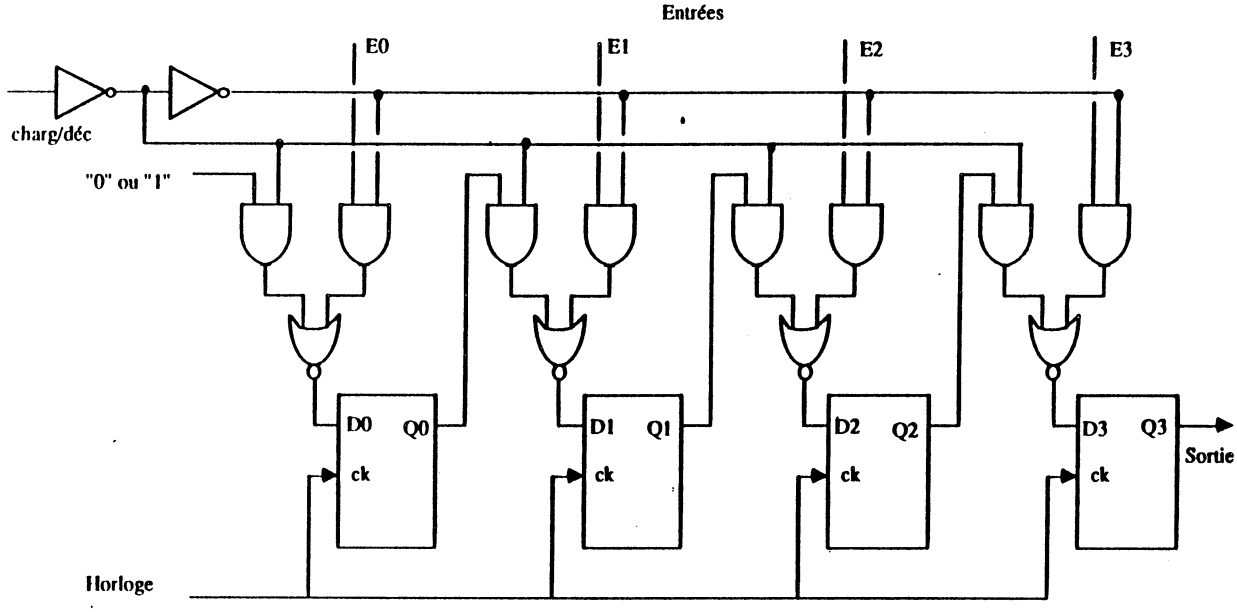


Figure 2.3.8 Schéma logique sérialisateur

Le registre sérialisateur, comme dans le cas des opérateurs antérieurs, peut être généré à partir de certains paramètres: nombre de bits, écartement entre bits, position des entrées et position de la sortie ....

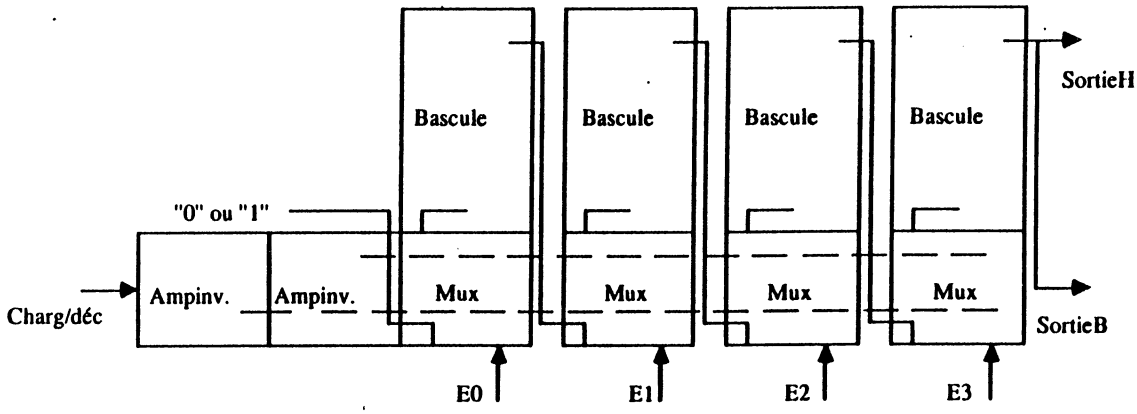


Figure 2.3.9 Structure sérialisateur.

3.4.1.3 Fonction de mémorisation

La fonction de mémorisation consiste à stocker la présence d'une information fugitive pendant un certain temps. Ce temps peut durer une période d'horloge ou l'intervalle de temps déterminé entre deux signaux de stockage ou d'écriture.

Plusieurs dispositifs sont utilisés pour la mémorisation des informations binaires: les bascules pour mémoriser une seule information, les registres parallèles pour en stocker quelques-unes (8, 16,....), ou les mémoires utilisées pour en stocker un grand nombre.

Dans cette partie nous allons présenter, les registres parallèles en raison de leur méthode de construction et de leur grande utilisation dans les circuits de communication analysés.

* **Opérateur registre parallèle**

La fonction réalisée par cet opérateur est le stockage de mots d'information. Il est utilisé dans l'implantation des ports d'interface parallèle, dans l'implantation des registres à lecture seule ou à écriture seule, ...

Ces registres sont constitués de n cellules mémoire indépendantes les unes des autres et d'un sélecteur associé à chacune d'elles, (voir figure 2.3.10). Le sélecteur assure le maintien de l'information contenue dans le registre tant qu'il n'y a pas une demande de chargement des nouvelles informations. Lorsqu'un signal de

mémorisation arrive, de nouvelles informations sont stockées.

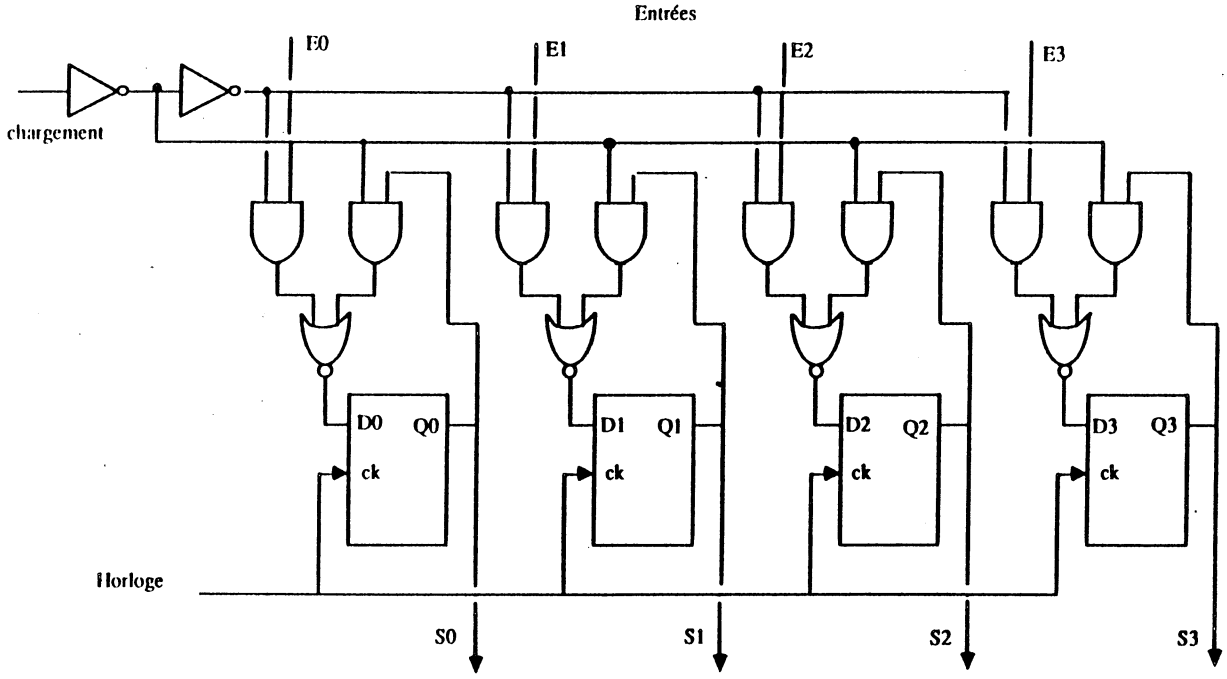


Figure 2.3.10 Schéma logique registre parallèle

La structure modulaire de cet opérateur est identique à celle employée pour le sérialisateur, figure 2.3.9.

Les paramètres utilisées pour la génération de cet opérateur sont le nombre de bits, l'écartement entre bits, la position des entrées et des sorties.

> **Remarque**: du point de vue structurel, le registre sérialisateur est très similaire au registre parallèle. La seule différence est la manière de connecter une des entrées du sélecteur. Pour le sérialisateur l'entrée correspond à la sortie de l'étage précédent, et dans le cas du registre parallèle l'entrée est le rebouclage de l'information sortie du même étage.

3.4.1.4 Fonction de comptage

Cette fonction inclut les fonctions de temporisation, de séquencement et de division de fréquence.

. Temporisation.- elle consiste à compter une impulsion périodique de fréquences donnée jusqu'à ce que la durée souhaitée soit atteinte. Ce comptage peut être fait avec un compteur ou avec un décompteur. Dans le premier cas les états de sortie du compteur doivent être décodés à la valeur de la durée de temporisation. Dans le deuxième cas, cette valeur doit être chargée dans le décompteur comme valeur d'initialisation du comptage. Lorsque le compteur atteint la valeur zéro la temporisation a été réalisée.

. Séquencement.- un compteur peut être utilisé pour réaliser un séquencement d'opérations. Chaque position du compteur est décodée (elle est caractéristique de l'action à effectuer).

. Division de fréquence.- l'application d'un signal de fréquence f sur l'entrée d'un compteur binaire donne comme résultat:

f/2 à la sortie du 1er étage,
f/4 à la sortie du 2ème étage,
.
.
.
$f/2^n$ à la sortie du nième étage,

ainsi, y aura-t-il autant de fréquences que d'étages dans le compteur.

Ces fonctions sont très souvent rencontrées dans les circuits de communication analysés: dans la génération des diverses phases d'horloges des circuits, pour fixer la fréquence de transmission pour le contrôle de la sérialisation ou désérialisation des données, pour réaliser des temporisations ou "chiens de garde",....

Remarque : il existe une vaste gamme de compteurs (synchrones, asynchrones,...) et différentes façons de les synthétiser. Dans cette présentation, nous allons uniquement considérer, comme exemple de réalisation, les compteurs et décompteurs synchrones par leur plus grande complexité d'implantation vis-à-vis des compteurs asynchrones.

* Eléments principaux d'un compteur synchrone

L'élément de base dans tout compteur est une bascule (D, J-K, ...). Dans le cas des compteurs synchrones, une logique combinatoire pour le comptage et le calcul de la retenue est associée à chaque étage du compteur. L'ensemble peut être réalisé au travers d'un demi-additionneur (pour les compteurs) ou d'un demi-soustracteur (pour les décompteurs), (figure 2.3.11).

Le calcul de la retenue peut être effectué:

. soit en série, entre deux étages adjacents (par exemple i-1 et i) et le résultat est utilisé à l'étage suivant i+1, (voir figure 2.3.11),

. soit en parallèle. Pour un étage i il faut prendre en compte les valeurs de tous les étages précédents (i-1 étages), voir figure 2.3.12.

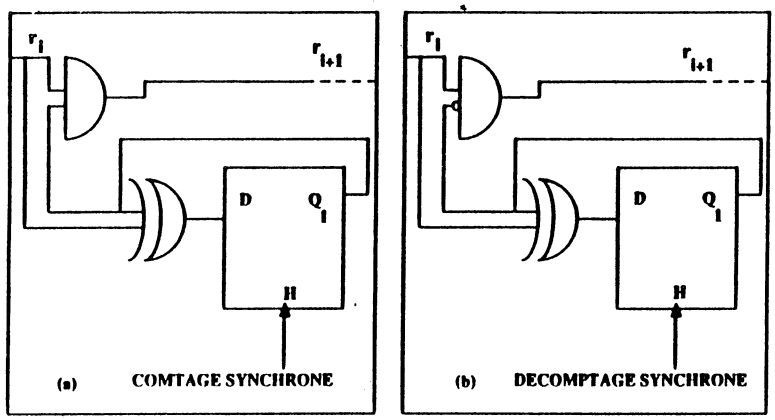


Figure 2.3.11 Calcul de la retenue série.

Les compteurs à retenue parallèle offrent la possibilité de compter très rapidement. Par contre, les compteurs à retenue série ne permettent pas d'atteindre des fréquences très élévées, car le temps de propagation de la retenue doit être inférieur au temps séparant le front de descente du signal d'horloge de son front de montée.

En raison de cette particularité, nous avons décidé d'implanter des compteurs et des décompteurs à retenue parallèle. Mais pour éviter d'avoir des portes logiques avec beaucoup d'entrées et peu adaptées à une implantation modulaire, nous avons décidé de réaliser un calcul de la retenue série par groupes de 4 bits, comme il est montré dans les figures 2.3.12, 2.3.13 et 2.3.14.

Les éléments qui constituent la logique pour le calcul de la retenue sont les portes: non-ou (2, 3 et 4 entrées) et une porte non-ou distribuée (pour le calcul de la retenue par groupe de 4 bits). Pour effectuer l'addition dans le cas des compteurs, une porte logique égalité (non-ou-exc) est nécessaire. Dans le cas des décompteurs la soustraction est réalisée à travers une porte ou-exclusif.

* Compteurs

Suite aux considérations faites ci-dessus, nous allons présenter trois types de compteurs synchrones:

. compteur avec remise à zéro (r.a.z) asynchrone (figure 2.3.12),

. compteur avec r.a.z synchrone (figure 2.3.13),

. compteur avec affichage et r.a.z synchrone (figure 2.3.14).

Un des objectifs recherchés dans la présentation de ces compteurs, mises à part leur synthèse et leur construction, est de montrer la structure modulaire présentée par ces opérateurs.

Ainsi par exemple, pour passer du premier type de compteur au deuxième (figures 2.3.15 et 2.3.16), suffit il de changer la bascule avec remise à zéro asynchrone par une bascule "normale" et ajouter une porte ET à son entrée pour obtenir un compteur avec r.a.z synchrone. Pour passer de ce compteur au troisième compteur avec affichage et r.a.z synchrone, il suffit de remplacer la porte ET par un sélecteur (figure 2.3.17). Ce troisième compteur offre la possibilité de définir la valeur de départ du compteur.

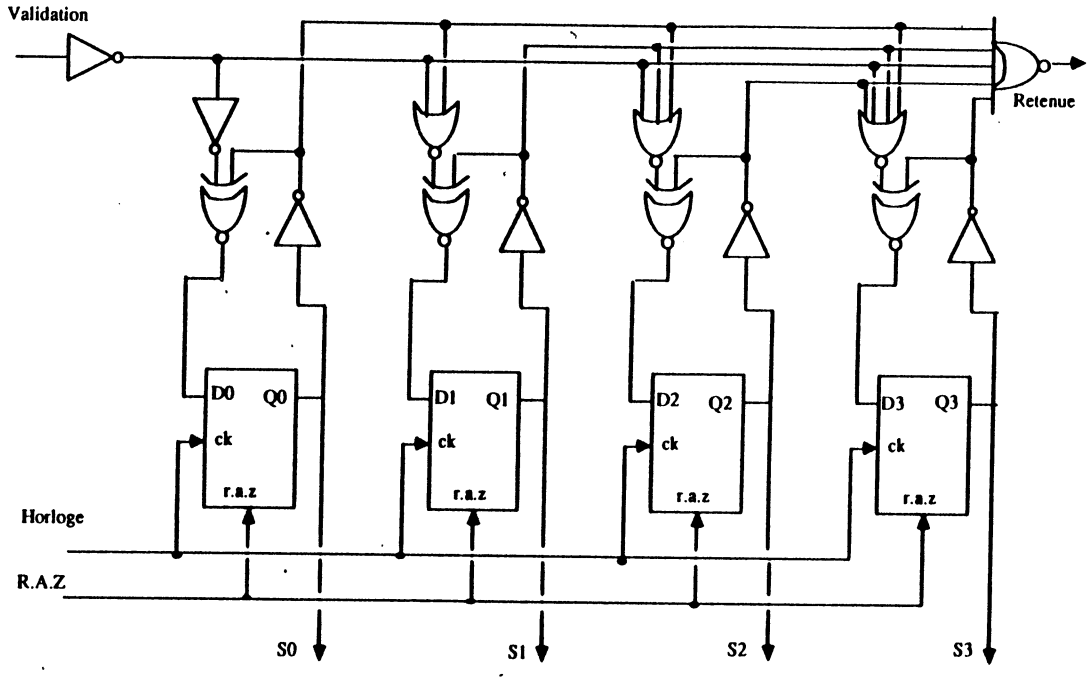


Figure 2.3.12 compteur synchrone avec r.a.z asynchrone

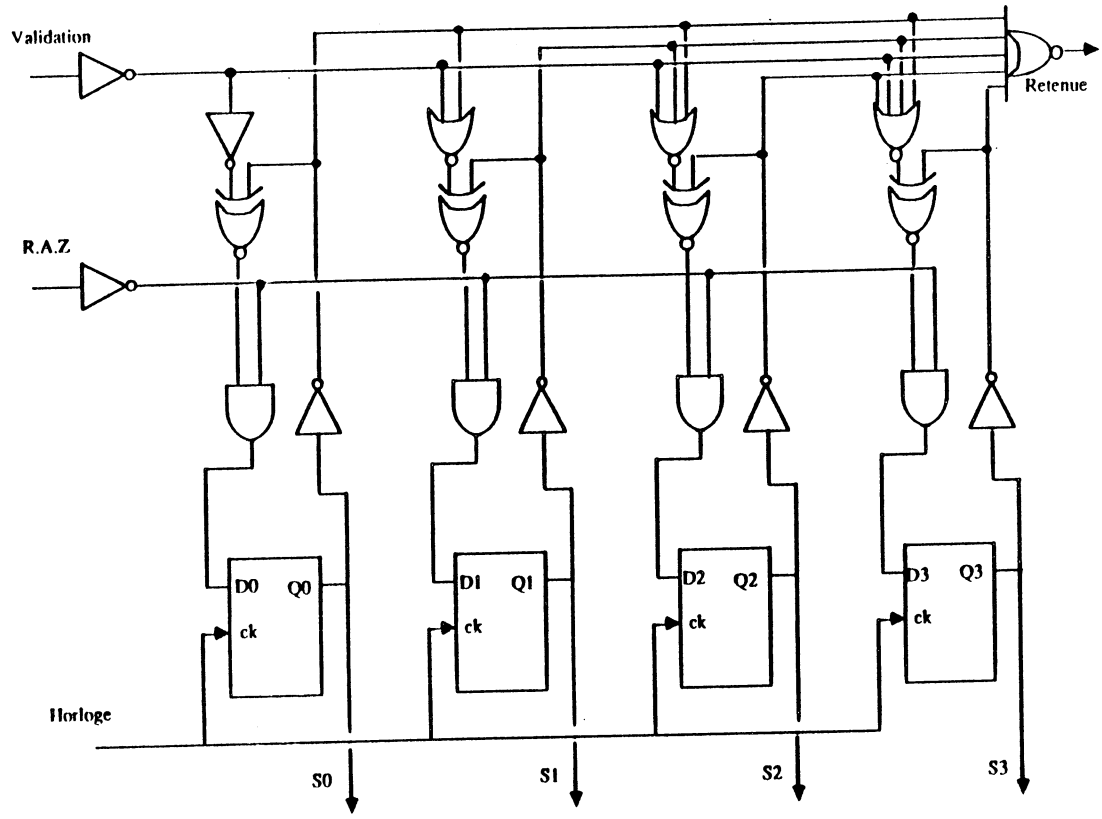


Figure 2.3.13 compteur synchrone avec r.a.z synchrone

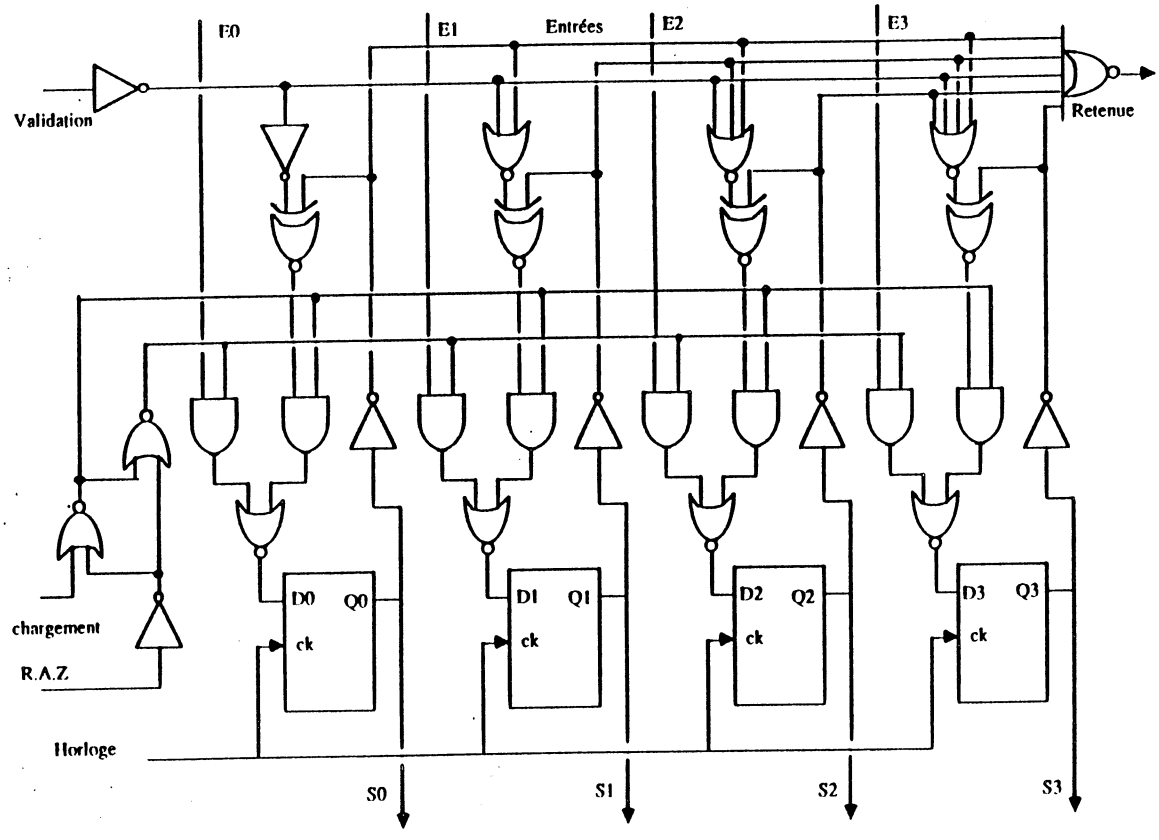


Figure 2.3.14 compteur synchrone avec affichage et r.a.z synchrone

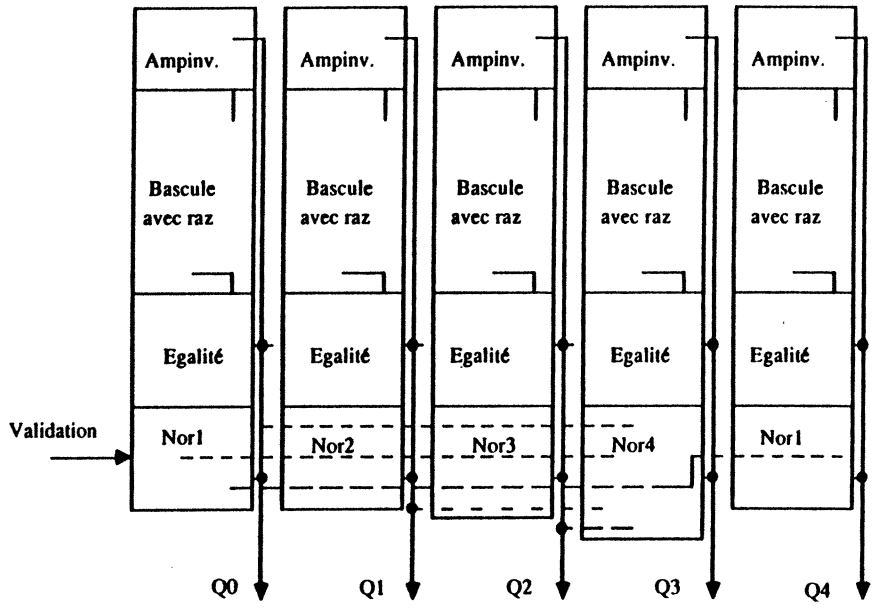


Figure 2.3.15 Structure compteur synchrone avec r.a.z asynchrone

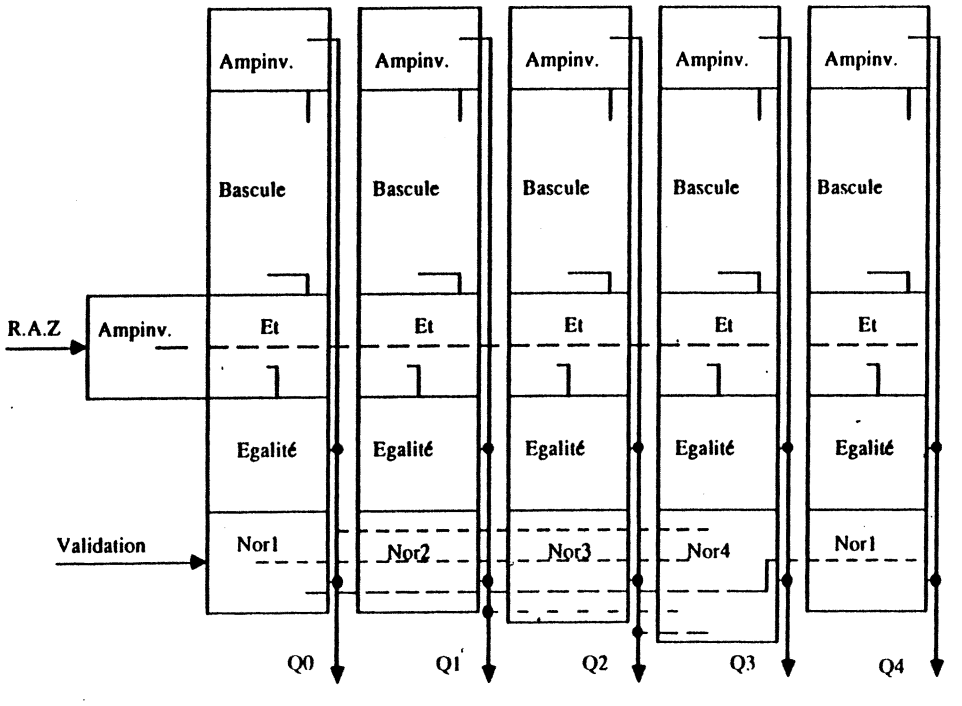


Figure 2.3.16 Structure compteur synchrone avec r.a.z synchrone

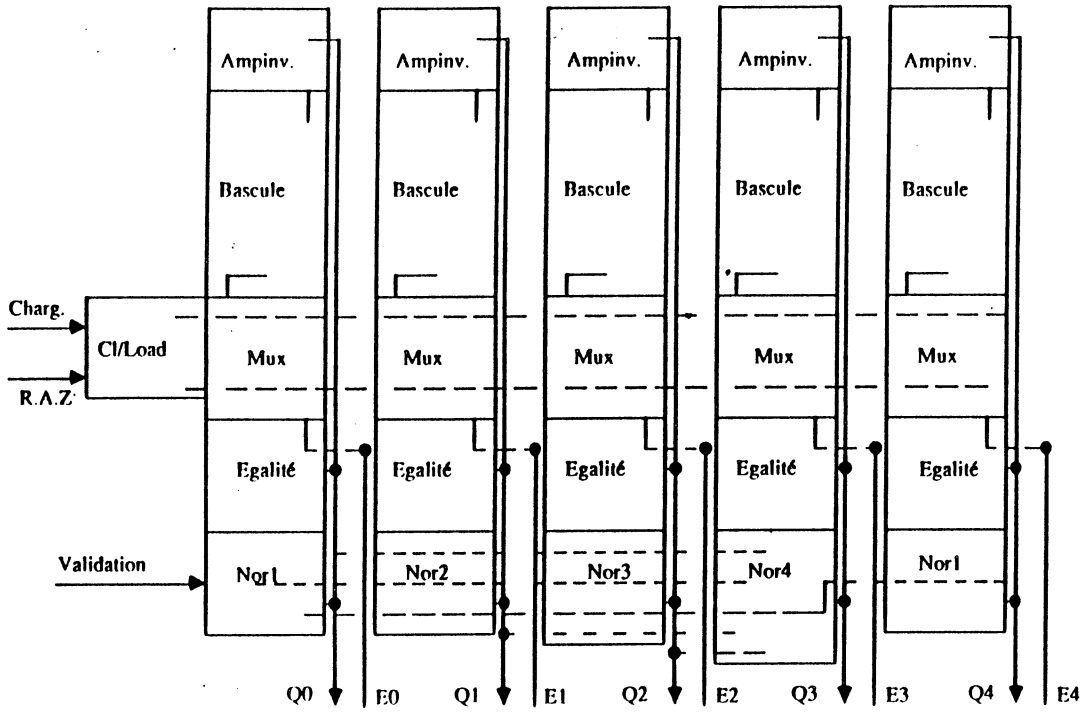


Figure 2.3.17 Structure compteur synchrone
avec affichage et r.a.z synchrone

* **Décompteur**

La fonction à réaliser par ce type d'opérateur est le décomptage. Ce type d'opérateur est très utilisé pour armer des "chiens des garde", faire des temporisations,...

Nous allons proposer ici un décompteur synchrone avec affichage et r.a.z synchrone, son schéma logique est illustré par la figure 2.3.18. Une description des éléments qui le constituent a été déjà faite ci-dessus.

Si l'on considère la structure modulaire présentée pour les compteurs, il est aussi possible de l'utiliser pour cet opérateur. Il suffit de remplacer deux cellules de la structure du dernier compteur présenté pour obtenir celle du décompteur :

. l'amplificateur inverseur par un simple amplificateur. Ainsi, au lieu de prendre les sorties inversées de chaque étage, faut il considérer les sorties directes,

. la cellule égalité réalisant la fonction d'addition doit être changée par une cellule ou-exc pour effectuer la fonction de soustraction.

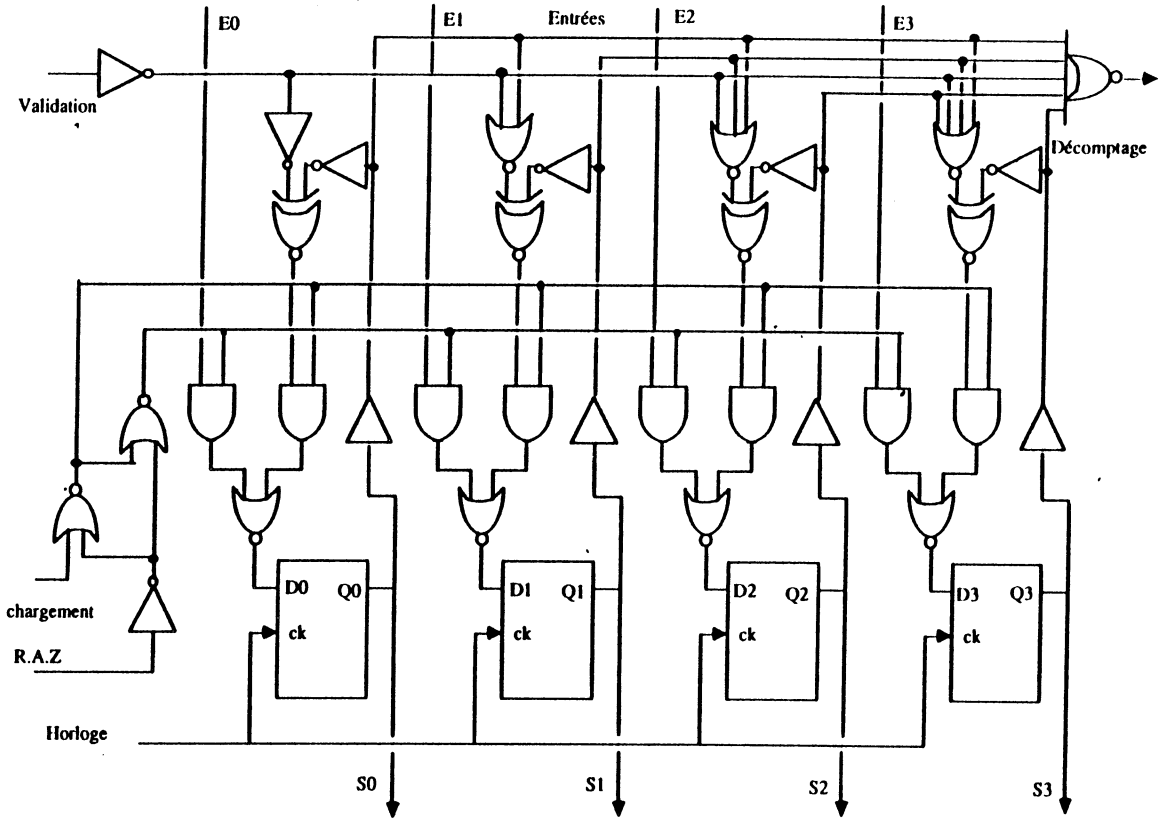


Figure 2.3.18 Décompteur synchrone avec affichage et r.a.z.s

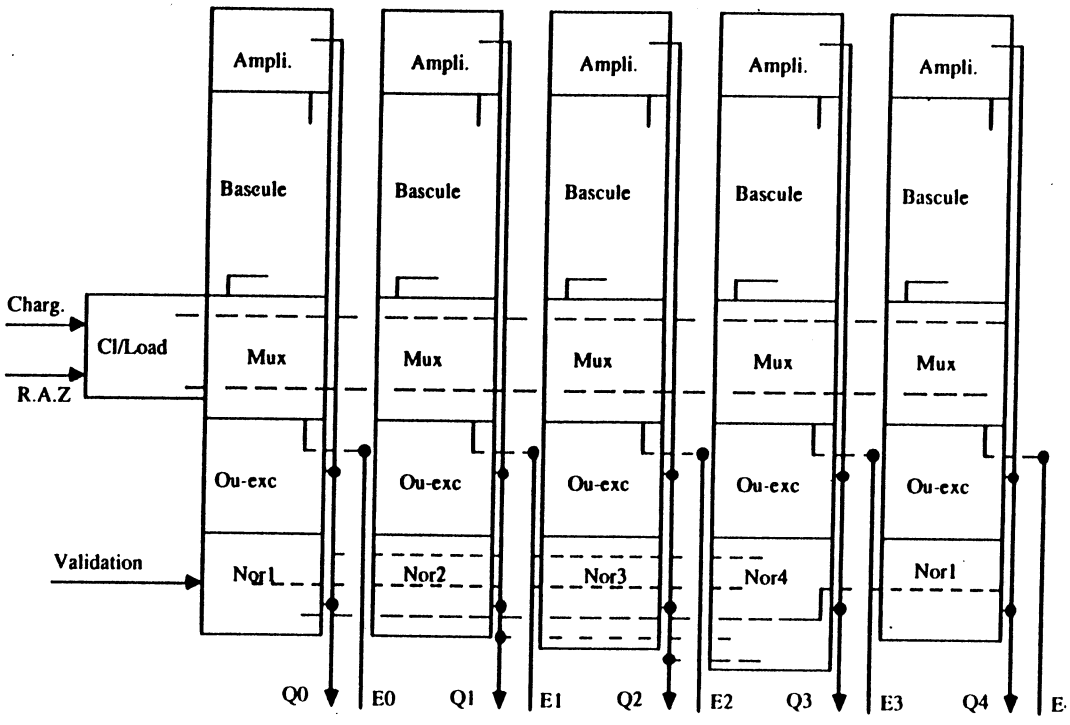


Figure 2.3.19 Structure décompteur synchrone
avec affichage et r.a.z synchrone

Le résultat obtenu par ces remplacements est montré dans la figure 2.3.19. Elle montre la même structure modulaire que pour les opérateurs précédents.

De la même manière que pour tous les opérateurs déjà présentés, la génération des opérateurs de comptage peut être faite par la définition de quelques paramètres: nombre d'étages du compteur (nombre de bits), écartement entre bits, position des entrées et position des sorties, ...

> **Remarque** : la sortie de chacun des étages des opérateurs de comptage présentés est amplifiée par un amplificateur ou un amplificateur inverseur, ceci pour assurer les caractéristiques dynamiques des opérateurs.

3.4.1.5 La fonction de calcul de CRC

L'utilisation d'un champ de séquence de contrôle de trame (FCS) permet de vérifier qu'une trame ou un message reçu sont bien conformes à ceux qui ont été émis.

La génération de cette séquence de contrôle est réalisée à travers des circuits dits générateurs de code de redondance cyclique (CRC). Ces circuits générateurs sont des registres polynômiaux, ainsi appelés car leur structure dérive d'un polynôme générateur associé.

Les propriétés de ces codes polynômiaux [PET-76] sortent du cadre de cette étude. Ici, nous allons uniquement présenter la manière de les implanter automatiquement.

Les fonctions réalisées par ces circuits sur une trame d'information sont:

. en émission (codage), la génération du champ FCS,

. en réception (décodage), la vérification du champ FCS.

* Structure générale

Plusieurs architectures existent pour l'implantation des circuits générateurs de CRC : en série, à base de registres à décalage, en parallèle à base d'un chemin de données parallèle et à base d'un PLA [DEP-85]. De ces trois architectures possibles, nous allons uniquement considérer l'architecture série.

La structure générale d'une architecture série, pour un circuit générateur de CRC, peut être définie comme le chaînage, à travers des portes OU-EXClusives, de plusieurs registres à décalage de longueurs (nombre d'étages) différentes.

. Le nombre de registres à décalage est défini par le nombre de termes du polynôme moins un. Ainsi par exemple, pour le polynôme générateur suivant:

$$g(x) = X^{16} + X^{12} + X^{5} + 1$$

nombre de registres = 3 (4 termes - 1)

. le nombre d'étages de chaque registre est défini par la différence des exposants de deux termes consécutifs. Dans le cas du polynôme précédent:

premier registre = 5 = 5 - 0 (X^5 et X^0);

deuxième registre = 7 = 12 - 5 (X^{12} et X^5);

troisième registre = 2 = 16 - 12 (X^{16} et X^{12});

Les résultats obtenus sont illustrés par les figures 2.3.20 et 2.3.21. Elles montrent respectivement :les schémas logiques des fonctions de codage et de décodage.

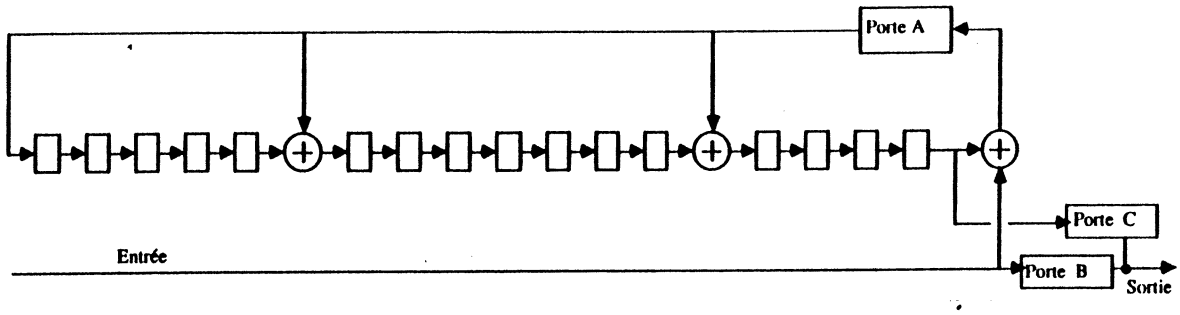


Figure 2.3.20 Codeur CRC

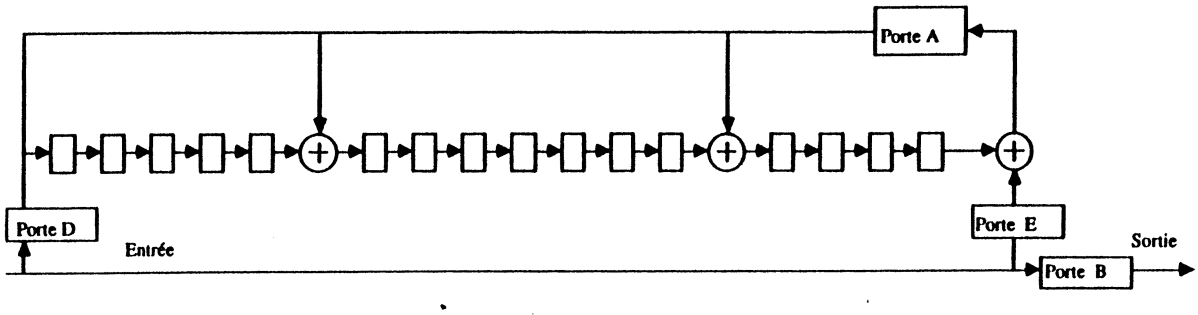


Figure 2.3.21 Décodeur CRC

Implantation

Lors du calcul de la séquence de FCS, une considération importante à faire est le type d'initialisation du circuit générateur: une initialisation à "0" ou une initialisation à "1".

1.- Initialisation à zéro

Le circuit générateur doit réaliser les fonctions suivantes:

. en émission l'initialisation à "0" des registres et l'inversion des n premiers éléments de la trame. n correspond à l'exposant de degré le plus élevé du polynôme,

. en réception l'inversion de la séquence FCS, avant d'entrer dans le registre. En l'absence d'erreurs, le registre FCS devra contenir des "0" après que le FCS soit entré.

2.- Initialisation à "1"

. dans le cas ci-dessus, une inversion des 16 premiers éléments binaires est équivalente à une initialisation à 1 du registre, et une inversion du FCS à la réception amène les registres à l'état "0".

Remarque: L'initialisation à "1" du registre permet de protéger contre les erreurs affectant les fanions d'ouverture de trame, qui peuvent être non détectables si le reste initial est zéro. Le récepteur a également la possibilité de ne pas inverser le FCS, mais, dans ce cas, il doit trouvées un reste unique. Par exemple, dans le cas du polynôme donné précédement le reste est: 0001110100001111 (de X15 à X0). Ce reste non nul permet de se protéger contre la possibilité de l'absence de séquence de fin de trame.

Une explication plus détaillée et une justification de cette méthode peuvent être trouver en [MAN-77] [MAC-79].

Donc, pour des raisons de simplicité d'implantation et d'une meilleure protection contre les erreurs de transmission, nous allons faire, pour la génération automatique des circuits générateurs de CRC, les considérations suivantes:

. l'initialisation à "1" des registres à décalage en émission, comme en réception,

. en émission, l'inversion de la séquence de FCS calculée avant d'être transmise,

. en réception, la détection d'un reste unique,

. la possibilité d'utiliser le même circuit générateur de CRC comme codeur et décodeur.

Si les figures 2.3.20 et 2.3.21 illustrent les schémas logiques séparés des générateurs de CRC en émission et en réception respectivement (selon le polynôme conseillé par le CCITT), le schéma logique d'un circuit générateur de CRC, en prenant en compte les remarques faites ci-dessus, est illustré par la figure 2.3.22 suivante:

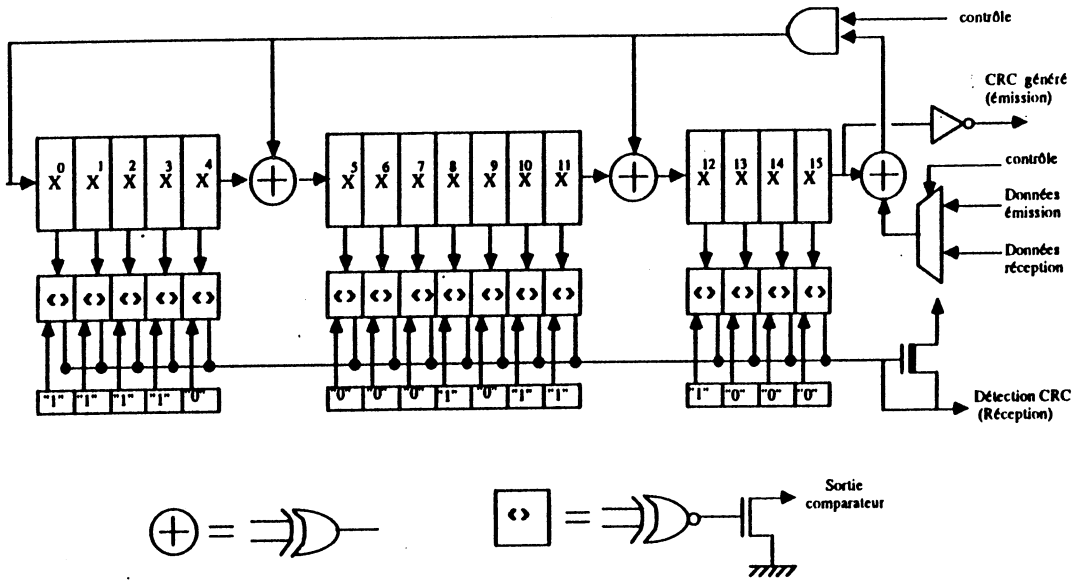


Figure 2.3.22 Circuit codeur/décodeur CRC

Pour l'implantation du schéma logique, les éléments utilisés seront: une bascule maître-esclave avec remise à "1", une porte OU-EXClusive, une porte ET, un sélecteur, des inverseurs et une logique combinatoire de comparaison (la même définie un peu plus haut) utilisée pour vérifier le reste unique.

blocs flexibles

A partir de la structure générale des circuits générateurs de CRC définie ci-dessus, il est possible de proposer une structure topologique d'accueil pour leur implantation automatique. La figure 2.3.23 illustre cette structure pour le cas le plus général: codage et décodage. Des structures plus simples (codage ou décodage) seront obtenues à partir de celle-ci.

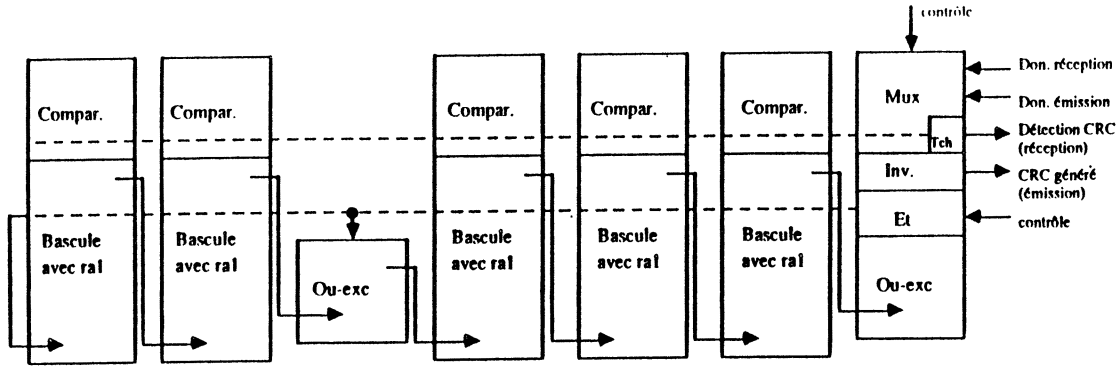


Figure 2.3.23 Structure générale des circuits générateurs de CRC

Enfin, pour terminer, il reste à indiquer les paramètres à utiliser pour la génération des opérateurs de calcul de CRC, ce sont les suivants:

. le polynôme générateur,

. le type de générateur à utiliser (en émission "codeur", en réception "décodeur" ou codeur/décodeur),

. en réception, définir le reste unique.

3.4.2 Les opérateurs programmables

Ces opérateurs, selon le type de fonction logique réalisé, peuvent être classifiés en deux groupes :

. les opérateurs combinatoires,

. les opérateurs de contrôle.

3.4.2.1 Les opérateurs combinatoires

Les fonctions qualifiées de logique aléatoire (qui se traduit par la présence des équations logiques sans régularité apparente) seront considérées dans ce type d'opérateurs dits combinatoires.

Ces fonctions (du type décodage, encodage, égalité, ...) peuvent être réalisées à partir de leurs équations logiques sous deux formes différentes :

. à base de portes ET et OU (NON-ET et NON-OU en logique inverse),

. à base d'un réseau logique programmable ou PLA (Programmable Logic Array).

La première solution présente l'inconvénient d'un manque de régularité et par conséquent une implantation automatique paramétrable difficile, si bien que ces fonctions peuvent être implantées, en rangées selon la méthode utilisée dans les circuits de cellules précaractérisées. Mais cette possibilité n'est pas considérée dans notre étude (ne remplissant pas les conditions ou critères fixés pour la réalisation de cette bibliothèque).

La seconde solution, par contre, présente plusieurs avantages: la régularité de dessin des PLAs et sa facilité de programmation.

L'implantation automatique de ces opérateurs peut être effectuée directement à partir des équations logiques définissant la fonction. Cette implantation sera faite à l'aide d'un outil logiciel (un Générateur Automatique de Séquenceurs (GASP) [FLA-84]). Une présentation plus détaillée de ce système sera faite dans 2.3.

Un inconvénient de cette solution, selon le type d'opérateur est: le nombre de monômes de la fonction à implanter et l'augmentation de la surface vis-à-vis de la première solution.

L'implantation automatique de diverses fonctions combinatoires sera présentée dans 2.4.2.

3.4.2.2 Les opérateurs de contrôle

Nous allons considérer l'architecture des circuits de communication comme deux parties séparées: une partie contrôle et une partie opérative. La partie opérative est constituée principalement des opérateurs qui ont été décrits dans les paragraphes précédents.

La partie contrôle regroupe les fonctions de séquencement (décision des actions à entreprendre en fonction de l'état des entrées et des compte-rendus) et de distribution de commandes (nécessaires pour exécuter l'action en cours).

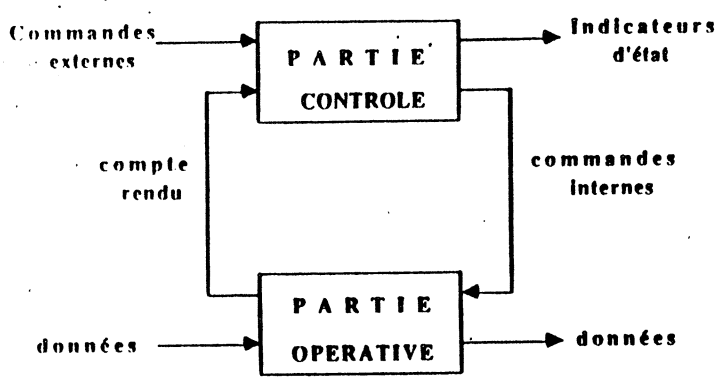


Figure 2.3.24 Partie opérative et partie contrôle

Si le terme de séquenceur est le plus généralement employé, contrôleur ou automate sont également significatifs de la fonction.

Dans le cas des circuits de communication analysés, les fonctions de contrôle à réaliser sont du type: gestion de la méthode d'accès, gestion de l'interface bus local, et contrôle de la partie opérative du circuit. Mais ces circuits n'ont pas besoin, comme dans les microprocesseurs, de séquenceurs de cycle d'instruction ni de séquenceurs de gestion des micro-instructions.

Les fonctions de contrôle mentionnées peuvent donc être générées et implantées automatiquement à l'aide de GASP sous la forme d'un séquenceur ou automate. Les architectures d'accueil utilisées pour l'implantation peuvent être: un mono-PLA ou une mémoire ROM.

Le langage de description utilisé est une description formelle: graphe d'états proches des réseaux de Petri.

Un exemple de réalisation de ces opérateurs de contrôle sera illustré dans le paragraphe 2.4.2.3.

Les principales caractéristiques du système GASP seront présentées en peu plus loin.

3.4.3 Divers

Il est certain qu'il existe, dans les circuits de communication analysés, des modules et donc des opérateurs qui ne sont pas considérés dans les divers types d'opérateurs présentés. Ceci est dû principalement à leurs caractéristiques fonctionnelles (voire logiques) ou à des contraintes électriques qui obligent à personnaliser l'opérateur en question.

Parmi ces opérateurs, on peut citer: les codeurs et décodeurs Manchester; les circuits de synchronisation; les circuits de récupération d'horloge ...

Si les fonctions sont bien répertoriées et caractérisées (comme les codeurs, décodeurs cités), il est toujours possible de les implanter manuellement et de les stocker comme des blocs ou comme des cellules existantes.

Il peut arriver que des fonctions soient spécifiques du circuit à réaliser; dans ce cas, le concepteur se verra obligé soit d'utiliser les cellules existantes pour l'implantation de l'opérateur, soit de le dessiner entièrement ou partiellement à "la main". Ceci a l'avantage d'enrichir la ou les bibliothèques de cellules et d'opérateurs.

3.5 *Outils à utiliser pour générer les opérateurs*

La mise en oeuvre des divers opérateurs définis dans le paragraphe précédent va être réalisée à l'aide des systèmes ou outils logiciel de CAO suivants:

. LOF : pour la partie opérative (opérateurs en tranches),

. GASP : pour la partie contrôle (opérateurs de contrôle), mais il est vrai qu'il permet en même temps la génération des opérateurs combinatoires de la partie opérative.

3.5.1 L O F (Langage d'Opérateurs Flexibles)

LOF permet la construction d'opérateurs flexibles par l'assemblage de cellules de base. Ces cellules définies dans une bibliothèque ont une taille fixe et dessinée à la main, elles peuvent être contenues dans un système de CAO, en l'occurrence CASSIOPEE.

La figure 2.3.25 illustre l'organisation de LOF et son environnement pour la synthèse d'un opérateur flexible et pour la constitution d'une bibliothèque de modules.

* Synthèse d'un opérateur flexible

Un opérateur flexible peut donc être synthétisé à partir de:

. un ensemble de cellules,

. un ensemble de règles d'assemblage, appelé module ou programme LOF,

Le résultat de l'exécution du programme LOF est stocké dans la base de données de CASSIOPEE.

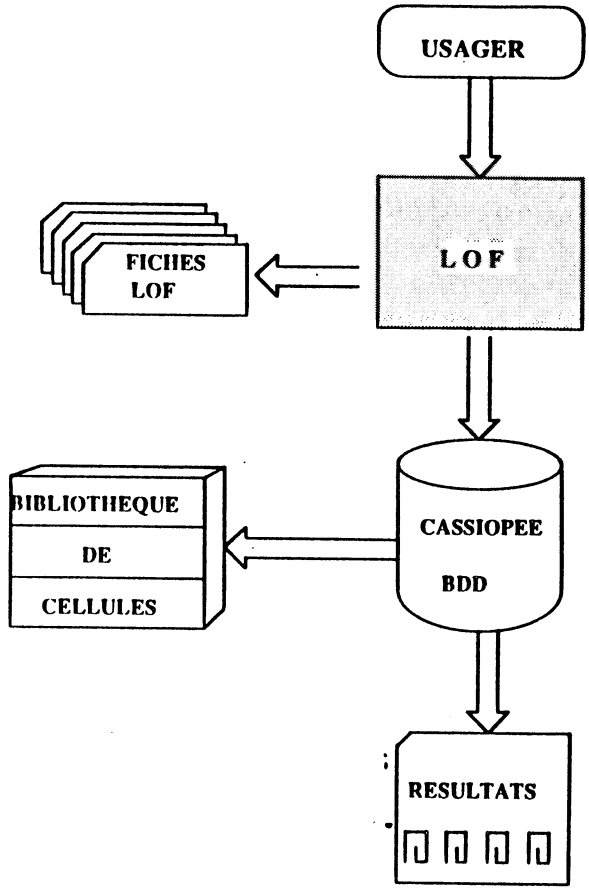


Figure 2.3.25 Organisation de LOF et son environnement

*** Bibliothèque de modules**

Il est possible de former une bibliothèque de modules LOF. Ils peuvent être mis au point séparément, compilés et regroupés dans un même fichier.

Tous les modules d'une bibliothèque peuvent s'appeler les uns les autres. Ceci permet une certaine mise en commun de cellules de base ou d'opérateurs simples pour la synthèse d'opérateurs complexes présentant des similarités.

C'est le système LOF qui assure la gestion de tous les modules LOF ainsi que les relations avec leur environment.

*** Quelques caractéristiques du langage LOF**

LOF est un langage procédural structuré sans déclaration de variables. Il peut manipuler les types de données suivants: réels, chaîne de caractères, fichier de texte, cellules, tableaux des différents types mentionnés sauf de fichier.

A noter que le type cellule est utilisé pour représenter une partie quelconque d'un dessin de masques.

Différents types de fonctions et opérateurs existent pour manipuler les cellules:

. fonctions pour connaître quelques caractéristiques des cellules; longueur, hauteur, surface, nombre de transistors, densité,....

. fonctions de symétrie, rotation, répétition,

. instantiation: une construction spécifique permet d' ajouter des fils de connexion (aluminium, poly, diffusion) sur une cellule de base pour créer une nouvelle cellule par instantiation,

. les opérations définies par LOF sont du type : assemblage par aboutement horizontal, vertical ou arbitraire,....

3.5.2 G A S P

GASP (Générateur Automatique de Séquenceurs Programmables), est un système qui, comme son nom l'indique, permet la synthèse de séquenceurs utilisés dans la partie contrôle des circuits intégrés et la synthèse de fonctions booléennes. L'architecture cible proposée est une logique régulière basée sur des PLAs et des ROMs.

La figure 2.3.26 illustre la chaîne suivie pour la génération et l'implantation automatique d'un séquenceur ou d'une fonction booléenne.

*** Les langages de description**

Plusieurs formes de description peuvent être utilisées selon le type de fonction à implanter.

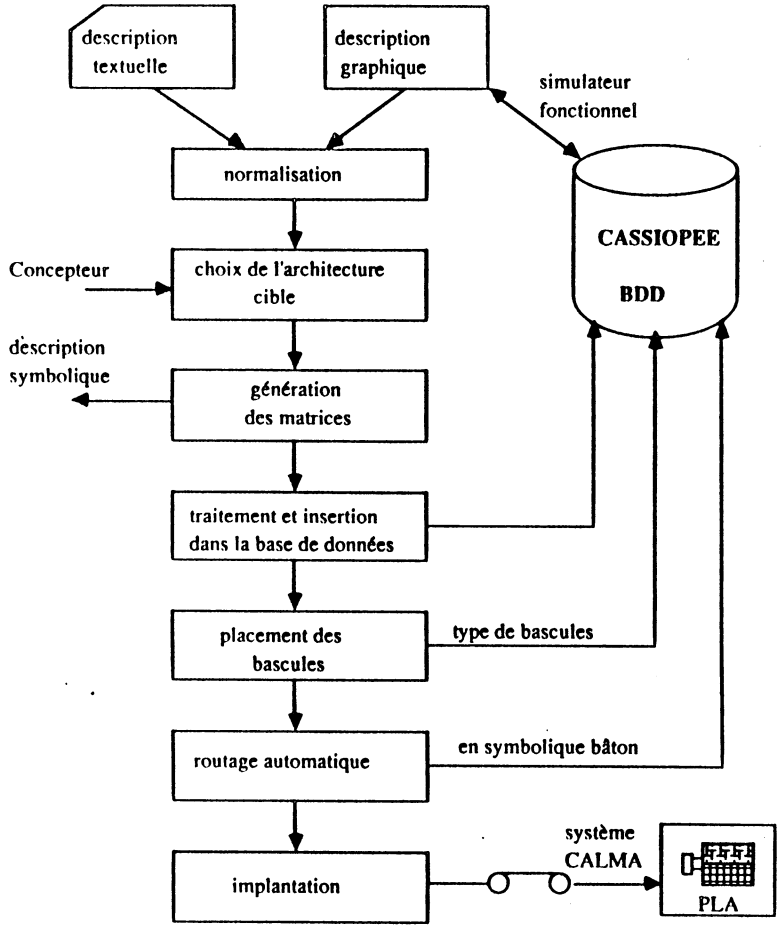


Figure 2.3.26 GASP et son environement

1.- Graphe de contrôle d'états

Le langage de description utilisé pour exprimer le séquencement (ou les différents états d'un automate) est une graphe de contrôle similaire au formalisme des réseaux de Petri.

Les éléments de base de ce modèle de description sont:

.les places.- elles correspondent aux noeuds du graphe de contrôle.

. les transitions.- elles permettent de relier 2 places ou plus.

2.- Les fonctions

Autre forme de description pour faciliter l'expression des équations logiques pour la synthèse des fonctions combinatoires, Gasp accepte le type de description suivant :

$$A= (B.C + B.C) + (D.E + F)$$

*** Les architectures cibles**

Il existe trois types d'architectures cibles basées sur des PLAS et des ROMs. La première architecture est une architecture à mono-PLA utilisée pour séquenceurs de petite et moyenne taille et permet la réalisation de séquenceurs avec un certain degré de parallélisme. Un exemple de ce type d'architecture est illustré dans la figure 2.3.27 :

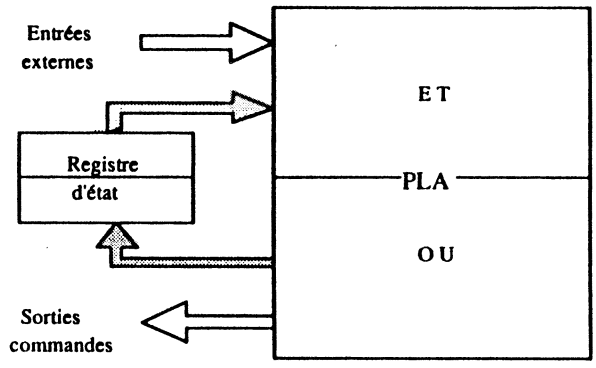


Figure 2.3.27 Schéma d'un séquenceur

Les deux architectures restantes basées sur des ROMs sont utilisées pour une approche de microprogrammation.

*** Optimisation logique**

Le système réalise l'optimisation logique du PLA à partir d'une méthode heuristique basée sur le bon codage de chaque noeud ou place au lieu de réaliser une optimisation logique sur l'ensemble des équations.

*** Placement et routages des bascules d'état**

La dernière phase de la synthèse automatique des séquenceurs est le placement et routage automatique des bascules d'état. Ces bascules permettent de maintenir et définir les états internes du PLA. Chaque bascule utilisée correspond à un bit du code associé à chacune des places du graphe de contrôle.

*** Simulation de l'architecture**

Après que le système ait traduit la description dans une architecture, il est possible de réaliser une simulation logico-fonctionnelle du résultat.

4. PRESENTATION DES BIBLIOTHEQUES DE CELLULES ET D'OPERATEURS FLEXIBLES

4 PRESENTATION DES BIBLIOTHEQUES DE CELLULES ET D'OPERATEURS FLEXIBLES

Comme il a été dit, la réalisation des opérateurs flexibles suppose l'existence d'un jeu de cellules de base.

Dans ce chapitre, nous allons présenter d'une part la bibliothèque des cellules réalisées pour la construction des opérateurs en tranches définis au § 2.3.4.2 et d'autre part la bibliothèque d'opérateurs.

4.1 *La bibliothèque de cellules*

Les objectifs poursuivis dans la réalisation de cette bibliothèque sont:

. de fournir les cellules de base nécessaires pour la construction des opérateurs en tranches,

. de montrer une technique possible de conception de cellules utilisables dans la construction de divers opérateurs,

. de servir de référence pour l'implantation de futures bibliothèques, dans le sens de la modularité et du style de conception.

En respectant la méthodologie définie au § 2.1.3 pour la réalisation des cellules, nous avons implanté une bibliothèque constituée des cellules suivantes :

. des bascules:
  . maître-esclave, BASCULED,
  . maître-esclave avec remise à zéro synchrone, BAS,
  . maître-esclave avec remise à zéro asynchrone, BASAC,

. portes logiques:
  . ou-exc OREXC,
  . égalité (non-ou exclusif) NOREXC,
  . non-ou (plusieurs entrées) NOR1, NOR2,NOR3, NOR4
  . ou, OR
  . et, AND, non-et, NAND,
  . inverseur, INV.

. des amplificateurs. :
  . amplificateurs: AMP1, AMP2,

. amplificateurs inverseurs: AMPINV1, AMPINV2,
. amplificateurs générateurs d'horloges sans recouvrement GENHOR (amplificateur compilable),

. autres cellules :
. sélecteur (multiplexeur 2 vers 1), SEL,
. cellule de contrôle, CL/LOAD,
. comparateur COMPAR, ...

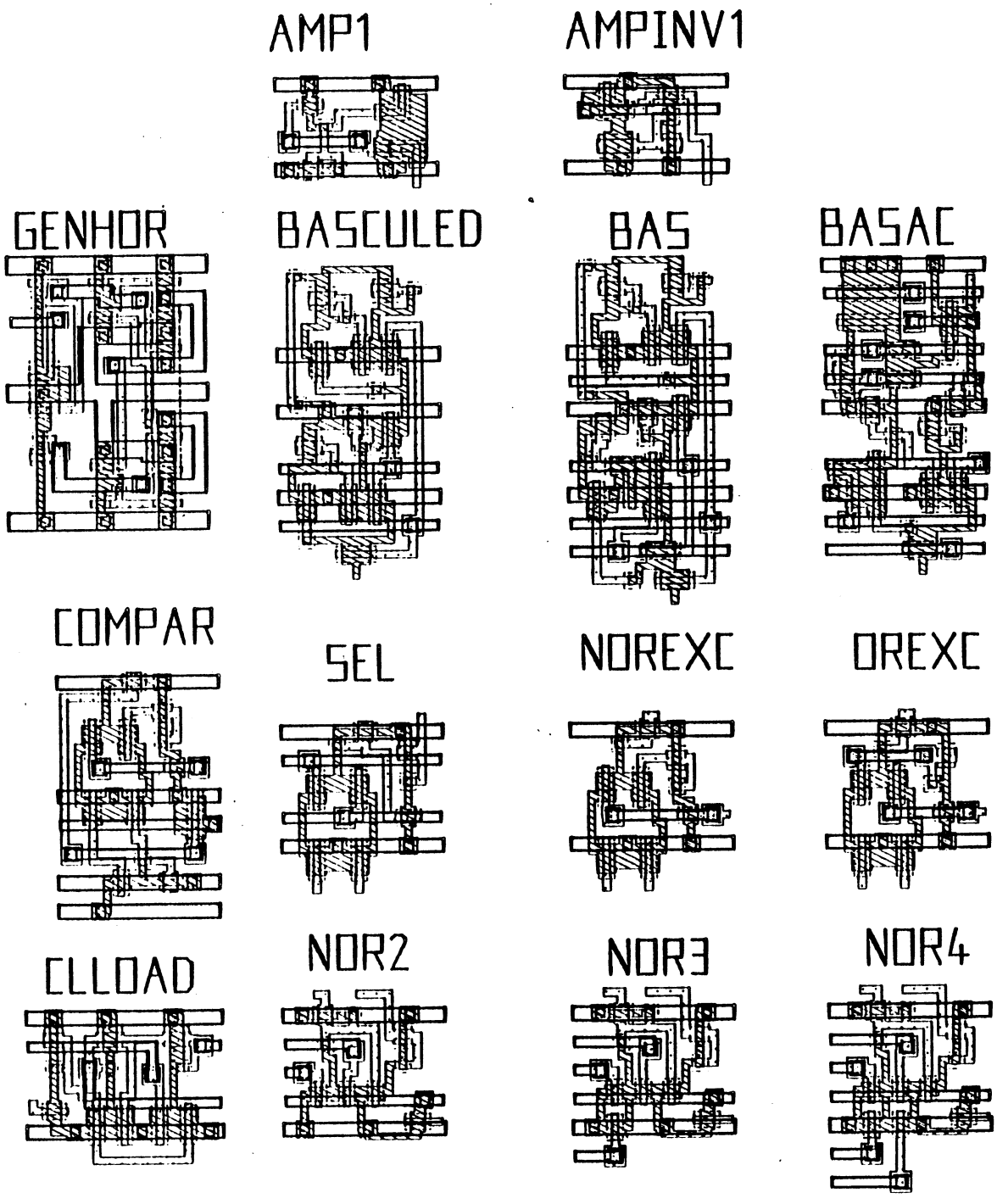


Figure 2.4.1

Les caractéristiques de ces cellules sont présentées dans l'annexe D, l'annexe C décrit les outils de CAO utilisés (CASSIOPEE) pour leur réalisation.

Cette bibliothèque a été réalisée dans l'optique d'utiliser le moins de cellules possibles et pour le plus grand nombre d'opérateurs. Quelques-unes de ces cellules sont illustrées dans la figure 2.4.1.

> **Remarque**: L'ensemble de cellules réalisées ne constitue pas une bibliothèque dans le sens d'une bibliothèque de cellules précaractérisées. Elle n'est pas complète et ne peut prétendre à elle seule être suffisante pour la conception d'un circuit quelconque.

4.1.1 *Amplificateurs compilables*

Il faut signaler que dans notre démarche pour constituer une bibliothèque de cellules, nous avons utilisé et intégré à notre bibliothèque, un jeu de cellules "compilables" réalisées par des concepteurs du CNET-Grenoble [CON-83].

Une cellule compilable est un programme qui permet la génération automatique d'une cellule à partir de paramètres dynamiques définis par l'utilisateur. Le programme agit sur une cellule réalisée originalement à la main qu'il fait varier comme une cellule à coulisse. Ainsi, à la différence d'une cellule précaractérisée, ces cellules compilables peuvent-elle faire varier leurs dimensions et leurs caractéristiques dynamiques. Un exemple de ce type de bibliothèque est proposé par [NAN-83] [DUY-83].

Dans le système Calma du CNET-Grenoble les cellules répertoriées sont :

. des amplificateurs internes pour commander des liaisons très capacitives,

. des amplificateurs d'entrée/sortie utilisés pour les plots d'entrée et les plots de sortie,

. des amplificateurs d'horloge deux phases sans recouvrement.

Les paramètres utilisés pour la génération de ces amplificateurs dits compilables sont: la capacité de charge et la fréquence de fonctionnement (le programme calcule le temps de commutation et le temps de propagation total).

Un exemple, de cette génération automatique est illustré par la figure 2.4.2, un amplificateur (générateur d'horloges deux phases sans recouvrement) généré à partir de paramètres différents.

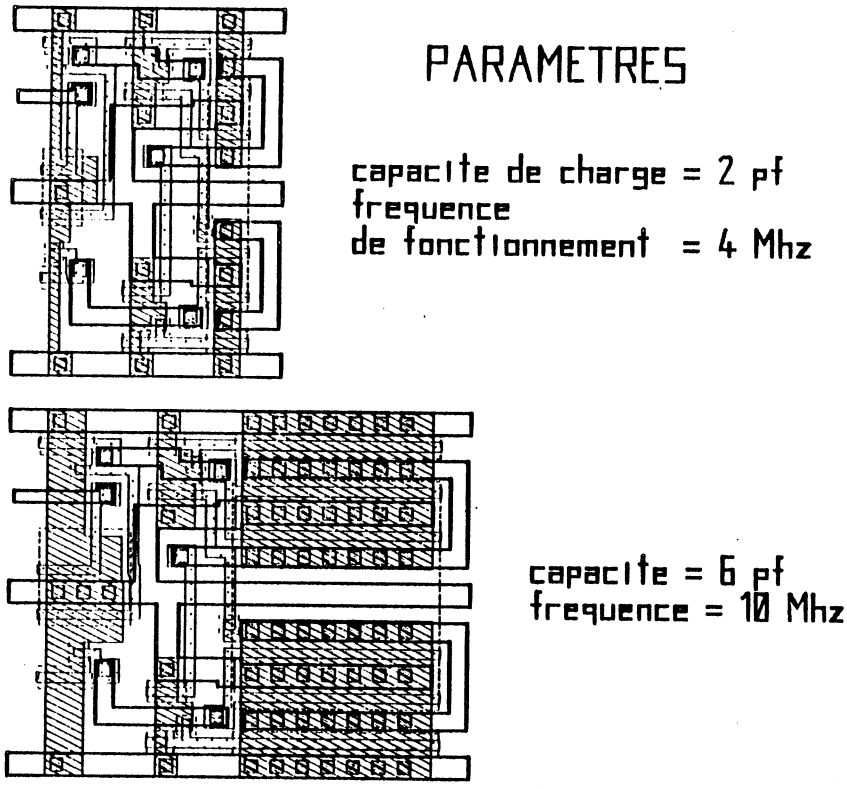


Figure 2.4.2 Générateurs d'horloges deux phases sans recouvrement

4.2 *La bibliothèque d'opérateurs*

4.2.1 Bibliothèque de fiches LOF réalisée

Chaque opérateur en tranches défini au § 2.3.2 sera généré à partir d'une fiche LOF. Ainsi, la figure 2.4.3 montre comment est constituée la bibliothèque d'opérateurs en tranches.

Plusieurs niveaux de hiérarchie peuvent être distingués :

. Le premier niveau correspond à l'interface utilisateur, présentée sous forme de menu. Elle permet la saisie des paramètres et l'activation de la (ou des) fiche(s) correspondante(s).

. Le deuxième niveau correspond aux fiches qui génèrent les opérateurs. Elles reçoivent les paramètres et utilisent les cellules nécessaires pour la construction de l'opérateur. Le résultat le fait remonter jusqu'à l'utilisateur.

La saisie des paramètres se fait en deux temps : la sélection de l'opérateur à générer, puis les paramètres qui lui sont liés.

Plusieurs résultats obtenus à partir de cette bibliothèque sont illustrés dans l'annexe E.

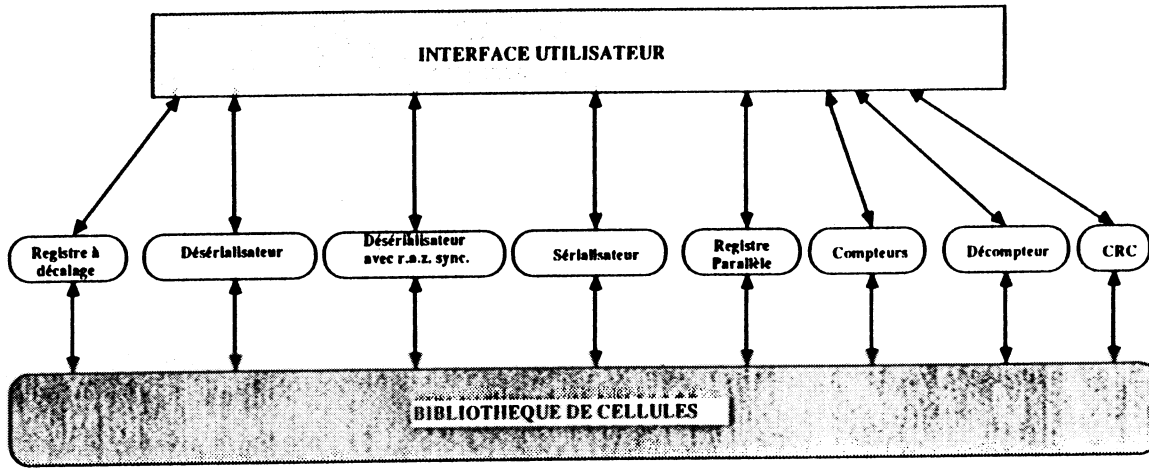


Figure 2.4.3 Organisation de la bibliothèque d'opérateurs en tranches

4.2.2 Les opérateurs programmables

Dans ce paragraphe, nous allons présenter plusieurs exemples de réalisations d'opérateurs programmables : opérateurs combinatoires et opérateurs de contrôle.

* Opérateurs combinatoires

1.- Un circuit communicateur a normalement besoin d'un dispositif qui permette la détection d'une trame circulant sur le réseau. Il réalise l'identification des délimiteurs ou fanions de début et fin de trame. Par exemple, dans le cas de la trame HDLC les deux fanions sont représentés par la même séquence : 01111110 . Dans le cas du circuit FIP-VLSI (cf 3.2), ces délimiteurs sont différents et représentés de la manière suivante:

```
Préambule 0101010101 .- indique le début d'une trame quelconque
Fin-trame XXXX101111 .- indique la fin de la trame
Syn-Nom   0101111000 .- indique qu'il s'agit d'une trame de nom
Syn-Don   0101000111 .- indique qu'il s'agit d'une trame de données
```

Pour la génération de l'opérateur combinatoire qui permet l'identification de ces 4 séquences, il suffit donc de définir les équations booléennes correspondantes:

```
Préambule :=  ^A.B.^C.D.^E.F.^G.H.^I.J      (^=barre)
Fin-trame :=  X.X.X.X.E.^F.G.H.I.J
Syn-Nom   :=  ^A.B.^C.D.E.F.G.^H.^I.^J
Syn-Don   :=  ^A.B.^C.D.^E.^F.^G.H.I.J
```

A partir de ces équations, le système GASP génère automatiquement en deux temps l'opérateur combinatoire (en forme de PLA) "identification ou égalité". D'abord la description logique du PLA (figure 2.4.4 (a)) et puis son implantation, (figure 2.4.4 (b)).

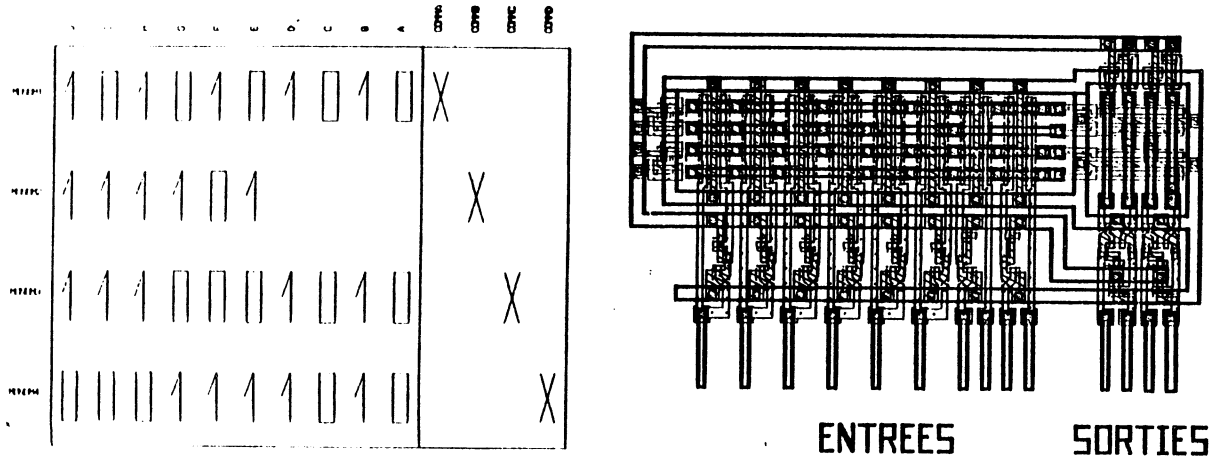


Figure 2.4.4 Opérateur combinatoire d'identification.

2.- Un autre exemple d'opérateur combinatoire est la réalisation d'encodeurs. Par exemple dans le circuit FIP-VLSI, il est nécessaire de générer une séquence d'initialisation de 4 mots de 16 bits (cf tableau 3.2.1). Cette séquence est générée à l'aide d'un encodeur et l'utilisation de deux signaux d'entrée i0 et il. Les équations logiques correspondantes sont les suivantes:

```
s0:= "0"                        s8:= ^il.^i0
s1:= il.i0                      s9:= ^il.^i0 + il.i0
s2:= il.^i0                    s10:= ^il.^i0 + il.^i0
s3:= il.^i0 + il.i0            s11:= ^il.^i0 + ^il.i0
```

```
s4:= ^il.i0                          s12:= ^il.^i0 + ^il.i0
s5:= ^il.i0 + il.i0                  s13:= ^il.^i0 + ^il.i0 + il.i0
s6:= ^il.i0 + il.^i0                 s14:= ^il.^i0 + ^il.i0 + il.^i0
s7:= ^il.i0 + il.^i0 + il.i0         s15:= "1"
```

De la même manière que dans l'exemple précédent, GASP effectue une minimisation logique sur les équations, génère la description logique du PLA correspondant (figure 2.4.5 (a)) et puis son implantation, (figure 2.4.5 (b)).

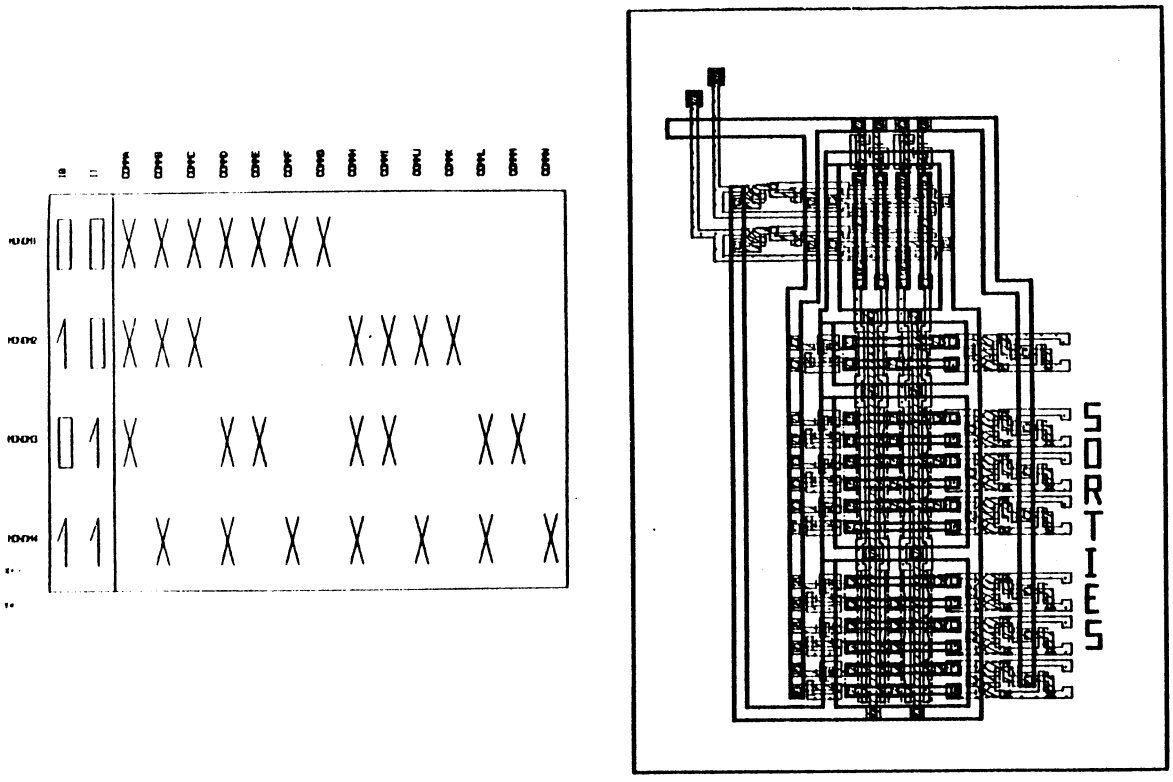


Figure 2.4.5 Opérateur combinatoire encodeur.

* **Opérateurs de contrôle**

La partie contrôle peut être implantée à l'aide d'un ou plusieurs séquenceurs et automates. Ici, nous allons présenter l'implantation d'un séquenceur qui réalise la gestion d'une interface type microprocesseur entre un circuit communicateur FIP et une station. Cet exemple correspond à une proposition effectuée dans les premières spécifications du circuit FIP 001.

Le fonctionnement du séquenceur est décrit par le graphe d'état de la figure 2.4.6. GASP génère automatiquement, à partir de la description du graphe, le séquenceur ou l'opérateur de contrôle correspondant, figure 2.4.7.

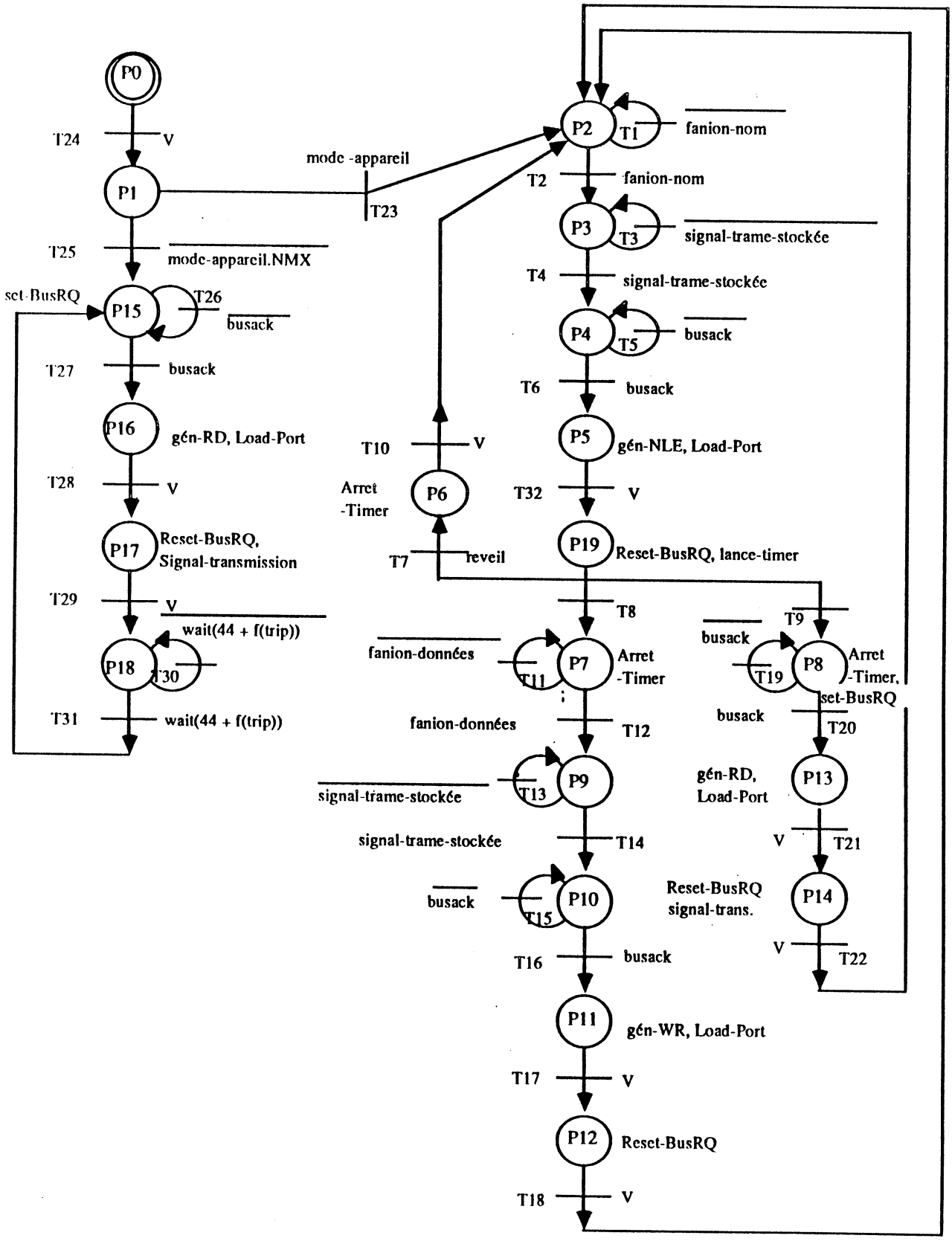

P0
T24
V
P1
mode -appareil
T23
P2
T1
fanion-nom
T2
fanion-nom
P3
T3
signal-trame-stockée
T4
signal-trame-stockée
P4
T5
busack
T6
busack
P5
gén-NLE, Load-Port
T32
V
P19
Reset-BusRQ, lance-timer
T25
mode-appareil.NMX
set-BusRQ
P15
T26
busack
T27
busack
P16
gén-RD, Load-Port
T28
V
P17
Reset-BusRQ,
Signal-transmission
T29
V
wait(44 + f(trip))
P18
T30
T31
wait(44 + f(trip))
T10
V
Arret
-Timer
P6
T7
reveil
T8
T9
busack
fanion-données
P7
Arret
-Timer
T11
P8
Arret
-Timer,
set-BusRQ
T19
busack
T20
fanion-données
T12
P13
gén-RD,
Load-Port
P9
T13
signal-trame-stockée
signal-trame-stockée
T14
V
T21
P14
V
T22
busack
P10
T15
Reset-BusRQ
signal-trans.
T16
busack
P11
gén-WR, Load-Port
T17
V
P12
Reset-BusRQ
T18
V

Figure 2.4.6

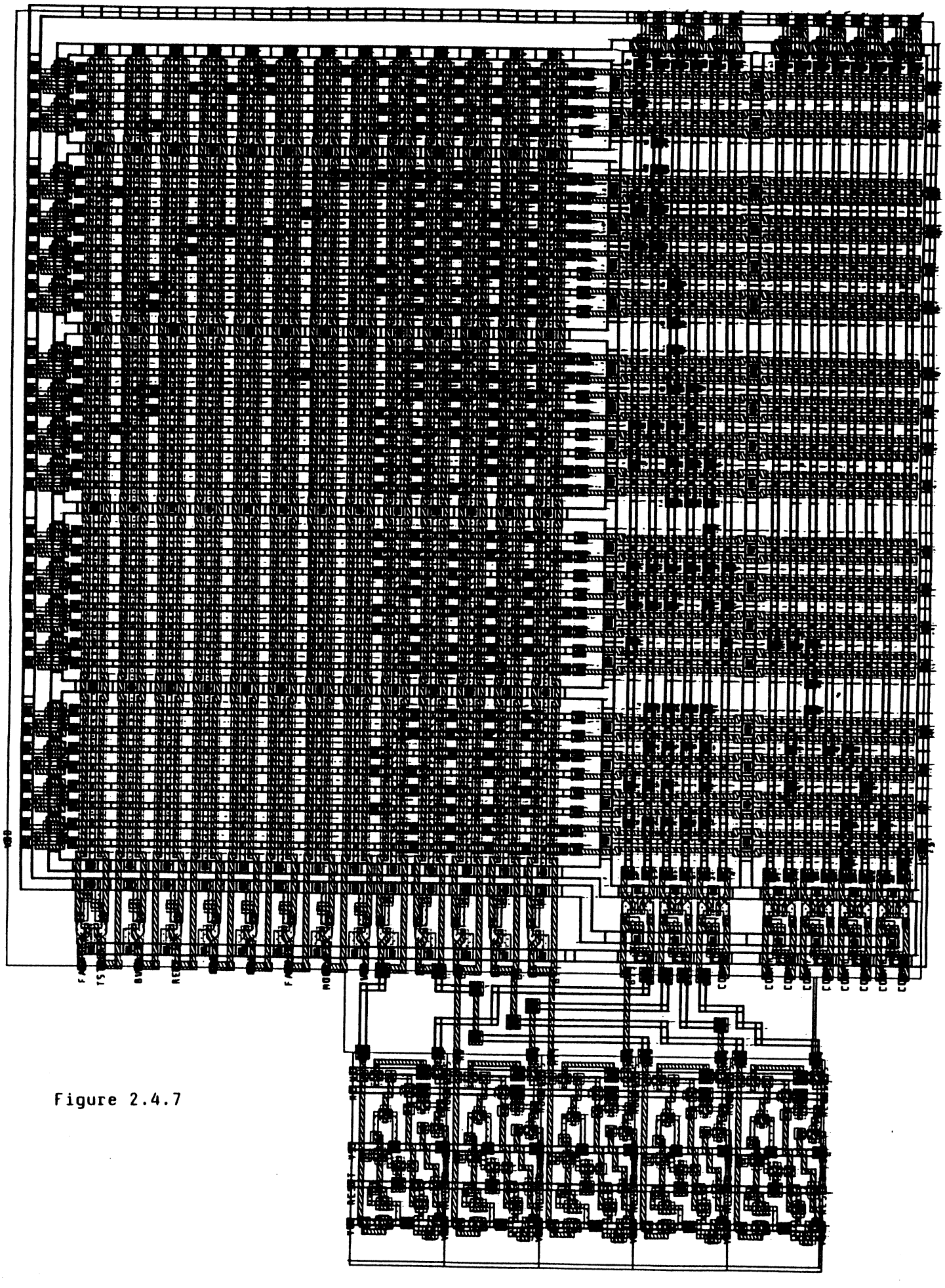


Figure 2.4.7

PARTIE 3

REALISATION

SPECIFICATIONS ET CONCEPTION D'UN CIRCUIT COMMUNICATEUR DESTINE A UN RESEAU LOCAL INDUSTRIEL TEMPS REEL

Cette partie de la thèse permet de présenter les spécifications d'un circuit intégré à haute densité d'intégration [DAN-86] [DIA-86], destiné à connecter des capteurs simples ou intelligents au réseau local industriel temps réel FIP. Développé dans le cadre des travaux qui sont connexes au projet FIP sur le plan national français, et en avance de phase par rapport aux travaux industriels, le travail décrit dans cette partie vise à démontrer la faisabilité d'un certain nombre de propositions concernant la méthode d'adressage utilisée, l'interface pour les stations capteurs, l'initialisation du composant, ... Par ailleurs, il a permis la mise au point d'une méthodologie de conception de circuit VLSI destiné à la communication pour les niveaux bas du modèle OSI. Celle-ci est décrite dans la deuxième partie.

1. INTRODUCTION

1 INTRODUCTION

Dans le domaine du contrôle-commande et de la productique, les besoins de communication sont multiples. Il existe beaucoup de réseaux locaux dits industriels (RLI), [LEL-83] [LET-84] [STR-85] [INT-85]. En fait, ces RLI correspondent à différents niveaux de réseaux utilisés dans ce domaine. La décomposition selon les quatre niveaux suivants est généralement la plus acceptée : à l'intérieur d'un robot ou d'une machine, entre les différents machines - robots - capteurs - actionneurs, pour les systèmes de supervision et entre différents systèmes.

Pour les deux premiers niveaux de réseaux locaux industriels (qu'ils soient militaires ou civils), il est nécessaire que le coût de connexion des stations au réseau soit proportionel au coût des stations elles-mêmes (coût qui peut être très bas dans le cas d'un capteur de position par exemple) et de plus que les exigences temps réel des applications soient résolues par le réseau et ses protocoles. Les contraintes économiques, la taille et la technologie des stations induisent la réalisation du communicateur en circuit intégré; le nombre d'exemplaires de stations que les industriels espèrent vendre au niveau national ou européen d'une part, les conséquences des spécifications au niveau technologique d'autre part, suggèrent une conception de ces communicateurs en circuits à la demande optimisés.

Cadre du travail.

Dans le cadre du groupe de travail RLI du Ministère de l'Industrie et de la Recherche, un système de transmission série, dit FIP, pour des échanges d'informations entre des capteurs, actionneurs, et automates réflexes a été proposé [FIP-84].

FIP réalise, en fait, le niveau temps réel dit communication "informations-processus", qui fait partie des deux premiers niveaux mentionnés ci-dessus.

Un exemple d'application utilisant FIP est illustré par la figure 3.1.1.

Les objectifs de FIP sont les suivants:

. coût réduit de câblage processus-équipements d'automatisme: choix d'une liaison série multi-point,

.faible coût de connexion des équipements à FIP: réalisation d'un composant FIP intégré,

. garantie du temps de transfert d'informations: choix d'un mode de fonctionnement en "étoile logique" entre maître-esclaves de scrutation périodique des stations,

. adaptation des débits d'informations aux automatismes et aux équipements,

. disponibilité, sécurité.

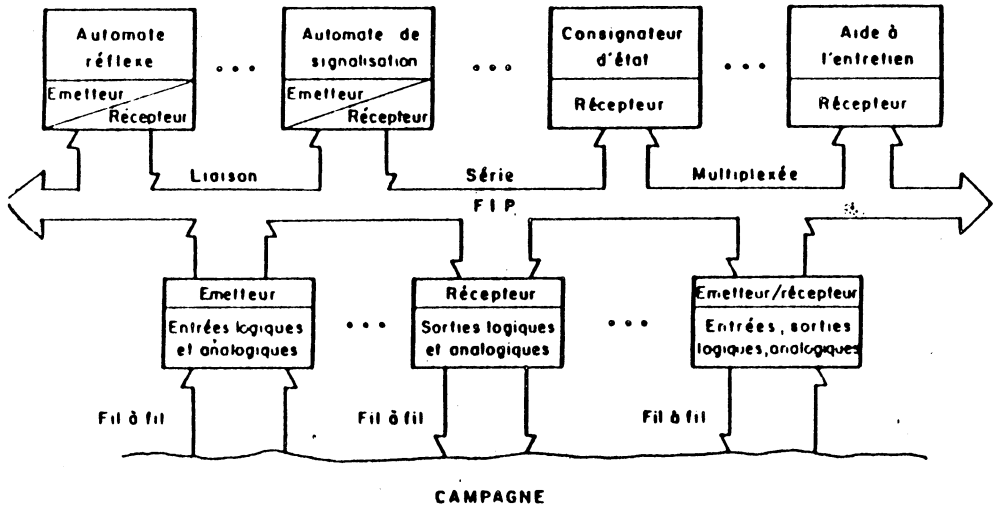


Figure 3.1.1 Exemple d'application utilisant FIP

Du fait de la connexion de différentes familles d'équipements, des automates aux capteurs, il existe différents développements de familles de composant autour du concept FIP:

. FIP 001 pour les automates, développé en prédiffusé par la CSEE,
. FIP 002 à venir pour les capteurs, dont un groupe de travail est actuellement en cours pour définir le cahier des charges.

Pour démontrer la faisabilité d'un certain nombre d'idées et pour proposer des solutions dans le cadre du travail sur le composant FIP Capteurs, nous avons conçu, en avance de phase, un circuit intégré VLSI FIP Capteurs.

Dans la suite du chapitre, ce circuit intégré sera mentionné sous le nom du circuit FIP-VLSI, pour des raisons de commodité de langage.

L'autre aspect développé dans la deuxième partie de cette section concerne la mise en oeuvre de la méthodologie présentée dans cette thèse pour la conception du circuit communicateur FIP-VLSI.

2. SPECIFICATIONS

DU CIRCUIT COMMUNICATEUR

2 SPECIFICATIONS DU COMPOSANT FIP-VLSI

2.1 *Caractéristiques générales du système de communication FIP*

L'architecture de FIP est une architecture en bus série à accès centralisé. Cette architecture, illustrée dans la figure 3.2.1, est constituée d'une station maître qui gère l'accès sur le bus série, et des stations esclaves ou secondaires qui réalisent entre elles les échanges d'informations.

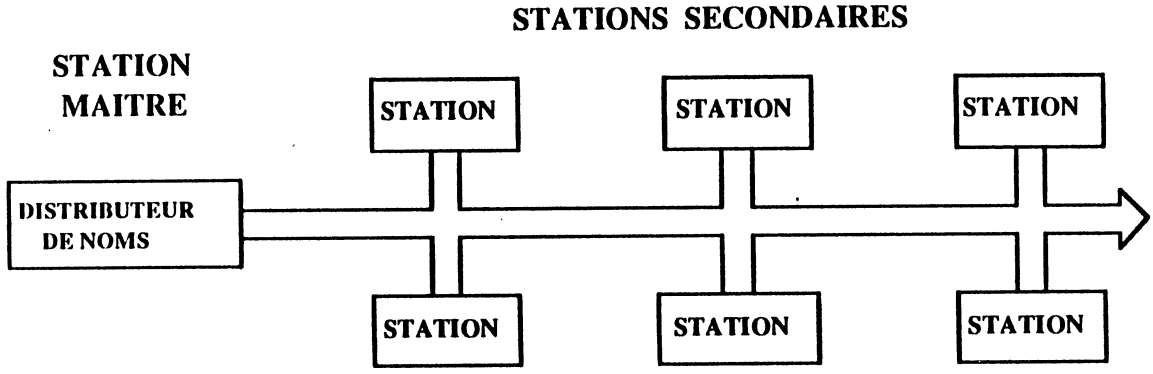


Figure 3.2.1 Architecture FIP.

2.1.1 Le réseau FIP

Les principales caractéristiques de ce réseau sont les suivantes:

. vitesse de transmission : entre 1 Mbps et 10 Mbps selon le type de réseau
. type de transmission : en bande de base,
. type de codage des informations : Manchester,
. interface avec les outils de ligne : indépendante du choix du medium.

2.1.2 Le protocole de communication

Un échange d'information sur le support de transmission se traduit par la succession d'une trame de NOM suivie d'une trame de DONNEES. La trame NOM ne contient qu'une seule adresse, selon la méthode d'adressage dit d'adressage source, concept qui est aussi défendu par exemple par le réseau PROCONTROL-P [PAN-84]. La trame de données est séparée de la trame de nom par un intervalle de temps borné et est de taille variable.

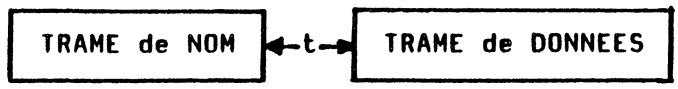


Figure 3.2.2 Enchaînement de la trame nom et de la trame données

La station maître propose une trame de nom, qu'elle prend dans une table de scrutation périodique, dans laquelle à chaque nom est associée une période propre. Une station esclave, qui "voit" passer le nom, se reconnaît comme émettrice et va émettre une trame de données. Les stations esclaves, qui se reconnaissent comme réceptrices sur le même nom, attendent de lire l'information dans la trame de données qui suit. Cette technique d'accès au medium est illustrée par la figure 3.2.3.

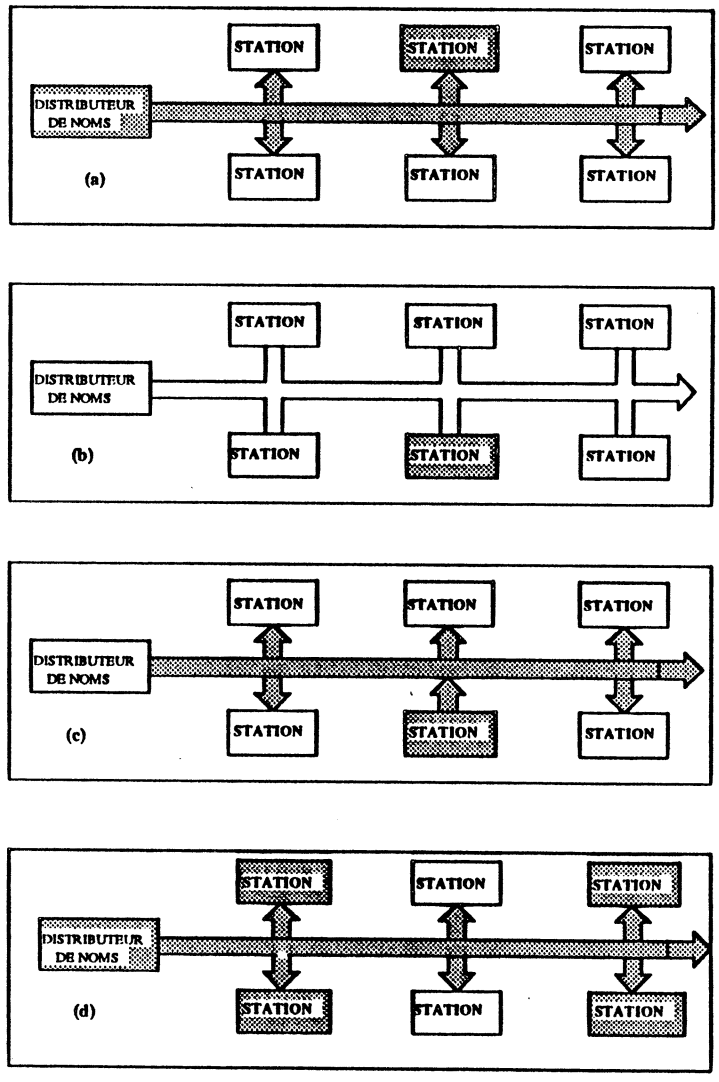


Figure 3.2.3 Technique d'accès au medium
a) Diffusion du nom, b) Identification d'un nom par un émetteur, c) Emission de la trame de données par l'émetteur, d) les stations réceptrices lisent l'information.

Du fait des transmissions périodiques, il n'y a pas un besoin strict de trame d'acquittement ou de demande de ré-émission.

2.1.3 Caractéristiques générales du circuit FIP

Le circuit FIP est un composant qui doit assurer l'émission/réception des informations en provenance ou à destination du support de communication selon les critères définis par le système de communication FIP. Les principales caractéristiques que celui-ci doit remplir, en plus de celles définies ci-dessus, sont:

. vitesse de transmission : 2.5 Mbit/s
. modes de fonctionnement : distributeur, appareil et observateur
. procédure de contrôle d'accès : centralisée selon protocole FIP
. taille de trames: variable
. interface bus local: parallèle 16 bits.

La figure 3.2.4 permet de fixer les idées sur la terminologie employée et le découpage du couple composant FIP-station.

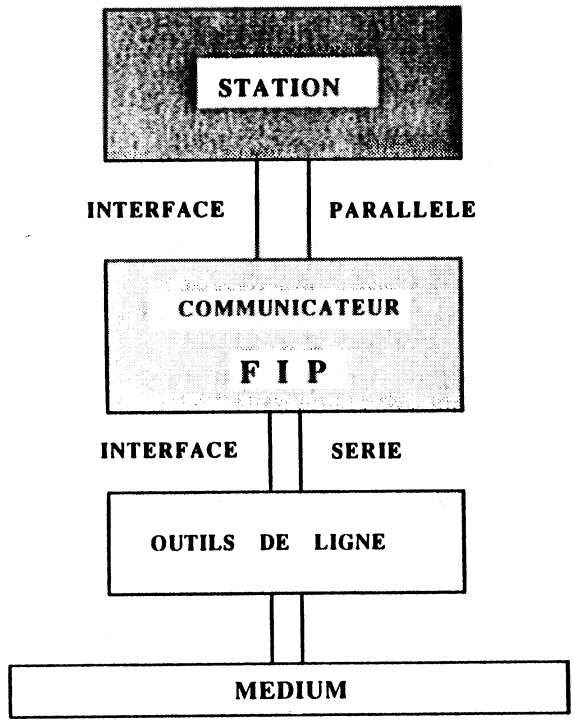


Figure 3.2.4

Différents aspects sont discutés ci-après : la définition de la structure des trames et la méthode d'adressage, la définition des modes de fonctionnement, l'interface avec la station, la reconnaissance de l'adresse, l'interface avec les outils de connexion au médium, et l'initialisation du circuit. De cette discussion, nous allons tirer les conséquences sur nos propres choix de spécification.

2.2 *Méthode d'adressage et structure des trames*

2.2.1 Méthode d'adressage

L'adressage source, qui est l'unique mode d'adressage proposé pour le système de communication FIP, permet le dialogue entre stations en fonction d'une configuration pré-établie statiquement à l'initialisation du système. Soulignons les particularités de ce type d'adressage [GUE-84].

.les receveurs ne sont pas connus a priori de l'émetteur,

.la possibilité existe pour le configurateur du système d'attacher statiquement un récepteur à un seul émetteur et d'éviter ainsi l'interférence entre plusieurs émetteurs au niveau d'un seul récepteur.

Admettons que ce mode d'adressage convienne pour les systèmes de contrôle-commande en phase de fonctionnement. Cependant, nous proposons que notre composant FIP-VLSI reconnaisse deux autres modes d'adressage:

. l'adressage source-destination, qui permet un fonctionnement plus souple de la communication car, durant le cycle de vie du couple composant FIP-station capteur (ou actionneur), il existe des étapes telles que le réglage à l'usine, les paramétrages des stations sur site et la maintenance où l'existence d'un mode d'adressage explicite est souhaitable;

. l'adressage sous-diffusion : ce mode d'adressage est utile surtout quand il s'agit de faire une action de synchronisation explicite d'un sous-ensemble de stations, action non forcément périodique. A ce propos, mentionnons l'existence possible d'une plage réservée aux actions non périodiques à l'intérieur d'une table de scrutation parcourue répétitivement par le maître.

2.2.2 Formats des trames

Dans les spécifications du circuit FIP-VLSI, nous proposons les trois modes d'adressage, définis dans la trame de nom et reconnus automatiquement par le circuit. Par ailleurs, la trame données est de taille variable.

Format de la trame de NOM

adressage source	adressage source-destinataire	adressage sous-diffusion
préambule (6)	préambule (6)	préambule (6)
synchro-NOM (3)	synchro-NOM (3)	synchro-NOM (3)
adr-émetteur (16)	adr-émetteur (16)	adr-émetteur (16)
crc (16)	adr-récepteur (16)	adr-récepteur 1 (16)
fin-trame (3)	crc (16)	.....
	fin-trame (3)	adr-récepteur i (16)
		crc (16)
		fin-trame (3)

Format de la trame de DONNEES

simple	multiple
préambule (16)	préambule (6)
synchro-DONNEES (3)	synchro-DONNEES (3)
donnée (16)	donnée 1 (16)
crc (16)	.....
fin-trame (3)	donnée i (16)
	crc (16)
	fin-trame (3)

2.2.2.1 Définition des champs qui constituent les trames

* PREAMBULE.- est une séquence de 6 éléments binaires de valeur logique "1". Il permet au circuit de génération d'horloge du récepteur de se synchroniser et d'être dans un état stable lors de l'arrivée de la trame .

* SYN_NOM.- détermine le début d'une trame de nom. La longueur de ce champ est de trois éléments binaires; elle est définie par la séquence de "violation de code" montrée dans la figure 3.2.5. T représente la période correspondant à un élément binaire.

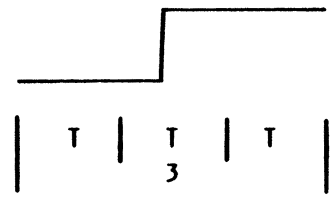


Figure 3.2.5 Synchronisation de NOM (SYN_NOM).

* SYN_DON.- détermine le début d'une trame de données (figure 3.2.6); elle est le complément de la synchro_nom,

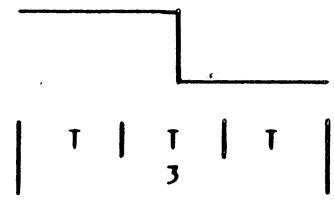


Figure 3.2.6 Synchronisation de DONNEES (SYN_DON).

* ADRESSES.-Ce champ peut être formé d'une ou plusieurs adresses. La première adresse détermine toujours une station émettrice et une ou plusieurs stations réceptrices. Les adresses suivantes permettent de définir uniquement des stations réceptrices.

Une adresse est un mot de 16 bits qui permet de répertorier 65.536 entités différentes. Dans ce champ chaque mot est précédé d'un bit de synchronisation de valeur "1". Les mots sont transmis en commençant par le bit de poids le plus faible, (figure 3.2.7).

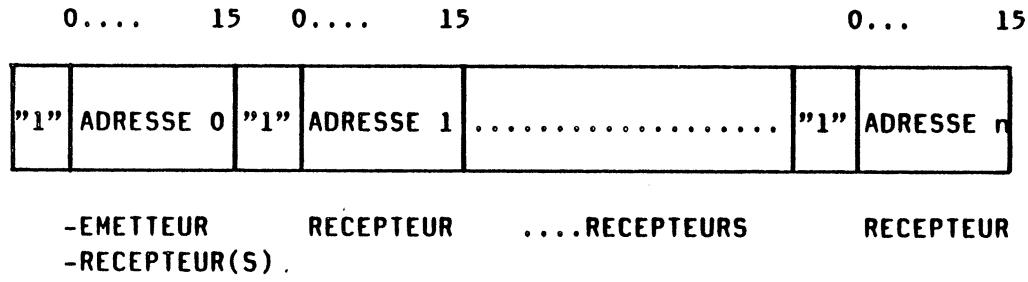


Figure 3.2.7 Champ d'ADRESSES.

La première adresse du champ a une signification particulière car elle permet de déterminer une station émettrice et une ou plusieurs stations réceptrices.

Dans le cas où existent plusieurs stations portant la même adresse:

- une station émettrice est définie lorsque les 16 bits constituant l'adresse correspondent à l'adresse de la station, (figure 3.2.8).

- une station réceptrice est définie si l'adresse de la station et l'adresse transmise diffèrent uniquement du bit de poids le plus faible, (figure 3.2.8).

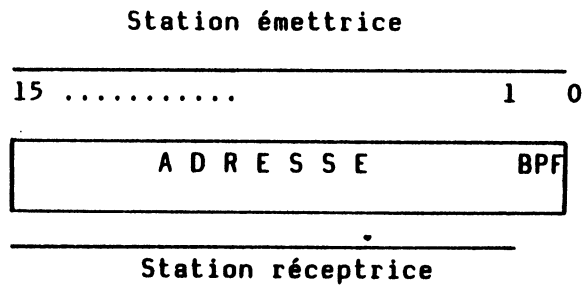


Figure 3.2.8 Représentation de la première adresse.

L'adresse d'une station est déterminée par son adresse physique, définie à l'initialisation de la station.

Il est vrai que cette façon de désigner les stations est "statique" car le dialogue entre stations est fait en fonction d'une configuration préétablie. Les mots suivant le premier mot permettent d'ajouter ou de nommer d'autre(s) récepteur(s) pour les faire participer à la communication; dans ce cas, les 16 bits qui constituent le mot doivent correspondre entièrement au nom de la station.

* DONNEES.- Ce champ contient les données d'information, il est de taille variable et peut contenir une ou plusieurs données. Par définition du système, le nombre d'informations transmises doit être très réduit, voire même à une seule donnée dans la plupart des applications.

Chaque donnée est un mot de 16 bits, précédé d'un bit de synchronisation de valeur "1". Les mots sont transmis en commençant par le bit de poids le plus faible, (figure 3.2.9).

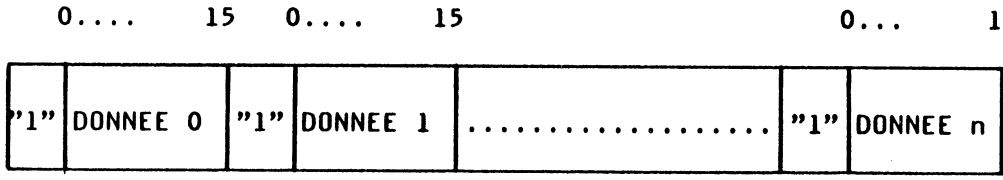


Figure 3.2.9. Champ d'INFORMATION

* FCS.- la séquence de contrôle de trame est une valeur de 16 éléments binaires obtenus par l'application, sur les champs de nom et d'information des trames de nom ou de donnée respectivement, d'un code de redondance cyclique (CRC), la méthode de conception du générateur de cette valeur a été exposée au § 2.3.4.

L'encodage est défini par le polynôme générateur

$$G(x) = X^{16} + X^{12} + X^{5} + 1$$

Le FCS est utilisé pour assurer l'intégrité des informations contenues dans les trames contre de possibles erreurs dûes à la transmission.

La transmission du FCS commence par le coefficient du terme le plus élevé et précédé d'un bit à "1" de synchronisation.

* FIN_TRAME.- c'est la séquence indiquant la terminaison d'une trame. Sa longueur est de 3 éléments binaires. Le premier est un "0" logique; les deux restants sont définis par une violation de code représentée par un niveau "haut" d'une durée de deux périodes,

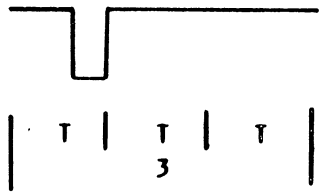


Figure 3.2.10. FIN_TRAME

Transparence.

La transparence des trames est assurée par l'utilisation d'une violation de code sur les délimiteurs de trame: SYN_NOM, SYN_DON et FIN_TRAME. Ces trois champs sont équivalents aux fanions de début et de fin de trame utilisés dans les trames des protocoles type HDLC, SDLC, etc. Dans ces protocoles, pour assurer la transparence il faut des circuits d'insertion/suppression de zéros, ce qui n'est pas nécessaire dans notre cas vu les délimiteurs que l'on utilise.

2.3 *Définition des modes de fonctionnement du circuit FIP-VLSI*

Dans le système de communication FIP, on peut distinguer différents modes de fonctionnement des stations qui constituent le réseau. Ces modes peuvent se résumer comme suit:

. mode distributeur, il est lié au fonctionnement de la station distributrice de noms.

. mode appareil, ce mode est définie pour les stations secondaires. Dans ce cas, le circuit peut être lié à une station émettrice, à une station réceptrice ou à une station émettrice/réceptrice.

. mode observateur, il permet d'introduire l'utilisation d'une station spécialisée pour surveiller le trafic sur le réseau. Cette station pourra déterminer le bon fonctionnement du système de communication en fonction des informations issues du circuit FIP-VLSI.

Donc, pour des raisons de souplesse du système et d'économie, il est souhaitable que dans un seul circuit puissent être intégrés ces 3 modes de fonctionnement.

2.4 *Interface côté station*

Nous assumons que l'information manipulée à l'interface est de caractère digital. Les conversions a/d et d/a sont extérieures au circuit.

2.4.1 Critères de choix du type d'interface

Le circuit FIP-VLSI doit être un circuit qui sera connecté à une grande diversité d'équipements. . Ceux-ci peuvent aller des capteurs (ou actionneurs) intelligents [DUB-85] capables de gérer jusqu'à 16 voies d'entrée ou sortie avec un protocole de dialogue de type poignée de main, jusqu'à des équipements très simples sans aucune intelligence; l'exemple caractéristique d'un tel équipement est une voie TOR, individualisée. Par ailleurs, la connexion à un équipement tel un automate bâti sur une architecture micro-processeur 8 bits doit être possible, avec des voies d'entrée-sortie (8) différenciées.

Cette diversité impose des contraintes importantes dans les spécifications de l'interface côté station. Ainsi, a-t-il fallu prendre en considération les critères suivants:

. paramétrisation possible des voies, en entrée et en sortie, qui a aussi pour conséquence la possibilité de définir des voies de données et des voies de commande,

. indépendance du choix de l'interface vis-à-vis de l'équipement à connecter,

. faible coût de connexion de la station au composant FIP-VLSI.

2.4.2 Choix de l'Interface

De l'analyse des sortes d'interfaces possibles à utiliser avec les divers équipements qui constituent les stations, on peut retenir les deux qui s'avèrent les plus adéquates: une interface microprocesseur et une interface type PIA.

spécifications du communicateur

*Interface Microprocesseur

L'interface type microprocesseur, vis-à-vis des critères annoncés, présente plusieurs inconvénients :

. inexistence d'une interface normalisée vis-à-vis des équipements à connecter, ceci impose l'adaptation des différents signaux,

. dans le cas des équipements simples, sans "intelligence", il faut soit utiliser un microprocesseur qui adapte les signaux issus du (ou vers le) circuit FIP pour établir la communication avec l'équipement, soit adapter ces signaux à l'aide de logique câblée externe.

Dans ces deux cas, les inconvénients majeurs sont une augmentation de la complexité de l'implémentation car il y aura autant de montages que d'équipements différents, et une augmentation dans le coût de la connexion.

*Interface Type PIA

L'autre solution est la réalisation d'une interface beaucoup plus simple et à la fois plus souple: une interface du type PIA. Elle peut être compatible avec presque tous les types de microprocesseurs, dans le cas des applications "intelligentes", et peut être utilisée très facilement dans le cas des applications "non intelligentes". Par exemple, l'utilisation des simples tampons suffit à contrôler des entrées/sorties du type tout ou rien ou du type commande.

De ces deux solutions possibles, et en considérant les critères mentionnés ci-dessus, c'est la deuxième qui a été retenue. Elle permet, entre autres, de favoriser les applications "non intelligentes".

L'interface choisie pour le composant FIP-VLSI est une interface comportant:

. 16 voies de données bidirectionnelles; ces voies sont programmables en deux groupes : un groupe de n voies programmées en entrée et un groupe de p voies programmées en sortie, sachant que n+p=16, avec 0=< n,p =< 16;

. 4 signaux de contrôle des échanges des informations; deux utilisés pour l'écriture (émission), et les deux autres pour la lecture (réception); ces signaux sont du type donnée_prête, donnée_prise,

. 2 signaux en entrée du circuit FIP-VLSI pour programmer n et p: la programmation sera effectuée à l'initialisation du circuit (voir 2.5).

. un signal d'erreur issu du circuit pour signaler une anomalie dans le fonctionnement.

2.4.3 Exemple d'utilisation de l'interface

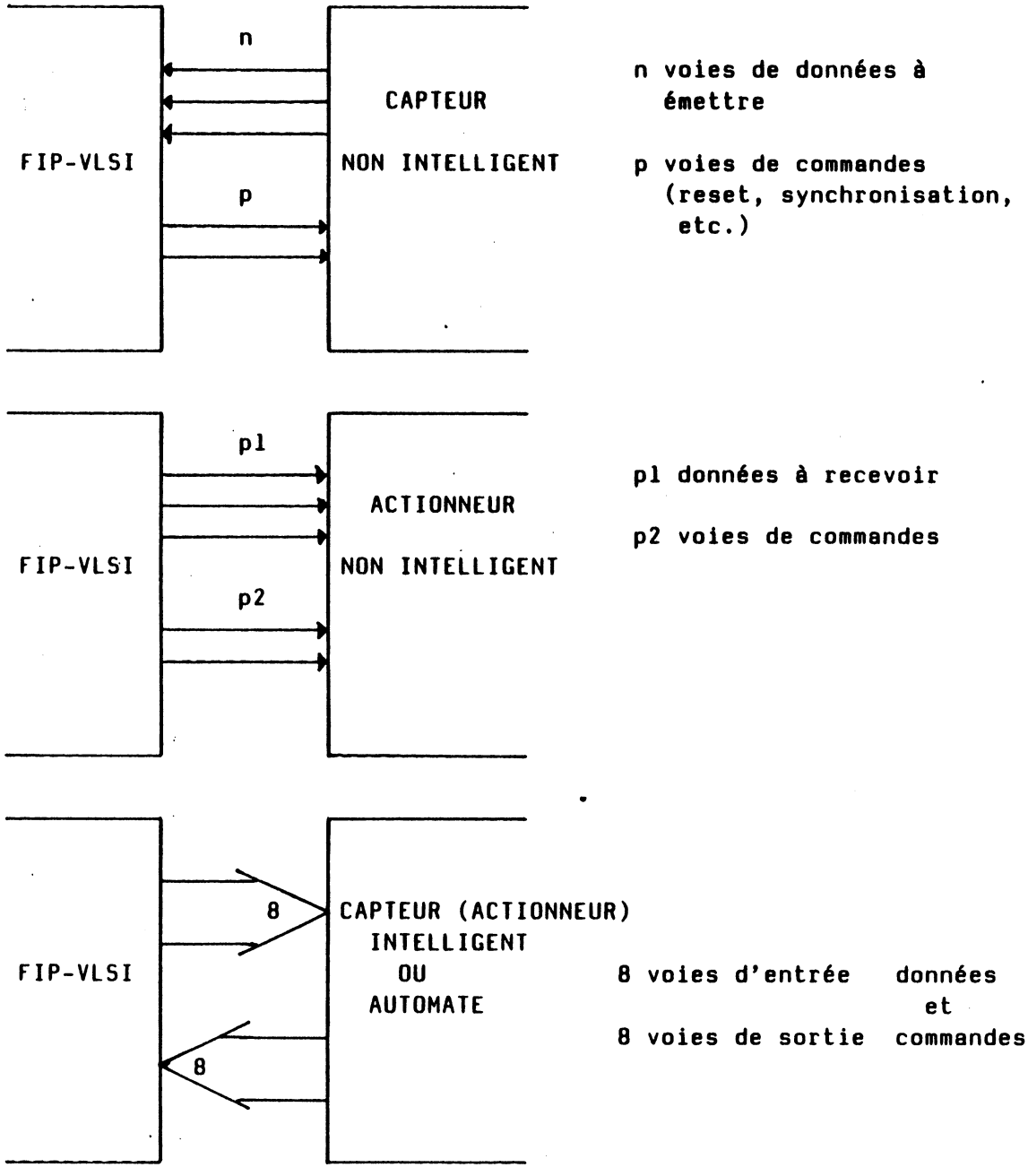


Figure 3.2.11 Exemples d'utilisation de l'interface côté station

Remarques: le fonctionnement de donnée_prête, donnée_prise permet, même en l'absence des signaux fin_de_donnée, d'effectuer plusieurs échanges à la cadence de l'émission et de la réception sur la ligne, ceci, en supposant que le signal donnée_prête est toujours fourni à temps par son émetteur à l'interface et que son absence signifie fin_de_donnée.

2.5 *Reconnaissance de l'adresse*

2.5.1 Techniques proposées

Une station sur le réseau sera sélectionnée pour participer à un échange d'informations lorsqu'elle reconnaît son adresse à l'intérieur de la trame de nom. Deux techniques peuvent être utilisées: reconnaissance de l'adresse à l'intérieur de la station ou à l'intérieur du circuit.

*Reconnaissance de l'adresse dans la station.

Cette méthode peut être décrite brièvement de la manière suivante: lorsque le circuit FIP-VLSI détecte une trame de nom, il extrait l'adresse qui la constitue et la présente à la station. La station la compare à l'adresse ou aux adresses qui lui ont été attribuées. Si le résultat de cette comparaison est positif, elle le fait savoir au circuit en lui indiquant le rôle qu'elle va jouer dans la communication: station émettrice ou réceptrice.

Si la station est émettrice, elle propose les données à transmettre. Si elle est réceptrice, elle demande la réception de la trame de données suivante. Si la station n'a pas reconnu l'adresse, elle demande au circuit la réception d'une nouvelle trame de nom.

*Reconnaissance de l'adresse à l'intérieur du circuit.

Cette méthode demande la programmation des adresses de la station à l'intérieur du circuit. Chaque fois qu'une trame de nom arrive, le circuit doit faire la comparaison entre l'adresse (ou les adresses) contenue(s) dans la trame et l'adresse ou (les adresses) définie(s) dans le circuit. En cas d'égalité le circuit pourra se reconnaître comme station émettrice ou comme réceptrice.

S'il s'agit d'une station émettrice; le circuit prend les données tenues prêtes par la station, données qui seront transmises dans la trame de données. S'il s'agit d'une station réceptrice, le circuit attend l'arrivée de la trame de données pour transférer les informations vers la station. Si aucune égalité n'a eu lieu, le circuit attend l'arrivée d'une nouvelle trame de nom sans rien faire savoir à la station de la dernière communication sur le réseau.

2.5.2 Avantages et inconvénients de chaque technique

L'analyse des avantages et des inconvénients de chacune de ces méthodes, peut nous permettre d'en tirer des conclusions pour faire le choix le mieux adapté.

Les avantages de la méthode de reconnaissance de l'adresse à l'intérieur de la station sont les suivants:

. souplesse pour définir l'adresse de la station,

. possibilité de définir une table d'adresses; celle-ci permet de définir plusieurs voies logiques de communication (couples émetteur-récepteur qui consomment chacun une adresse) dans lesquelles la station est impliquée. Cette possibilité peut être bien exploitée par les stations qui ont un certain degré d'intelligence.

les inconvénients:

. augmentation de la complexité de l'interface composant FIP-Station,

. alourdissement des tâches confiées à la station,

. ralentissement possible de la cadence de transmission des trames; ce problème se pose lorsque le temps de réponse des équipements formant partie de la station est supérieur au délai accordé par le système pour la reconnaissance de l'adresse.

La possibilité d'intégrer l'identification de l'adresse à l'intérieur du circuit résoud les inconvénients de la méthode précédente, mais elle présente les défauts suivants:

. manque de souplesse pour programmer la (ou les) adresse(s) dans le circuit;

. augmentation de la complexité du circuit,

. diminution de la flexibilité de communication pour les applications intelligentes si la décision se fait sur l'intégration d'une seule adresse.

Pour les raisons déjà énoncées et pour favoriser les stations "non intelligentes", nous avons choisi de reconnaître l'adresse à l'intérieur du circuit.

2.5.3 Programmation de l'adresse de la station

Il reste à définir, pour compléter le mécanisme de reconnaissance d'adresses, la méthode à utiliser pour programmer l'adresse à l'intérieur du circuit.

Deux méthodes pour la programmation de cette adresse sont possibles: une méthode dynamique et une méthode statique.

Méthode dynamique

Toute station peut dynamiquement modifier l'adresse programmée dans le circuit. Cette méthode n'est pas envisageable pour des raisons de sécurité liées au fonctionnement du réseau FIP ou des réseaux de ce niveau.

Méthode statique

Par statique il faut entendre que l'on ne peut programmer l'adresse qu'à certains moments et sous un état bien déterminé du couple composant FIP-VLSI et station. Dans la méthode statique plusieurs solutions existent. On peut en citer quelques-unes:

. programmation de l'adresse dans une mémoire programmable du type EPROM, EEPROM,

. programmation de l'adresse directement sur les broches du circuit,

. programmation par codage de l'adresse à l'initialisation du circuit.

1.- La première solution peut très bien convenir pour remplir les spécifications de ce circuit communicateur. Elle peut aussi convenir tant du point de vue de l'utilisateur que du point de vue application; elle pourrait donner plus de flexibilité au moment de l'implantation du système. Mais, cette solution dépend de la technologie à utiliser dans la fabrication du circuit (en l'occurrence, en NMOS L3). Malheureusement, une telle technologie n'est pas encore disponible pour la réalisation de ce circuit.

2.- L'implantation de la seconde solution nécessite d'utiliser, uniquement pour la définition de l'adresse, 16 broches du circuit. Ceci présente plusieurs inconvénients du point de vue coût de fabrication du circuit:

. le premier est une augmentation de la surface du circuit ce qui a pour conséquence une diminution du rendement (rapport du nombre de puces bonnes au nombre total de puces),

. un second inconvénient est le coût d'encapsulation du circuit, lequel représente une part importante dans son prix de revient.

3.- La troisième solution est une méthode qui permet de réduire le nombre de broches à utiliser pour programmer l'adresse.

La méthode consiste à stocker l'adresse à travers 4 broches, en entrée du circuit, connectées par des "straps" au bus de données, sur lequel une séquence particulière sera présente à l'initialisation du circuit.

La séquence générée par le circuit est constituée par 4 combinaisons (TOP0,.....et TOP3) montrées dans le tableau 3.2.I suivant. Chaque combinaison est d'une durée d'une période d'horloge. Ces 4 combinaisons permettent de définir 65536 adresses possibles.

TOPS D'HORLOGES

BUS DE DONNEES

N° bit	TOP3	TOP2	TOP1	TOP0
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

Tableau 3.2.I. Séquence d'initialisation.

La sélection d'une adresse se fera par le choix de 4 voies parmi les 16 qui forment le bus de données, chaque voie permet de coder 4 bits de l'adresse, (voir tableau 3.2.I). Ces voies seront connectés sur les 4 broches dédiées à la programmation de l'adresse.

La figure suivante illustre la méthode utilisée pour la sélection de l'adresse.

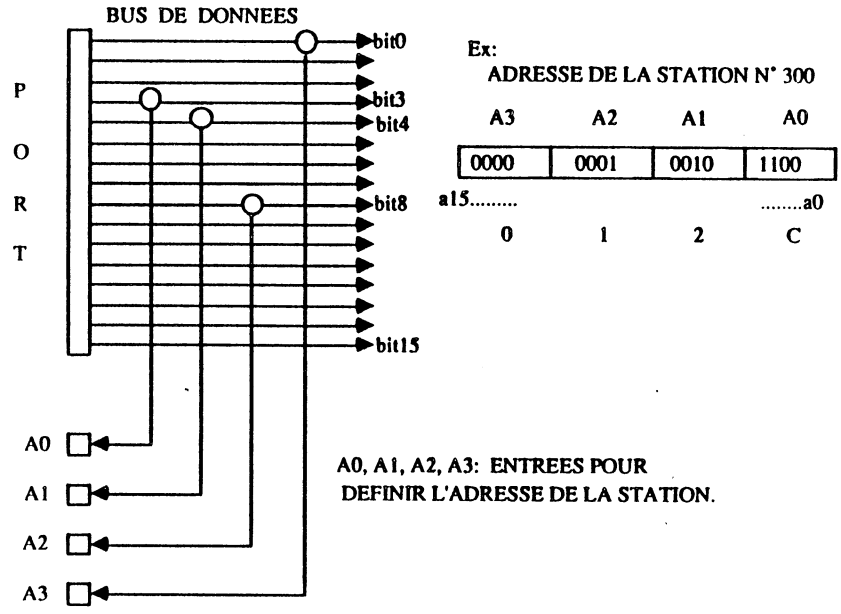


Figure 3.2.12 Méthode de sélection de l'adresse de la station

2.6 *Interface côté medium*

L'interface du circuit FIP-VLSI côté support de transmission doit encoder et décoder complètement les trames série, de telle sorte que seuls des circuits d'amplification seront nécessaires pour l'interfacer au support de communication, (figure 3.2.17).

Une caractéristique de cette interface est d'être indépendante vis-à-vis du medium de communication utilisé dans le réseau (câble coaxial, fibres optiques, paire torsadée).

2.6.1 Type de Codage.

La modulation des transmissions sera faite en bande de base et le codage des informations en code MANCHESTER biphase.

Une particularité de ce type de codage est de transmettre le signal d'horloge en même temps que les données. Ceci donne la possibilité au récepteur de récupérer l'horloge et de se resynchroniser.

Les figures 3.2.13 et 3.2.14 représentent respectivement l'encodage MANCHESTER des informations binaires de valeur "0" et "1" logique:

- le "0" logique, sera représenté par un front descendant au milieu de la période correspondant à un élément binaire,

(figure 3.2.13).

Figure 3.2.13 Encodage MANCHESTER d'un "0" logique.

- le "1" logique, sera représenté par un front montant au milieu de la période correspondant à un élément binaire.

Figure 3.2.14. Encodage MANCHESTER d'un "1" logique.

Ainsi, par exemple les chronogrammes des trames de NOM et de DONNEES, lorqu'elles transitent sur le medium de communication, sont-elles représentés par les figures 3.2.15 et 3.2.16.

Figure 3.2.15 chronogramme TRAME__de__NOM

Figure 3.2.16 chronogramme TRAME__de__DONNEES

2.6.2 Indépendance vis-à-vis du médium.

L'indépendance vis-à-vis du médium de communication sera obtenue uniquement si le circuit communicateur fournit les signaux nécessaires aux amplificateurs de ligne pour qu'ils puissent réaliser la mise en forme des trames pour leur transmission sur le support de communication. Il faut aussi prévoir une entrée pour la réception des informations.

La figure 3.2.17 montre les signaux qui vont constituer l'interface avec les outils de ligne (un transmetteur et un récepteur): trois signaux pour l'émission (Tx, Txdd et RTS) fournis par le circuit et un pour la réception (Rx).

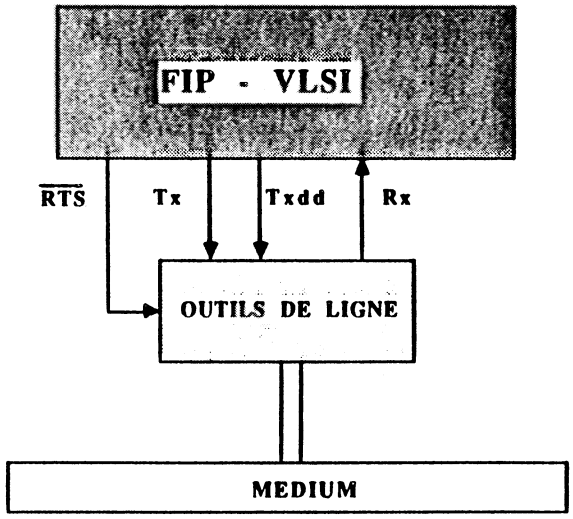


Figure 3.2.17 Interface côté support de communication.

- Tx est un signal en sortie du circuit qui fournit directement les données codées prêtes à être transmises.

- Txdd est le même signal présenté sur Tx mais retardé de 100ns. Ceci est utilisé par certains amplificateurs pour améliorer la définition des niveaux (courant continu) qui forment les impulsions [NS-80].

- RTS de l'anglais "prêt à émettre" (ready to send) est aussi un signal de sortie pour contrôler la transmission. Lorsqu'il existe une trame à émettre, il passe à l'état actif et valide le transmetteur (amplificateur d'émission) pour l'émission des signaux fournis par Tx. En repos, RTS inhibe l'amplificateur en le mettant dans l'état haute impédance pour ne pas gêner les autres transmetteurs lorsqu'ils émettent sur le réseau.

- Rx est une entrée du circuit pour la réception des informations issus du récepteur (amplificateur de réception).

2.7 *Initialisation du circuit*

Le circuit FIP-VLSI doit comporter un mécanisme qui puisse le configurer dans un état connu et stable. Ce mécanisme peut être un signal de remise à zéro (RESET) qui effectue l'initialisation du circuit.

Lorsque le signal RESET est en état actif, plusieurs fonctions réalisées par le circuit peuvent être définies. Ces fonctions sont les suivantes: le mode de fonctionnement de la station, le chargement de l'adresse physique de la station la définition du sens de transfert des fils qui forment le bus de données.

2.7.1 Mode de fonctionnement

La définition du mode de fonctionnement du circuit sera faite à travers deux fils qui permettent de coder les 3 modes de fonctionnement du FIP-VLSI: distributeur, appareil et observateur. Ce codage est résumé dans le tableau suivant:

mode	fil_1	fil_2
Distributeur	0	1
Appareil	1	1
Observateur	X	0

Tableau 3.2.II. Définition des modes de fonctionnement du circuit FIP-VSLI.

2.7.2 Chargement de l'adresse physique de la station

Un choix important dans les spécifications du circuit est la définition de l'adresse physique de la station. La décision a été prise dans le paragraphe 2.4.3 ainsi que la présentation de la méthode à utiliser pour la programmer dans le circuit.

2.7.3 Définition du sens de transfert du bus de données

La programmation des deux groupes de lignes, qui peuvent constituer le bus de données, est effectuée par deux lignes de contrôle d'entrée selon la technique suivante:

La première ligne sera connectée sur un des 16 bits du bus de données et utilisée comme curseur pour définir le nombre de lignes en entrée et celui de lignes en sortie.

On va considérer que le bus de données est organisé du bit de poids le plus faible (bit 0) au bit de poids le plus fort (bit 15); lorsque la première ligne est connectée à n'importe quel bit des données, tous les bits de poids plus faibles seront considérés comme des bits en sortie et le restant comme des bits en entrée. Par exemple, lorsque le curseur se situe sur le bit 8 du bus de données, il définit 8 bits en entrée et 8 en sortie.

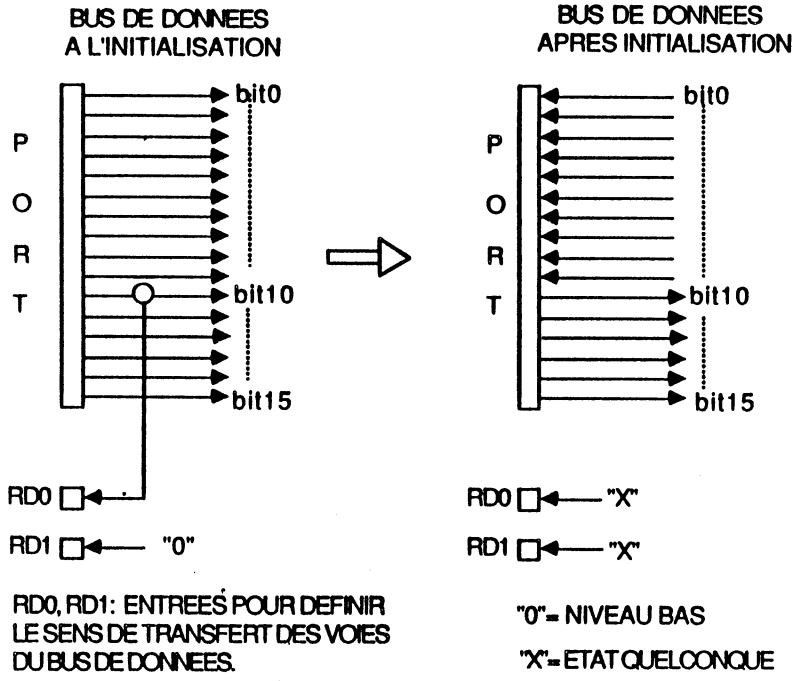


Figure 3.2.18 Exemple de définition du sens de transfert des voies

L'expression 0 =< n,p =< 16 indique la fourchette de valeurs que les deux groupes de voies n et p peuvent en prendre, le cardinal de ces valeurs est égal à 17. Puisque, la première ligne de contrôle permet seulement de définir 16 valeurs, la deuxième ligne mentionnée sera utilisée pour définir la 17ième combinaison. Celle-ci permet de définir les 16 voies en entrée ou en sortie (selon la convention choisie), lorsque la ligne est validée.

2.8 *Définition de la base de temps du circuit*

Une caractéristique souhaitée du circuit FIP-VLSI est de pouvoir fournir un débit de 2.5 Mbits/seconde.

Pour des raisons de récupération des erreurs dûes à l'introduction des impulsions parasites dans la transmission, il est conseillé d'échantillonner le signal reçu à une fréquence au moins 8 fois supérieure au débit de transmission et ainsi pouvoir reconstituer les informations en ayant éliminé les impulsions parasites supérieures à 50 ns. Si la fréquence d'échantillonnage est plus élevée la largeur de l'impulsion parasite détectée sera plus faible.

Un quartz connecté à l'extérieur du circuit, d'une fréquence 8 fois supérieur au débit de transmission, devra être employé pour fixer la base de temps du circuit. Le circuit doit aussi fournir un signal d'horloge pour usage externe de 4 fois le débit.

2.9 *Définition de l'organisation externe du circuit*

Pour terminer les spécifications du circuit communicateur FIP-VLSI, il reste à donner le brochage du circuit .

La figure 3.2.19 montre l'organisation externe du circuit, laquelle est formée de 39 broches classifiées en plusieurs groupes. Le circuit pourra ainsi être intégré dans un boîtier de 40 broches.

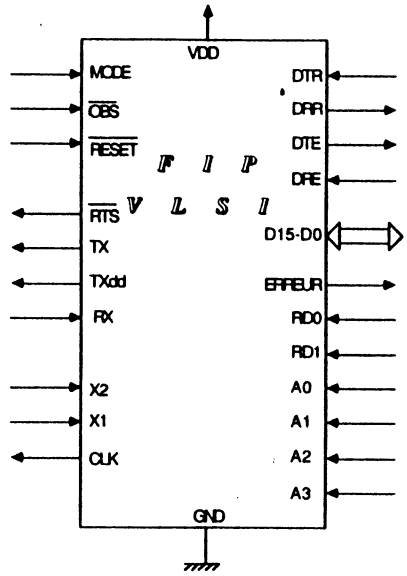


Figure 3.2.19 Organisation externe du circuit communicateur **FIP-VLSI**

Les broches qui constituent le circuit sont décrites dans l'annexe F.

spécifications du communicateur

Remarque.- Tous les signaux d'entrée ou de sortie sont compatibles TTL et seuls les signaux d'entrées/ sorties qui forment le bus de données sont en logique trois états.

3. CONCEPTION DU CIRCUIT COMMUNICATEUR

FIP – VLSI

3 CONCEPTION DU CIRCUIT COMMUNICATEUR FIP-VLSI

3.1 *Présentation de la méthodologie de conception adoptée*

La réalisation d'un circuit intégré comporte deux processus indépendants : la conception et la fabrication.

La conception est la traduction des spécifications fonctionnelles du circuit dans un jeu de masques. Ces masques sont utilisés dans le processus de fabrication du circuit.

Le coût de revient d'un circuit intégré à la demande prend donc en compte le coût de conception, le coût des masques, le coût de fabrication et le coût de test.

Si l'on considère uniquement les facteurs qui interviennent dans le coût de réalisation dûs à la conception, on s'aperçoit qu'ils dépendent directement de l'expérience et donc de la productivité moyenne du (ou des) concepteur(s) et des moyens de CAO donnés.

" La productivité de conception dépend essentiellement de la méthode de conception utilisée, la répétitivité du circuit, et de l'expérience du ou de l'équipe de concepteurs sur la conception du même type de circuit et même technologie" [BOU-85].

C'est précisement une méthodologie, dans le sens de l'augmentation de la productivité de la conception, que nous allons essayer de valider dans la conception du circuit communicateur FIP-VLSI.

Cette méthodologie consiste à adopter la double démarche suivante:

. une analyse descendante des spécifications du circuit à concevoir; cette analyse permet de définir les modules fonctionnels puis les opérateurs qui les réalisent;

. la construction du circuit à partir de la bibliothèque d'opérateurs flexibles, supposée existante, qui correspond aux opérateurs définis ci-dessus.

La figure 3.3.1 illustre la méthodologie proposée. Les blocs hachurées sont considérées comme des parties déjà existantes.

Comme la bibliothèque d'opérateurs flexibles ne peut être que dédiée à un type de circuits en particulier, pour des raisons d'optimisation et de complétude, nous rappelons que notre méthodologie s'adresse en l'occurrence aux circuits de communication.

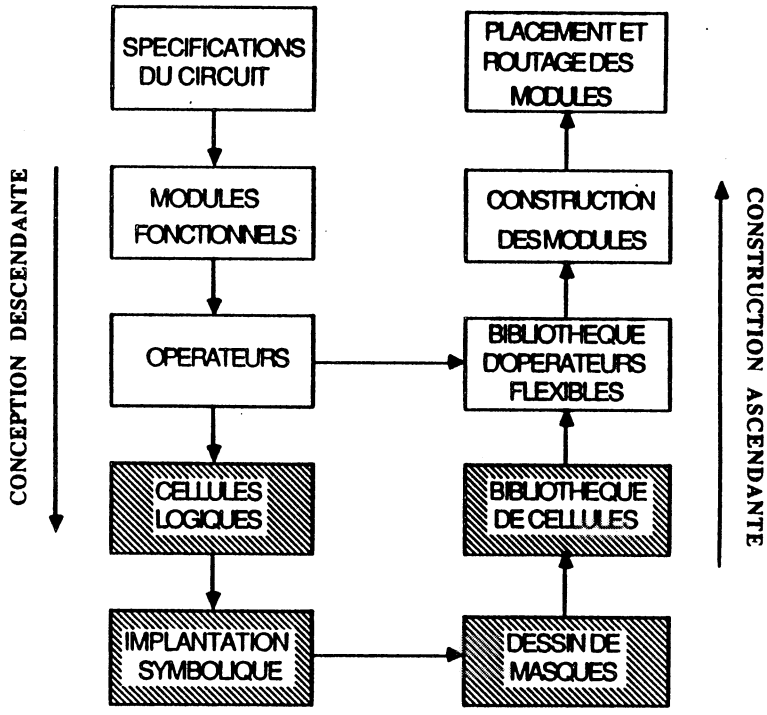


Figure 3.3.1

3.2 *Analyse descendante dans le cas du circuit FIP-VLSI*

La mise en ouvre de la méthodologie proposée, selon l'approche descendante, consiste à réaliser:

. l'analyse fonctionnelle des différents modules constituant le circuit,

. puis l'analyse des différents types d'opérateurs qui forment chaque module, pour arriver à la description logique de chaque opérateur.

3.2.1 Analyse Fonctionnelle

Le circuit FIP-VLSI, indépendamment des problèmes de gestion d'erreurs et d'initialisation, réalise les fonctions classiques d'un circuit contrôleur de communication. Il intègre un ensemble de fonctions des niveaux liaison (MAC) et physique (PS) des couches 2 et 1 respectivement du modèle IEEE 802, (la figure 3.3.2 illustreces fonctions).

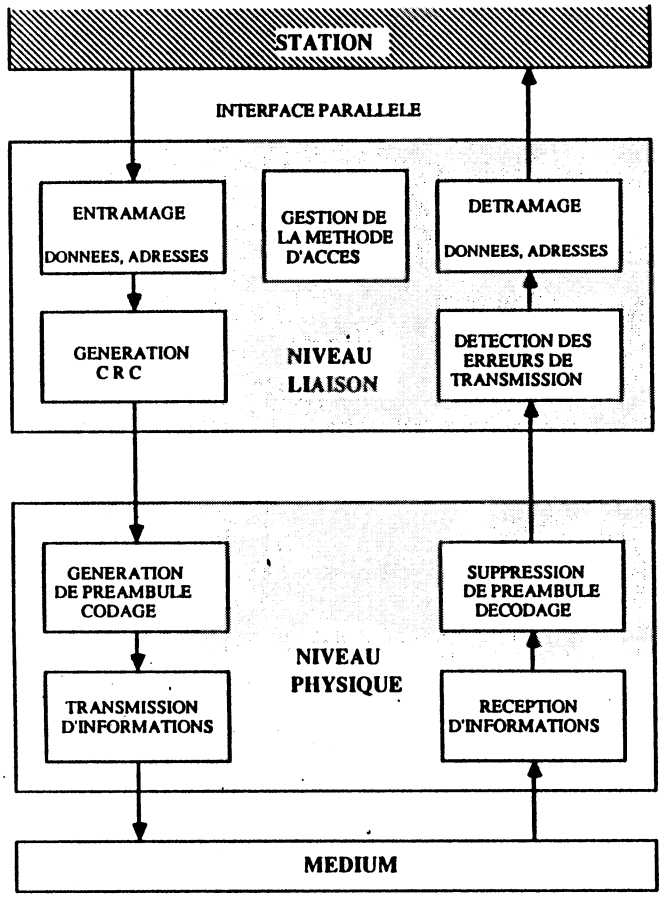


Figure 3.3.2

Fonctions réalisées au niveau MAC

1.- En émission:

. réception des données proposées par la station et construction de la trame,

. présentation d'une séquence de bits série au niveau physique pour sa transmission sur le medium,

. calcul et ajout à la trame de la valeur de CRC correspondante.

2.- En réception:

. réception de la séquence de bits série en provenance du niveau physique,

. passage à la station des trames qui lui ont été adressées,

. élimination des trames qui ne concernent pas la station,

. vérification du champ FCS de la trame pour détecter de possibles erreurs dûes à la transmission.

3.- gestion de la méthode d'accès:

. envoi et reconnaissance de la trame de nom.

Fonctions réalisées au niveau PS

1.- En émission:

. génération du préambule et du motif de synchronisation,

. codage de la trame,

. ajout de la séquence de fin_trame.

2.- En réception:

. détection de la trame et mise en forme des données qui la constituent,

. détection du préambule, motifs de synchronisation et fin_trame,

. validation de la réception de la trame,

. décodage et transmission des données vers le niveau MAC.

3.2.2 Description du fonctionnement du FIP-VLSI

La spécification des fonctions que le circuit FIP-VLSI doit réaliser peut prendre une forme plus claire à travers un modèle procédural.

Ce modèle donne la possibilité de regrouper plusieurs fonctions dans une procédure ou macrofonction. Cette tendance permet donc une spécification compréhensible du circuit et quelle que soit sa complexité, le nombre d'objets manipulés dans la description tend à rester constant.

Le modèle ne peut pas être considéré comme une formalisation pour l'implantation matérielle du circuit car celle-ci dépend des contraintes d'efficacité et disponibilité dûes à une technologie particulière.

Le modèle sera exprimé comme un programme informatique écrit en pseudo-PASCAL.

Dans la description algorithmique du circuit, donnée dans l'annexe G, les considérations suivantes ont été prises en compte concernant le formalisme utilisé:

. la description est faite suivant un ordre hiérarchique descendant,

. les variables globales permettent d'indiquer qu'une même variable peut être employée par deux modules ou plus,

. l'utilisation des variables structurées (struct) permet de représenter une même variable de plusieurs manières,

. plusieurs fonctions utilisées ne sont pas détaillées dans la description à cause de leur simplicité,

. le parallélisme n'est pas exprimé explicitement dans la description.

3.2.3 Architecture interne du circuit FIP-VLSI

Bien que le choix d'une telle architecture ne soit pas encore une démarche courante, celle du FIP-VLSI est constituée d'une partie opérative et d'une partie contrôle.

En plus de ces deux parties, la figure 3.3.3 fait apparaître les bus internes d'interconnexion et les signaux d'entrées/sortie du circuit.

3.2.3.1 Partie Opérative

On peut considérer que la partie opérative du circuit est principalement constituée de deux parties indépendantes: une partie émission et une partie réception.

*Partie Emission, elle est constituée des modules suivants:

. registre d'émission: il reçoit les données en parallèle en provenance de la station et les transforme en une séquence de bits,
. entramage: il réalise le multiplexage entre les différents champs qui composent la trame,

. générateur de CRC: effectue le calcul du code de redondance cyclique sur les données à transmettre,
. générateur de préambule: produit une séquence de 6 bits à "1",
. encodeur biphase: réalise le codage MANCHESTER des données,
. violation de code: génère le motif de synchronisation. Il effectue une violation de code sur la trame après la transmission du préambule.

*Partie Réception:

. remise en forme: reçoit et remet en forme les données,
. décodeur MANCHESTER: transforme les données reçues sur la ligne série en informations binaires. En même temps, il permet de récupérer l'horloge transmis avec les données pour resynchroniser le circuit,
. détection trame: permet la reconnaissance du type de trame et valide l'arrivée des informations,
. registre de réception (détramage): transforme la séquence de données série dans un mot de 16 bits pour être envoyé vers la station,
. reconnaissance d'adresse: effectue la comparaison entre l'adresse de la station et les adresses contenues dans la trame de nom,
. vérification du CRC: détecte les erreurs possibles dûes à la transmission.

En plus des parties émission et réception, le circuit comporte aussi l'interface avec la station. Elle est constituée des modules suivants:

. port parallèle: assure l'interface physique entre la station et les parties émission et réception,
. système d'initialisation: génère la séquence d'initialisation utilisée pour définir l'adresse du circuit,
. direction données: permet de définir le sens de transfert des voies qui forment le bus de données,
.registre d'état: permet d'indiquer l'état de réception des trames.

Enfin pour terminer la description de la partie opérative du circuit, il reste à mentionner le générateur d'horloge qui génère la base de temps du circuit.

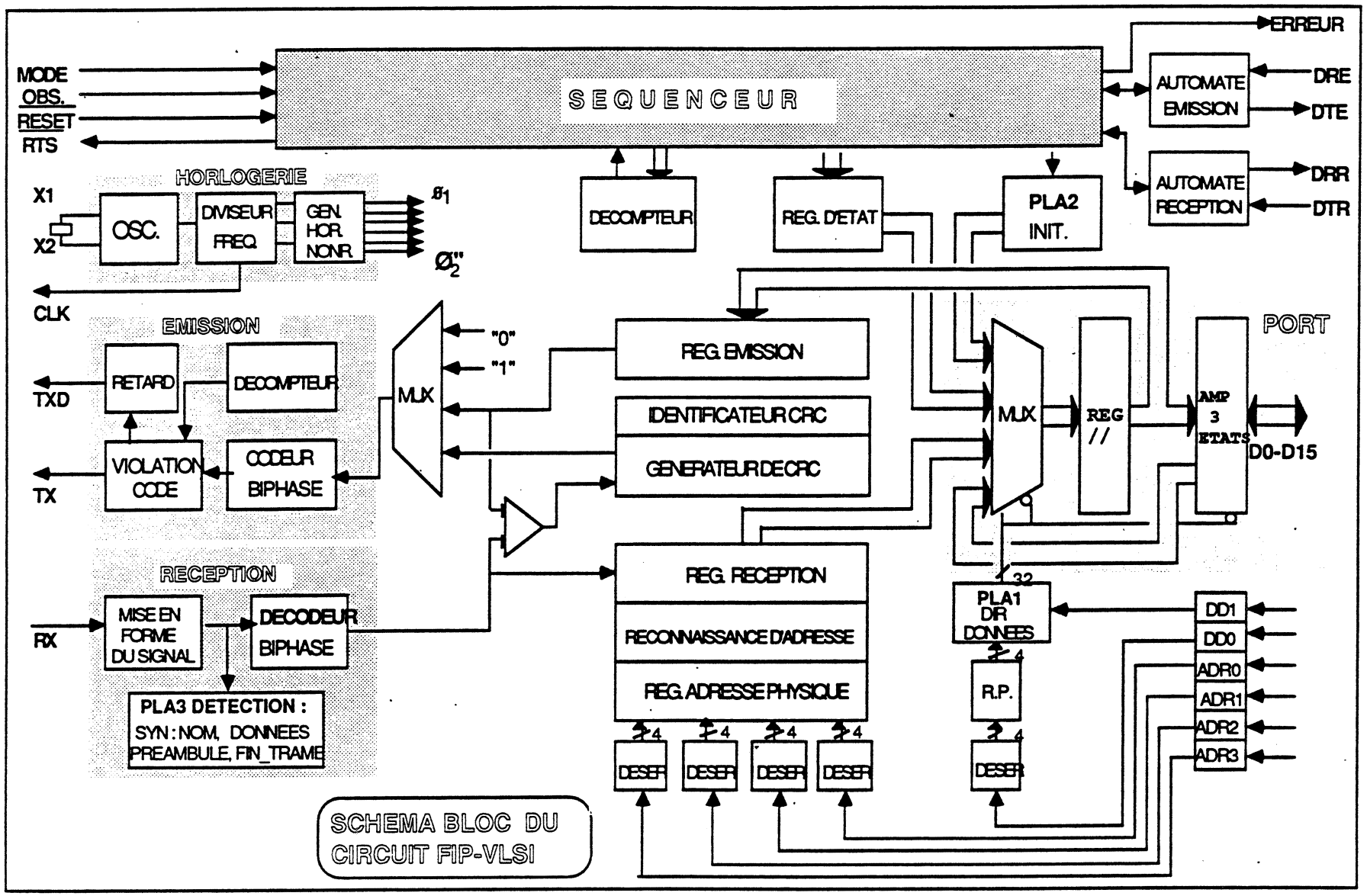

MODE
OBS.
RESET
RTS
SEQUENCEUR
ERREUR
AUTOMATE EMISSION
DRE
DTE
AUTOMATE RECEPTION
DRR
DTR
HORLOGERIE
X1
X2
OSC.
DIVISEUR FREQ.
GEN. HOR. NONR.
Ø1
Ø2"
CLK
DECOMPTEUR
REG. D'ETAT
PLA2 INIT.
EMISSION
RETARD
DECOMPTEUR
TXD
VIOLATION CODE
CODEUR BIPHASE
TX
MUX
"0"
"1"
REG. EMISSION
IDENTIFICATEUR CRC
GENERATEUR DE CRC
MUX
REG //
AMP 3 ETATS
PORT
D0-D15
32
RECEPTION
RX
MISE EN FORME DU SIGNAL
DECODEUR BIPHASE
PLA3 DETECTION :
SYN : NOM, DONNEES
PREAMBULE, FIN_TRAME
REG. RECEPTION
RECONNAISSANCE D'ADRESSE
REG. ADRESSE PHYSIQUE
PLA1 DIR DONNEES
R.P.
DD1
DD0
ADR0
ADR1
ADR2
ADR3
4
DESER
SCHEMA BLOC DU CIRCUIT FIP-VLSI

Figure 3.3.3

3.2.3.2 Partie Contrôle

La partie contrôle génère, à partir du compte rendu de la partie opérative et de la décision des actions à entreprendre en fonction de l'état des entrées, les signaux nécessaires pour commander chacun des modules constituant la partie opérative. Elle doit s'occuper aussi de gérer la méthode d'accès sur le réseau et le contrôle de la communication avec la station.

3.2.3.3 Quelques remarques sur l'architecture du circuit

Définition des bus internes

Les interconnexions, pour acheminer les données et les commandes entre les différents modules du circuit, sont réalisées par des bus statiques. Ce choix a été fait en prenant en compte les considérations suivantes:

. aucune considération sur le temps ne doit être faite sur les signaux transmis sur le bus; ceci simplifie l'analyse d'implantation,

. les problèmes électriques que l'on peut rencontrer par l'adoption de cette solution sont:

. l'augmentation de la capacité du bus, dûe aux capacités introduites par tous les éléments connectés sur le même fil,

. l'augmentation de la résistivité, dûe à la longueur des fils,

ces inconvénients sont résolus par l'utilisation systématique d'amplificateurs internes pour chaque élément connecté à un bus (de données ou de contrôle). Ceci est réalisé au niveau des opérateurs utilisés.

3.2.4 Synthèse des modules

L'étape suivante de la méthode descendante proposée, consiste à décrire les modules fonctionnels en termes d'opérateurs. Pour ceci, on est amené à définir le schéma logique de chacun des modules.

3.2.4.1 Exemple de synthèse d'un module

L'interface de données entre FIP-VLSI et la station est réalisée à travers le module **port parallèle.**

La figure 3.3.4 illustre le schéma logique du module port. Il est défini par un registre parallèle 16 bits. A l'entrée de chaque bit du registre, un multiplexeur permet de sélectionner des informations différentes (séquence d'initialisation, les valeurs du registre d'état, les données en émission et/ou en réception).

Le registre doit aussi avoir un amplificateur trois états à la sortie de chacun des bits qui le constitue pour contrôler le transfert bidirectionnel des informations avec la station.

Cet exemple permet d'identifier deux types d'éléments:

. un opérateur: un registre parallèle 16 bits,
. deux cellules: une porte trois états et un multiplexeur.

Il est évident que l'opérateur registre parallèle, lui aussi, est formé par des cellules. Il peut être construit à partir des deux cellules de base reliées entre elles: une bascule (maître-esclave, D,...) et une cellule sélecteur 2 vers 1, et répétées 16 fois. Une cellule d'un amplificateur inverseur sera nécessaire pour commander les lignes de contrôle du registre. La figure 3.3.4(b) illustre cet exemple.

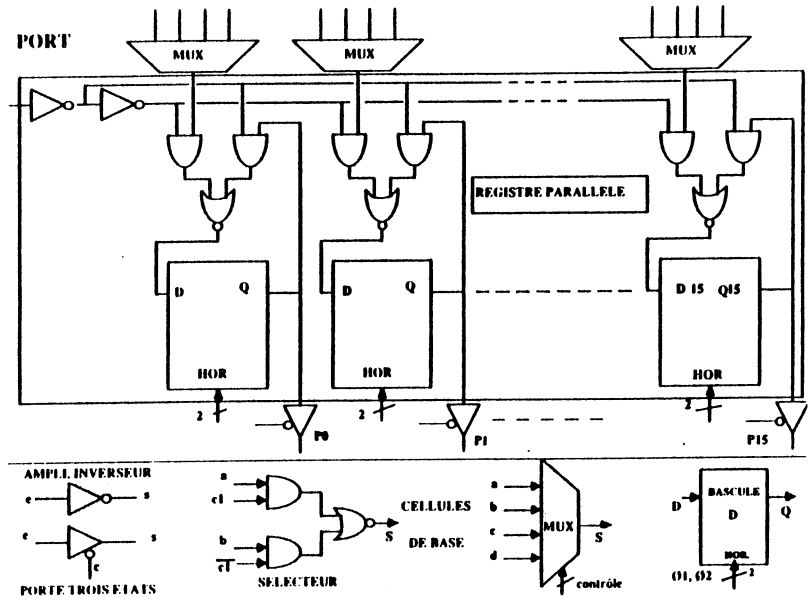


Figure 3.3.4 (a) Module port parallèle 16 bits,
(b) cellules qui constituent l'opérateur.

Cette même approche a été suivie pour la synthèse des modules restants pour pouvoir déterminer les opérateurs et cellules nécessaires pour leur implantation.

3.2.4.2 Résultats de la synthèse des modules

La synthèse de chacun des modules est résumée dans le tableau 3.3.I suivant. Ce tableau illustre aussi les résultats obtenues par l'analyse descendante appliquées au circuit FIP-VLSI.

CONCEPTION DESCENDANTE →

		FONCTION	MODULE	OPERATEUR	CELLULES
NIVEAU LIAISON	INTERFACE STATION	* INTERFACE PARALLELE	PORT	REG. PARALLELE	1.- BASCULE M-S (A) - SELECTEUR (B) - AMP. INV. MULTIPLEXEUR AMP. 3 ETATS
			S. INITIALISATION	ENCODEUR	PLA
			DIRECTION DONNEES	REG. PARALLELE REG A DECALAGE DECODEUR	IDEM 1 (A) PLA
			REGISTRE D'ETAT	REG. MEMORISATION	BISTABLE
			CONTROLE DE LA COMMUNICATION AVEC LA STATION	AUTOMATES D'ETATS FINIS	(A) PLA
	EMISSION	*ENTRAMAGE	ENTRAMAGE	SERIALISATEUR	IDEM 1 MULTIPLEXEUR
		*GENERATION CRC	GENERATEUR CRC	GENERATEUR CRC	2.- BASCULE M-S avec R.A.1 - Portes:NON-ET (C),OU-EXC (D).
	RECEPTION	*DETECTION DES ERREURS DE TX.	DETECTION CRC	GENERATEUR CRC COMPARATEUR	IDEM 2 3.- Portes: OU (E), EGALITE (F).
		*DETRAMAGE	DETRAMAGE	DESERIALISATEUR	(A)
		*GESTION DE L'ADRESSE	RECONNAISSANCE DE L'ADRESSE	REG. PARALLELE REG. A DECALAGE COMPARATEUR	IDEM 1 (A) IDEM 3
		*METHODE D'ACCES	METHODE D'ACCES	SEQUENCEUR	PLA (A)
NIVEAU PHYSIQUE	EMISSION	*GENERATION DE PREAMBULE	GENERATION DE PREAMBULE	COMPTEUR	4.- (A), (B) - Portes: (F) NON-OU (2,3,4 entrees) (G), INVERSEUR (H).
		*CODAGE	CODEUR BIPHASE	"LOGIQUE ALEATOIRE"	(A), (D)
			VIOLATION CODE	DECOMPTEUR "LOGIQUE ALEATOIRE"	5.- (A), (B), (D), (G), (H). Portes: ET (I), (E), (H).
	RECEPTION	*RECEPTION D'INFORMATIONS	REMISE EN FORME	"LOGIQUE ALEATOIRE"	(A), (C), (E), (H).
		*DECODAGE	DECODEUR	REG. A DECALAGE "LOGIQUE ALEATOIRE"	(A) (A) ,(D)
		*SUPPRESSION DE PREAMBULE	DETECTION TRAME	DESERIALISATEUR DECODEUR	(A) PLA
		** BASE DE TEMPS	GENERATEUR D'HORLOGES	OSCILLATEUR DIV. FREQUENCE	OSCILLATEUR IDEM 4 GEN. HORLOGES NON RECOUVRANTES
		** PARTIE CONTROLE	CONTROLE	SEQUENCEUR	(A) PLA

Tableau 3.3.I

Ce tableau montre, en plus des fonctions et des modules, les opérateurs constituant chaque module et les cellules associées aux opérateurs. A partir de ce tableau, la partie opérative et la partie contrôle du FIP-VLSI sont synthétisées de la manière suivante:

3.2.4.3 Synthèse de la partie opérative

Elle peut être synthétisée par trois types d'opérateurs:

1.- La synthèse en forme d'opérateurs en tranches pour les opérateurs suivants:

a) registres à décalage, registres parallèles, sérialisateur, désérialisateur, registre de mémorisation, décompteurs, et comparateurs. Le nombre de bits de chaque opérateur et le degré de transparence des opérateurs seront utilisés pour sa génération automatique.

b) dans le cas du générateur et identificateur de CRC, ils seront définis par le polynôme générateur et le reste unique (ru) à identifier.

Pour le circuit FIP-VLSI, ces paramètres sont:

$$G(X) = X^{16} + X^{12} + X^{5} + 1, \quad \text{et} \quad ru = 0001110100001111 \qquad (15 \ldots\ldots 0)$$

Il faut signaler qu'un seul générateur de CRC sera utilisé dans le circuit. Son fonctionnement sera multiplexé et il travaillera en émission ou en réception. Ceci interdit la possibilité de réaliser l'auto-test du circuit pour vérifier le bon fonctionnement des circuits d'émission et de réception. Dans le cas contraire il faut deux générateurs de CRC indépendants.

2.- La synthèse automatique à partir de leurs équations logiques des opérateurs combinatoires suivants:

. l'encodeur (PLA2), employé pour la génération de la séquence d'initialisation,

. deux décodeurs: un décodeur (PLA1) employé pour définir le sens de transfert des voies qui forment le bus de données et l'autre (PLA3) pour détecter le préambule, la synchronisation et la fin de la trame.

3.- Il existe quelques modules réalisant une fonction particulière; par exemple le générateur d'horloge, le module de violation de code, le codeur et le décodeur biphase, qui présentent une structure irrégulière. Cette particularité va entraîner un traitement spécial pendant leur implantation.

3.2.4.4 Synthèse de la partie contrôle

Quant à la partie contrôle, un choix a été fait dans le sens de centraliser la génération de toutes les commandes de contrôle du circuit et la méthode d'accès au réseau. Dans un opérateur de contrôle, ce choix est traduit par la définition d'un séquenceur mono-PLA.

Les commandes nécessaires pour contrôler le transfert de données entre la station et le circuit seront engendrées par deux automates: un automate pour la réception de données (dialogue circuit -> station) et l'autre pour l'émission (dialogue station -> circuit).

3.2.5 Plan de masse

Un plan de masse correspond à la définition topologique d'un circuit en fonction de chacun des modules implantés qui le constituent.

Dans notre cas, le plan de masse du circuit FIP-VLSI sera principalement déterminé par:

. la topologie des modules à implanter,

. la simplification des interconnexions entre modules, et

. la facilité de connexion avec les plots du circuit.

Ces considérations ont été prises, d'une part pour gagner du temps dans la construction du circuit, ainsi comme pour mettre en relief la construction ascendante proposée dans la méthodologie (par l'assemblage des modules).

3.2.6 Simulation logico-fonctionnelle

Il faut signaler que chacun des modules a été simulé à l'aide du simulateur logico-fonctionnel FIDEL [THA-84], ainsi que l'ensemble du circuit FIP-VLSI.

3.3 *Mise en oeuvre de la construction ascendante du circuit*

La deuxième partie de la méthodologie proposée est la construction ascendante du circuit. Cette partie de la méthodologie, illustrée par les parties non hachurées de la figure 3.3.1, consiste à effectuer :

. l'implantation des différents modules à partir de la bibliothèque d'opérateurs flexibles et la bibliothèque de cellules,

. le placement et routage de tous les modules pour former le circuit.

3.3.1 Implantation des opérateurs

Ainsi, à partir des bibliothèques de cellules et d'opérateurs flexibles, l'implantation des différents opérateurs a-t-elle été faite de deux manières selon leurs caractéristiques: automatiquement et manuellement.

Implantation automatique

Un des buts de la réalisation du circuit FIP-VLSI a été justement la mise en oeuvre des bibliothèques de cellules et d'opérateurs flexibles définies au § 2.3.

Ainsi, à partir des opérateurs en tranches, des opérateurs combinatoires et des opérateurs de contrôle, les opérateurs, définis dans le tableau 3.3.I seront-ils implantés automatiquement selon les paramètres correspondants.

Implantation manuelle

Il est certain que tous les opérateurs ne sont pas générés automatiquement, d'une part parce que plusieurs d'entre eux présentent une structure irrégulière propre aux caractéristiques du circuit et d'autre part parce que quelques-uns ont besoin d'un traitement particulier.

Dans le cas du circuit FIP-VLSI, et plus particulièrement pour l'implantation des modules suivants: violation de code, codeur et décodeur biphase, on a été obligé de dessiner plusieurs cellules supplémentaires.

Ces nouvelles cellules ont été dessinées en respectant les mêmes règles définies lors de l'implantation de la bibliothèque de cellules. Ceci a l'avantage d'enrichir la bibliothèque.

Le tableau 3.3.II présente en résumé la manière selon laquelle les opérateurs qui constituent les modules du circuit ont été implantés. S'ils ont été implantés automatiquement les paramètres correspondants sont indiqués. Autrement, il s'agit d'une implantation manuelle ("structure irrégulière").

Le tableau montre aussi la manière selon laquelle l'approche de construction ascendante est réalisée, ceci est vu en parcourant le tableau de droite à gauche, sa mise en oeuvre sera présentée dans les paragraphes suivants.

CONSTRUCTION ASCENDANTE ←

		FONCTION	MODULE	OPERATEUR	CELLULES	PARAMETRES
NIVEAU LIAISON	INTERFACE STATION	* INTERFACE PARALLELE	PORT	REG. PARALLELE	1.- BASCULE M-S (A) - SELECTEUR (B) - AMP. INV. MULTIPLEXEUR AMP. 3 ETATS	16 bits, 16 (4 vers 1) 16 (plot 3 etats)
			S. INITIALISATION	ENCODEUR	PLA	Equation logique (2)
			DIRECTION DONNEES	REG. PARALLELE REG A DECALAGE DECODEUR	IDEM 1 (A) PLA	4 bits 4 bits Equation logique (1)
			REGISTRE D'ETAT	REG. MEMORISATION	BISTABLE	11 bits
			CONTROLE DE LA COMMUNICATION AVEC LA STATION	AUTOMATES D'ETATS FINIS	(A) PLA	2 et 3 bits Graphe reception (1) Graphe emission (2)
	EMISSION	*ENTRAMAGE	ENTRAMAGE	SERIALISATEUR	IDEM 1 MULTIPLEXEUR	16 bits 1 (4 vers 1)
		*GENERATION CRC	GENERATEUR CRC	GENERATEUR CRC	2.- BASCULE M-S avec R.A.1 - Portes:NON-ET (C),OU-EXC (D).	Polynome (1)
	RECEPTION	*DETECTION DES ERREURS DE TX.	DETECTION CRC	GENERATEUR CRC COMPARATEUR	IDEM 2 3.- Portes: OU (E), EGALITE (F).	Polynome (1) reste unique
		*DETRAMAGE	DETRAMAGE	DESERIALISATEUR	(A)	17 bits
		*GESTION DE L'ADRESSE	RECONNAISSANCE DE L'ADRESSE	REG. PARALLELE REG. A DECALAGE COMPARATEUR	IDEM 1 (A) IDEM 3	16 bits 4 bits (4 fois) 16 bits
		*METHODE D'ACCES	METHODE D'ACCES	SEQUENCEUR	PLA (A)	Graphe (3)
NIVEAU PHYSIQUE	EMISSION	*GENERATION DE PREAMBULE	GENERATION DE PREAMBULE	COMPTEUR	4.- (A), (B) - Portes: (F) NON-OU (2,3,4 entrees) (G), INVERSEUR (H).	5 bits (*)
		*CODAGE	CODEUR BIPHASE	"LOGIQUE ALEATOIRE"	(A), (D)	"Structure irregulière"
			VIOLATION CODE	DECOMPTEUR "LOGIQUE ALEATOIRE"	5.- (A), (B), (D), (G), (H). Portes: ET (I), (E), (H).	3 bits "Structure irregulière"
	RECEPTION	*RECEPTION D'INFORMATIONS	REMISE EN FORME	"LOGIQUE ALEATOIRE"	(A), (C), (E), (H).	"Structure irregulière"
		*DECODAGE	DECODEUR	REG. A DECALAGE "LOGIQUE ALEATOIRE"	(A) (A) ,(D)	5 bits "Structure irregulière"
		*SUPPRESSION DE PREAMBULE	DETECTION TRAME	DESERIALISATEUR DECODEUR	(A) PLA	10 bits Equation logique (3)
		** BASE DE TEMPS	GENERATEUR D'HORLOGES	OSCILLATEUR DIV. FREQUENCE	OSCILLATEUR IDEM 4 GEN. HORLOGES NON RECOUVRANTES	OSCILLATEUR 3 bits 5, 10, et 20MHz 40, 6, et 2 pf.
		** PARTIE CONTROLE	CONTROLE	SEQUENCEUR	(A) PLA	7 bits Graphe (3)

Tableau 3.3.II

3.3.2 Implantation des modules

Un module peut être défini par un ou plusieurs opérateurs. L'implantation des modules peut donc être faite soit directement par la génération automatique de l'opérateur, soit par la génération indépendante de chaque opérateur constituant le module, suivie de leur assemblage.

Dans le tableau 3.3.II, les modules correspondants au niveau physique du modèle OSI ont été pratiquement tous assemblés manuellement.

3.3.3 La couronne de plots

Les plots d'entrée et de sortie, ont été générés automatiquement à partir de la bibliothèque d'amplificateurs compilables définie dans le système CALMA.

Les plots d'entrée peuvent être définis par le plot lui-même, une protection et un amplificateur. Dans le cas du FIP-VLSI, un plot de ce type a été utilisé.

Les plots utilisés pour l'alimentation sont aussi définis dans CALMA, ils sont constitués d'un simple pavé qui permet l'interface entre le circuit et le monde extérieur.

Les plots de sortie sont constitués par le plot et un amplificateur de sortie. Ces plots, selon leurs caractéristiques, ont été divisés en 3 groupes: un groupe formé de 5 plots, un autre de deux et le dernier d'un seul plot. Leur génération automatique a eu besoin de la définition des caractéristiques suivantes:

. la capacité de charge prévue pour chaque élément,

. la fréquence de fonctionnement.

Le plot trois états, utilisé pour l'interface des données avec la station, est un plot déjà caractérisé. Il a été récupéré d'un autre circuit réalisé dans la même technologie et avec des caractéristiques similaires à celles définies pour le circuit FIP-VLSI.

Enfin, la couronne de plots est constitué par:

. 12 plots d'entrée,
. 8 plots de sortie.

. 16 plots 3 états,
. 2 plots pour le quartz ou l'oscillateur (similaires aux plots d'alimentation),
. 2 plots d'alimentation
. 1 plot pour la polarité substrat.

Il faut signaler que le circuit FIP-VLSI prévoit 39 plots (40 avec la polarité substrat qui n'est pas considérée dans l'encapsulation). Donc, pour profiter des 40 plots à utiliser lors de l'encapsulation, un plot de sortie a été ajouté pour exploiter les données décodées de la trame reçue.

La figure 3.3.5 montre la couronne de plots et le placement définitif de chacun d'eux.

3.3.4 Placement et routage des modules du FIP-VLSI

La dernière étape dans le dessin des masques du composant FIP-VLSI a été le placement de tous les modules et la réalisation du routage nécessaire pour leur interconnexion.

La construction de la couronne de plots a été réalisée en fonction du placement des modules pour faciliter les interconnexions.

* Placement des modules

En fonction des considérations faites pour le plan de masse du circuit, le placement des modules a été réalisé pour former 4 blocs topologiques indépendants, (figure 3.3.5). Ces 4 blocs réalisent chacun une fonction correspondant au modèle OSI.

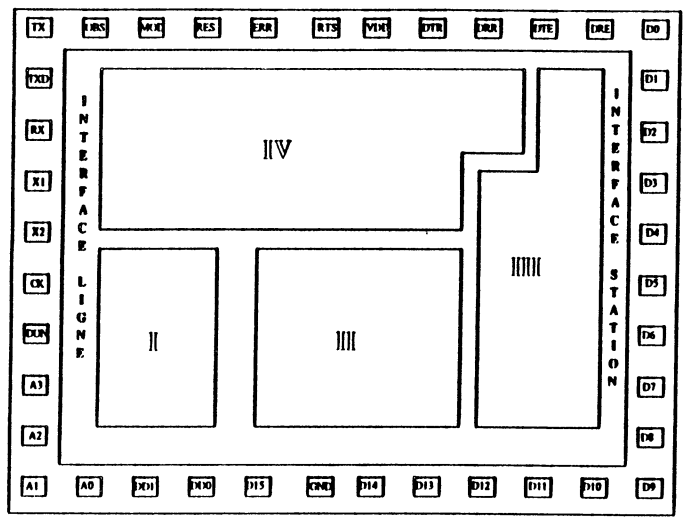


Figure 3.3.5

Le bloc I regroupe les modules qui réalisent les fonctions correspondant à l'émission et à la réception du niveau physique tels qu'ils sont définis dans les tableaux 3.3.I et 3.3.II. En plus, ce bloc contient le module qui génère la base de temps du circuit. Le placement de ces modules est illustré par la figure 3.3.6.

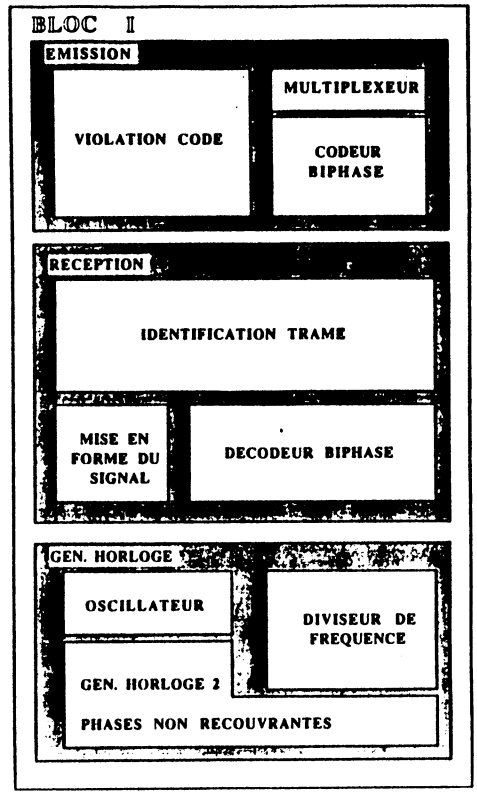


Figure 3.3.6

Le bloc II réalise les fonctions d'émission et de réception correspondant au niveau liaison. Ce bloc est formé de plusieurs modules générés automatiquement et leur régularité a permis de les assembler automatiquement par simple empilement. La figure 3.3.7 illustre les modules concernés.

Le bloc III regroupe les modules qui réalisent les fonctions propres à l'interface côté station. Les modules qui forment ce bloc sont illustrés par la figure 3.3.8. Les bus faisant partie de ce bloc ont été générés par programme et réliés au port.

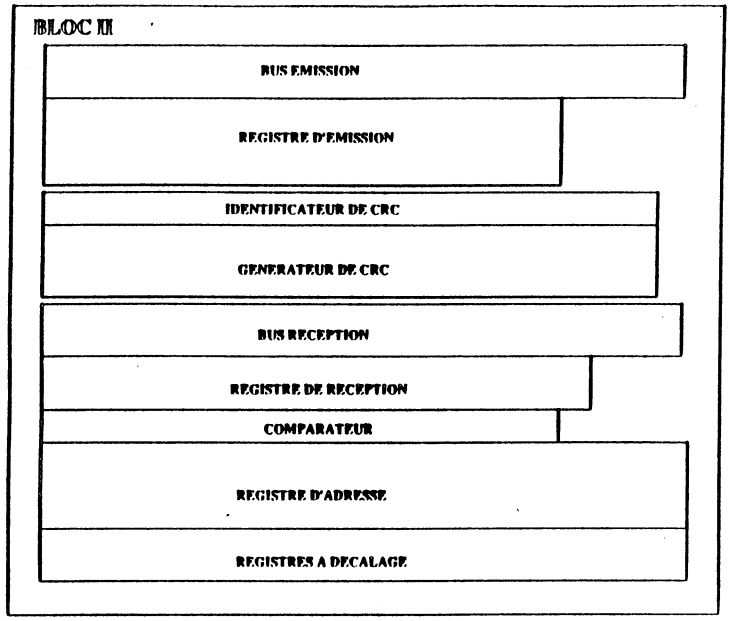


Figure 3.3.7

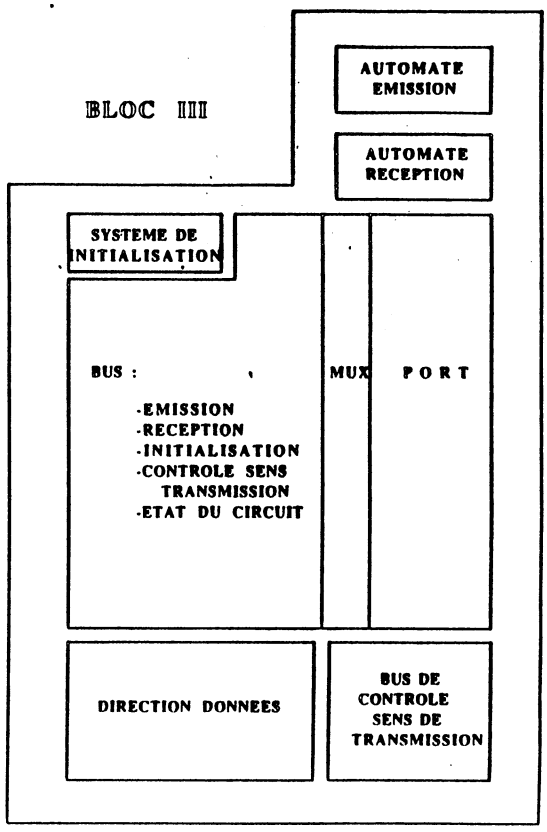


Figure 3.3.8

Enfin, le bloc IV correspond à la partie contrôle du circuit, et comme il a déjà été dit, elle réalise la méthode d'accès du circuit. Ce bloc est aussi constitué par les modules fortement liés à la partie opérative (registre d'état, un décompteur et une série d'amplificateurs pour plusieurs signaux de sortie). La figure 3.3.9 illustre ces modules.

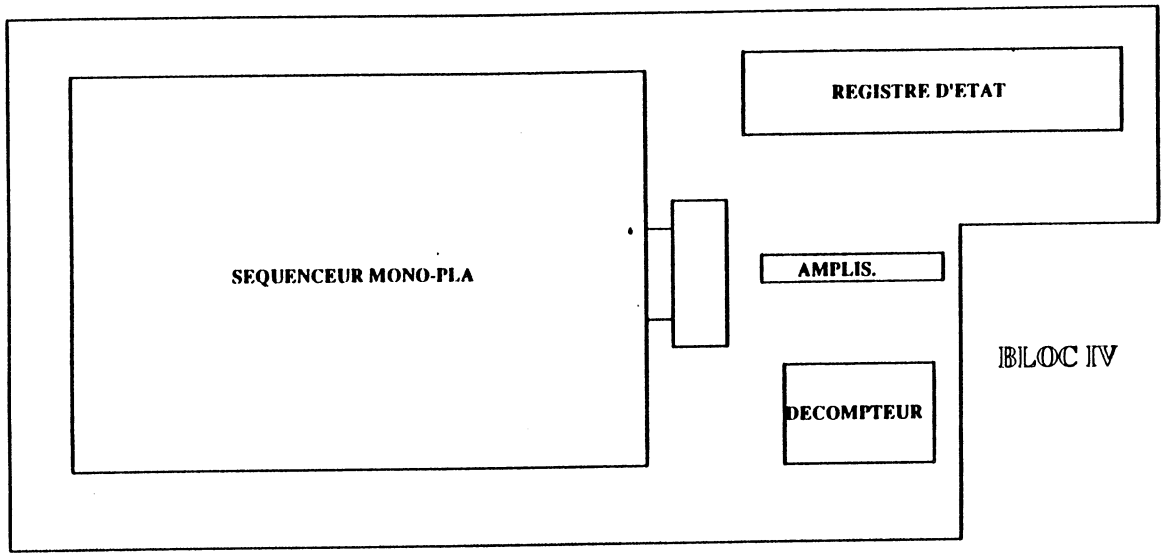


Figure 3.3.9

* Routage des blocs

Globalement, le routage du circuit a été effectué d'une manière manuelle, à l'exception de quelques modules qui ont été interconnectés à l'aide de LOF. L'idéal aurait été d'utiliser un routeur automatique.

CONCLUSIONS

CONCLUSIONS

1.- Un des enseignement que nous avons pu tirer sur cette expérience de réalisation d'un circuit VLSI à la demande à partir d'une bibliothèque d'opérateurs flexibles est la diminution considérable du temps de conception. Mais, si nous voulons diminuer encore ce temps il est impératif d'avoir des outils pour le routage automatique des modules. Ceci permettra au concepteur, en plus du temps de conception gagné, d'avoir la possibilité d'essayer différents placement topologiques des modules, avec comme but de chercher à diminuer la surface occupée. Un autre avantage pour le concepteur sera l'élimination des erreurs dûes aux connexions, lesquelles sont très fréquentes en routage manuel.

Un avantage de la méthodologie dédiée à une classe de circuits est de pouvoir définir les modules fonctionnels caractéristiques et de les répertorier dans une bibliothèque. L'exemple du circuit FIP a permis de mettre en relief la construction d'une bibliothèque d'opérateurs flexibles et les opérateurs qui peuvent la constituer. Mais, il est aussi possible d'envisager la création de modules "figés" qui réalisent des fonctions bien spécifiques aux circuits de communication, par exemple les fonctions de codage et décodage des informations ou bien même des fonctions propres au circuit, par exemple le module de génération de la base de temps.

Le résultat de notre étude est montré dans la photographie ci-contre du circuit. Les bibliothèques de cellules et d'opérateurs flexibles fonctionnent en technologie NMOS L3 du CNET-CNS.

Un défaut de la méthodologie proposée est de refaire la bibliothèque de cellules à chaque changement de technologie.

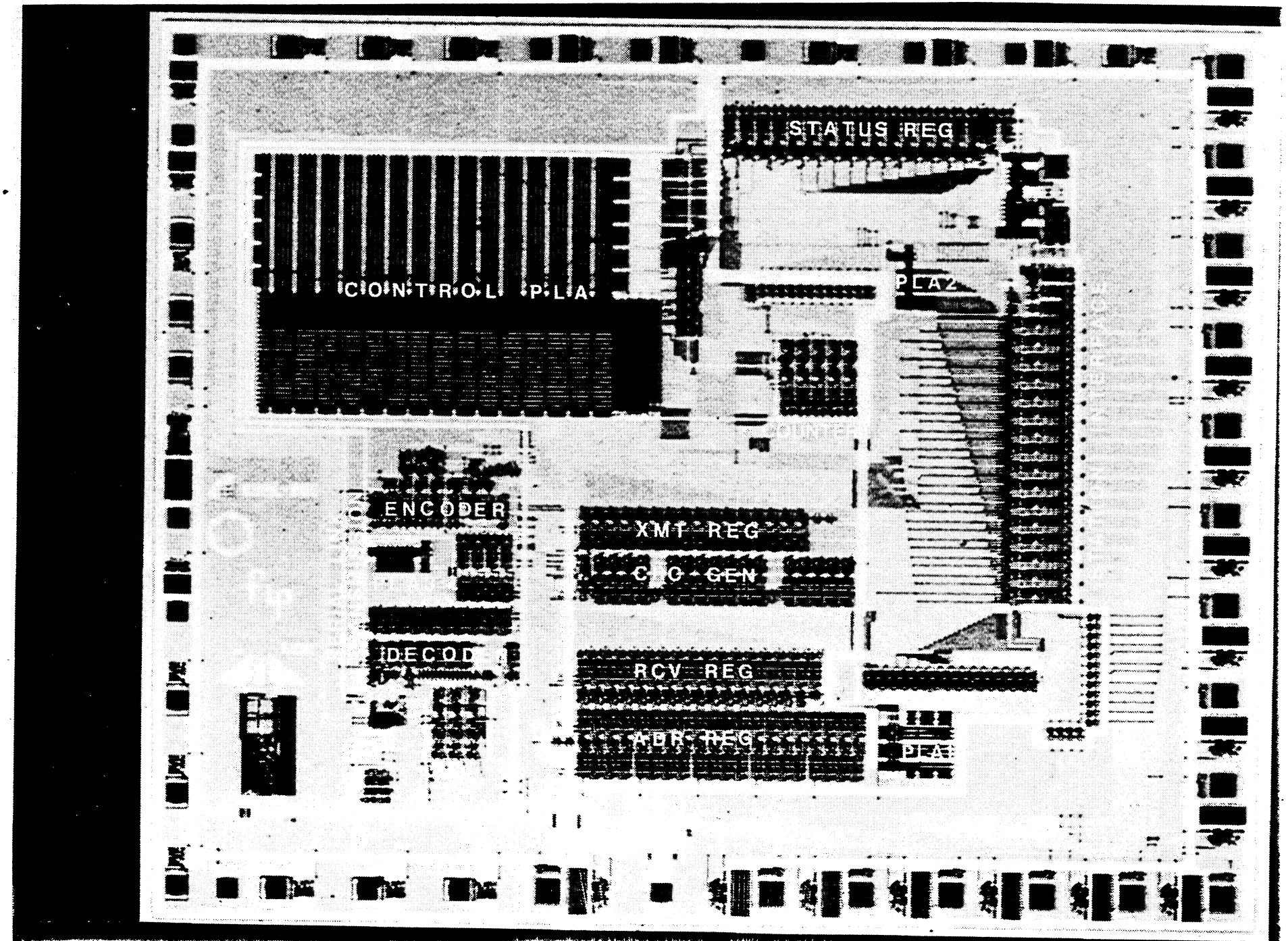
2.- La surface occupée par le circuit FIP-VLSI est d'environ 38 mm2. Bien qu'elle soit un peu grande, il faut indiquer qu'aucun compactage n'a été effectué sur le dessin des masques du circuit. On estime qu'un simple compactage pourrait réduire d'environ 20% la surface du circuit. Cette solution n'est pas la seule pour obtenir une réduction de la surface. Il est possible de gagner environ 10% par l'utilisation complète des transparences des opérateurs. Ainsi, pensons nous qu'il est possible de réduire sans difficulté de 25% la surface totale du circuit, ce qui correspond déjà à un circuit de dimension plus raisonnable compte tenu de la méthode d'implantation.

La consommation du circuit est d'environ 500 mWatts.

3.- Bien que le communicateur FIP-VLSI ne soit pas un circuit complexe (environ 15.000 transistors) pour un circuit de ce genre, il réalise plusieurs fonctionnalités intéressantes à retenir: la possibilité de permettre la connexion des capteurs ou actionneurs simples, celle également de permettre la connexion des stations intelligentes, automates réflexes comprises. Signalons par ailleurs

la prise en compte des fonctionnalités globales du couple composant-station (type d'adressage facilitant la maintenance, l'insertion dynamique d'un couple composant-station, ...)

Le domaine des circuits de communication est un domaine en constant développement. Le savoir-faire aussi bien en architecture qu'en conception de ce type de circuit est loin de celui obtenu équivalent pour les microprocesseurs ou les circuits de traitement du signal. Ce circuit, qui est un circuit VLSI réalisé par une équipe de recherche, destiné à tester des solutions et des concepts nouveaux, n'est qu'une contribution aux efforts dans ce domaine, sachant que ce domaine s'étend rapidement aux différentes couches du modèle OSI.


STATUS REG
CONTROL PLA
PLA2
ENCODER
XMT REG
C C GEN
DECOD
RCV REG
ADR REG
PLA

BIBLIOGRAPHIE

REFERENCES BIBLIOGRAPHIQUES

[ANC-83] Capri: A Design Methodology and a Silicon Compiler for VLSI Circuits Specified by Algorithms
François Anceau
R.R. no 346, Rapport de Recherche, IMAG, 14 Janvier 1983

[BEY-82] A Design Methodology Based Upon Symbolic Layout and Integrated Circuits CAD Tools
A.M. Beys, B. Hennion, J. Lecourvoisier, G. Mazaré, A. Puissochet
Proceedings DAC. Las Vegas, Nevada, Juin 1982

[BER-84] A Building Tool for Flexible Blocks Library
J.M. Bergé, J. Rouillard, L.O. Donzelle, D. Rouquier
Proceedings ICCD. Port Chester, New York, Octobre 1984

[BIA-81] Implantation de la Logique Aléatoire d'un Circuit Intégré
R. Bianchi
Rapport de D.E.A. Microélectronique , ENSIMAG , Juin 1981

[BOU-85] La Conception de Circuits Intégrés Silicium au CNET
J. Boulvin, A. Girard, J.-L. Lardy
L'écho des Recherches no. 121 3ème trimestre 1985.

[CAN-86] Conception des Circuits Intégrés MOS, Eléments de Base-Perspectives
M. Cand, E. Demoulin, J.-L. Lardy, P. Senn
ED : Eyrolles, Paris, 1986

[CAS-84] CASSIOPEE
Note technique /CNS/CCI/ 27. Mai 1984.

[CLA-78] An Introduction to Local Area Network
David D. Clark, Kenneth T. Pogran, David P. Reed
Proceedings of the IEEE, Novembre 1978
vol 66, no 11, pp 54-56

[CNE-85] Fiche d'Etudes No. 2
CCI/CNET-CNS, 1er. Janvier 1985.

[CON-82] Implantation Symbolique MDMOS
A. Structure et Caractéristiques Electriques, Topologie et Représentation
B. Conq
Note Technique NT/CNS/CCI/03, CNET-Grenoble, Mai 1982

références bibliographiques

[CON-82] Implantation Symbolique MDMOS
B. Notice Utilisateur
B. Conq
Note Technique NT/CNS/CCI/04, CNET-Grenoble, Septembre 1982

[CON-83] Implantation Symbolique en N MOS
C - Compléments
B. Conq, J. Majos
Note Technique NT/CNS/CCI/ 20, CNET-Grenoble, Mai, 1983

[COR-81] Systèmes Informatiques Répartis Concepts et Techniques
Cornafion
Ed: Dunod Spécialité Informatique, Paris, 1981

[CRA-82] CRASH: Une Méthode de Conception Symbolique pour le NMOS
A. Roset
Note Technique NT/CNS/CCI/05, CNET-Grenoble, Juillet 1982

[CSM-85] 802.3 Carrier Sense Multiple Access with Collision Detection
ISO/Dis 8802/3, ANSI/IEEE Std. 802.3, 1985
Published by IEEE

[CUS-84] Customizing VLSI Integrated Circuits
A User's Guide to The Selection an Design of Gate Array, Standard Cell and Full Custom Circuits
Independent Resource Corporation, 1984

[CVT-86] CAD-VLSI for Telecommunication CVT-Projet
Abstract of the Draft Final Report, Mars 1986

[DAN-83] Small Local Area Network with a Collision-Free Technique for Connecting Heterogeneous Microcomputer Systems
Ng.X. Dang, M. Diaz Nava, N. Husovic
Microprocessing and Microprogramming
Euromicro-Journal Octobre-Novembre 1983
vol-12, no 3-4, pp 167-174

[DAN-85] Circuits Intégrés VLSI dans les Systèmes de Communication de Données
M. Dang, M. Diaz Nava
Actes du Congrès de Nouvelles Architectures pour les Communications, Ed: Eyrolles, Paris, Octobre 1985, pp 181-189

[DAN-86] Design of a VLSI Communicating Circuit for an Industrial Local Network for Process Control and Automated Production
M. Dang, M. Diaz Nava, G. Michel
EuroMicro.86, Venise, Italie, Septembre 1986

[DAY-83] The OSI Reference Model
John D. Day, Hubert Zimmermann
Proceedings of IEEE, Décembre 1983
vol. 71, no 12, pp 1334-1341

[DEN-82] FIRST: A Silicon Compiler for VLSI Signal Processors
P.B. Denyer, D. Renshaw, N. Bergmann
Proceedings of ESSIRC 1982

[DEP-85] Architecture Experimentation
Gary F. De Palma
VLSI Systems Design, November 1985

[DIA-82] Implantation d'un Réseau Local de Type Bus pour Micro-Ordinateurs Hétérogènes
M. Diaz Nava
Rapport DEA. ENSIMAG, Juin 1982

[DIA-86] VLSI Specification and design for a process control LAN controller
M. Diaz Nava, NG. X. Dang.
IFIP International Symposium on Local Communications Systems: LAN and PBX Toulouse, Novembre 1986

[DUB-85] Les Capteurs Intelligents
A Dubail, J.M. Favennec
APIST Journées AFCET. CIAME. 15 octobre 1985. Paris.

[DUY-83] Structured-Design Systems Takes Over The Complexities in VLSI circuits
Bob DUYN, Stephen Timberger
Electronics, 25 Août 1983

[ESW-81] Collision-Free Access Control for Computer Communication Bus Networks
K.P. Eswaran , V.C. Hamacher, G.S. Shedler
IEEE Transactions on Software Engineering,
Vol SE-7, 1981

[FIL-81] Méthode d'Implantation de la Logique Aléatoire
M. Fillon
Rapport de D.E.A. Microélectronique, ENSIMAG , Juin 1981

[FIP-84] FIP : Proposition d'un Système de Transmission Série Multiplexée pour les Echanges d'Informations entre des Capteurs, des Actionneurs et des Automates Réflexes
Livre Blanc. Ministère de la Recherche et de l'Industrie. France. Avril 1984.

références bibliographiques

[FLA-84] A Complete & Automatic System for Sequencer Design
E. Flamand.
Proceedings ICCD. Port Chester, New York, Octobre 1984.

[GAJ-83] New VLSI Tools
Daniel D. Gajski , Robert H. Kuhn
Computer , Décembre 1983

[GUE-84] Broadcasting Real-time Data in Source Addressed Messages
R.C. GUETH, J. KRIZ, S. ZUEGER
Real-Time Systems Symposiun. Austin, Texas. Décembre 1984.

[HEN-80] CASSIOPEE: An Integrated CAD System for Integrated Circuits
B. Hennion, G. Mazaré
Proceedings of the ESSCIRC 80. Grenoble, Septembre 1980.

[HEN-85] A New Algorithm for Third Generation Circuit Simulators: The One-Step Relaxation Method
B. Hennion, P. Senn, D. Coquelle
Proceedings DAC. Las Vegas, Nevada, Juin 1985.

[INT-83] 82586 LAN Communications Controller
Intel, 1983

[INT-85] The BITBUS Interconnect Serial Control Bus Specification
INTEL 1985.

[ISO-83] ISO, Basic Reference Model for Open Systems Interconnection
ISO 7498, 1983

[JAN-84] 802.5 Token Ring Access Method and Physical Layer Specifications
ISO/TC 97/SC 6, IEEE Draft Standard 802.5, August 1984
Ed: IEEE

[JBU-85] 802.4 Token Passing Bus Access Method
ISO/Dis 8802/4, ANSI/IEEE Std. 802.4, 1985
Ed: IEEE

[JOH-79] Bristle Blocks: A Silicon Compiler
D. Johannsen
Proceedings 16th Design Automation Conference, 1979

[KAM-86] Protocols for Communicating in the Factory
Michael A. Kaminski Jr.
IEEE Spectrum, Avril 1986, pp 56-62

[KUN-82] Why Systolic Architectures ?
H. T. Kung
Computer vol. 15, no. 1, Janvier 1982, pp 37-47

[LEB-80] Automatic Layout of Symbolic MDMOS Circuits
A. Leblond, G. Serrero, A. Verdillon
IEEE 1980 International Conference on Circuits and Computers for Large Scale Systems, New York, Octobre 1980

[LEL-83] On Real-Time Distributed Computing
G. Le Lann
Proceedings IFIP 83. Paris, Septembre 1983.

[LET-84] Les Réseaux Locaux Industriels
Lettre d'Information Réseaux, no 13, Décembre 1984.

[LIN-85] TI Antes up Its Chips for IBM Token Ring LAN
J. Robert Lineback, Clifford Barney
Electronics, 15 Juillet 1985

[LIV-76] Livre Orange
Sixième Assemblée Plénière du CCITT'
Genève, Octobre 1976

[LLC-85] 802.2 Logical Link Control
ISO/Dis 8802/2, ANSI/IEEE Std. 802.2, 1985
Ed: IEEE

[LOU-83] The Micro VAX I Data-Path Chip
Glenn Louis, Tom (Iu-meng) Ho, Ed Cheng
VLSI Design, Décembre 1983

[MAC-79] Téléinformatique
C. Macchi, J.-F. Guilbert
Ed: Dunod spécialité informatique, Paris,1979

[MAG-85] Controller IC Contends with Multiple Protocols
Jim Magill, George Wong
Computer Design. Février 1985

[MAN-77] Manuel Technique Transpac
Spécifications Techniques d'Utilisation du Réseau,
Diffusé par Transpac, Paris, Septembre 1977

[MEA-80] Introduction To VLSI Systems
Carver Mead, Lynn Conway
Addison Wesley , Octobre 1980

références bibliographiques

[MOT-82] Microprocessors Data Manual 1982
Motorola Semiconductor

[NAN-83] Cell-Layout Compilers Simplify Custom-IC Design
Scott Nance, Chris Starr, Bob Duyn, Mike Kliment
EDN, 15 Septembre 1983

[NS-80] Transmission Line Drivers and Receivers for EIA Standards RS-422 and RS-423
Interface Data Book
National Semiconductor 1980, pp 10.66-10.90

[OSI-79] ISO TC 97. SC P6, Reference Model of Open Systems Interconnexion
ISO 1979

[PAN-84] PROCONTROL P, un Système de Controle-Commande Moderne pour Centrales
L. M. PANIS, M. SALM
Revue Brown Boveri no 8 tome 71, Août 1984.

[PET-76] Error Correcting Codes
W. W. Peterson, E.-J. Weldon
Ed: The MIT Press.

[QUI-83] The Systematic Design of Systolic Arrays
P. Quinton
Publication interne IRISA, no. 193, Mars 1983.

[ROL-85] Les Circuits Périphériques de Communication de Données de la Famille 68000
Jean-Marie Rollain
Electronique Industrielle no. 86, 1er. Avril 1985

[ROU-86] La Compilation de Silicium
D. Rouquier
Seminaire CNET-CNS, 30 Mai 1986

[REI-83] TESS : Evaluateur Topologique Prédictif pour la Génération Automatique des Plans de Masse de Circuits VLSI
Ricardo Augusto Da Luz Reis
Thèse Docteur Ingénieur INPG Grenoble, Janvier 1983

[REY-85] IMHOTEP : Un Générateur Automatique d'Architectures pour Circuits Intégrés de Filtrage Numérique
Jean-Fréderic Reyss-Brion
Thèse de Docteur de l'INPG Grenoble, Mai 1985

[SHO-80] An Annotated Bibliography on Local Computer Networks
J.K. Shock
Xerox, Palo Alto Research Center, Avril 1980

[SOU-83] MacPitts: An Approach to Silicon Compilation
Jay R. Southard
Computer , Décembre 1983

[STR-85] Hybrid Trio Takes Charge of The Bus-to-CPU Interface in Military Avances Gear
E. Strachar. Electronic Design. 4 Août 1985.

[SUZ-81] Etude des Parties Operatives à Eléments Modulaires pour Processeurs Monolithiques
Altamiro Amadeu Suzim
Thèse Docteur Ingénieur INPG Grenoble, Novembre 1981

[THA-84] Functional Modelling for Logic Simulation
H.El Tahawy, G. Mazaré, M. Poize, A. Puissochet
Proceedings ICCD. Port Chester, New York, Octobre 1984.

[THO-83] Automatic Data Path Synthesis
Donald E. Thomas, Charles Y. Hitchcock III, Thaddeus J. Kowalski, Jayanth V. Rajan, and Robert A. Walker
Computer, Décembre 1983

[VOE-86] OSI : Helping Computers Communicate
John Voelcker
IEEE Spectrum, Mars 1986, pp 61-7]

[WIT-85] Choosing The Best Local Area Network for any Application
Richard I. Wittlin, Daniel V. Ratner
Computer Design, Février 1985, pp 143-149

[ZIL-84] Zilog Data Book 1984

ANNEXES

ANNEXE A

LES AVANTAGES DE L'INTEGRATION

AVANTAGES DE L'INTEGRATION [CUS-84]

Dans toute industrie qui fabrique des dispositifs électroniques ou qui les incorpore dans ses produits, les circuits VLSI permettent d'augmenter ses marges de bénéfices et sa part du marché. Dans beaucoup de domaines, l'utilisation des circuits intégrés à la demande ou circuits personalisés est devenue nécessaire pour maintenir une position de compétitivité.

Les avantages des circuits VLSI dérivent de la possibilité d'intégrer un circuit électronique équivalent à des centaines de milliers de composants discrets dans une toute petite puce de silicium. Ceci conduit à une réduction important de la taille, du poids, de la consommation et des coûts des circuits électroniques, et une augmentation de leur fiabilité et de leur vitesse de fonctionnement.

La figure A.1 illustre l'accroissement de la densité d'intégration résultant de l'amélioration des techniques utilisées dans la conception et la fabrication des circuits intégrés.

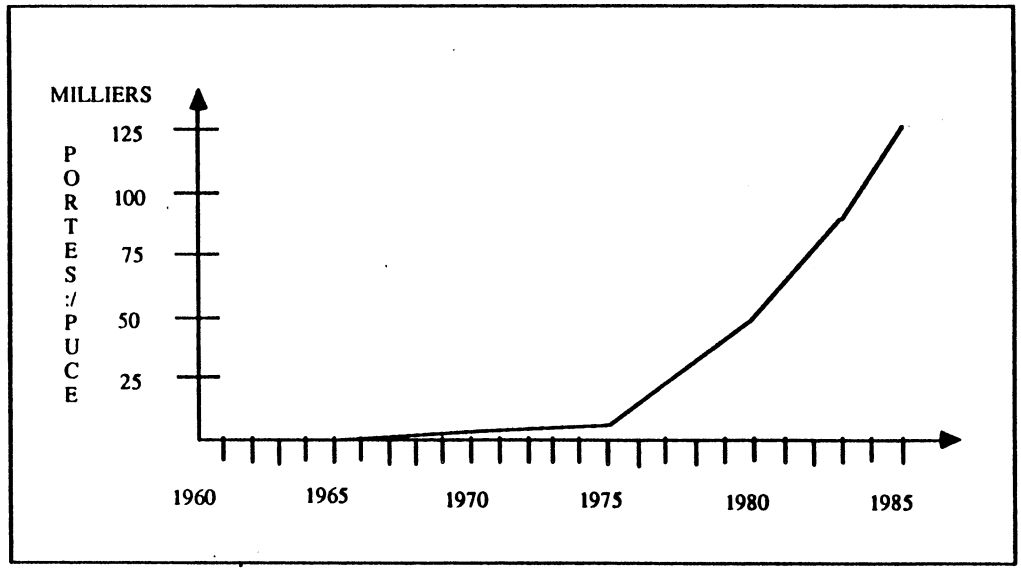


Figure A.1 Niveaux d'intégration

Les bénéfices économiques obtenus par l'utilisation des circuits VLSI peuvent être substantiels. Certaines études estiment que chaque circuit intégré SSI/MSI ("Small-Scale Integration/ Medium-Scale Integration) sur une carte (ou le nombre de composants discrets occupant la même surface) représente un coût total de $10 à $15. Une carte normale contient 20 de ces dispositifs, et elle peut être réduite à un seul circuit VLSI avec un coût unitaire de $15; il est évident que les seules économies dans les coûts de production sont énormes.

Annexe A

Le temps et les coûts de développement peuvent être aussi considérablement réduits par l'emploi des circuits VLSI, lorsque les outils de CAO éliminent complètement le besoin de la construction de prototypes et de maquettes. Les possibilités de simulation des circuits par ordinateur assurent une grande sûreté de conception, alors que la diminution de la consommation et la haute densité d'intégration des VLSI accroissent d'une manière importante la fiabilité du produit terminé.

ANNEXE B

CARACTERISTIQUES DE PLUSIEURS CIRCUITS D'ENTREE/SORTIE SERIE

CARACTERISTIQUES GENERALES DE QUELQUES CIRCUITS D'ENTREE/SORTIE SERIE

CONSTRUCTEUR	Motorola	Motorola	Motorola	Intel	Intel	Mostek	Zilog	Zilog	RTC	Signetics
Circuit	6850 ACIA	6852 SSDA	6854 ADLC	8251 PCI	8273 PPC	68564 SIO	Z80 SIC	8030 / 8530	2652 MPCC	68562
N° de broches	24	24	28	28	40	48	40	40	40	48
Technologie	MOS canal N	MOS canal N	MOS canal N	MOS canal N	MOS canal N	MOS canal N	MOS canal N	H MOS	MOS canal N	H MOS
Alimentation	5 volts	5 volts	5 volts	5 volts	5 volts	5 volts	5 volts	5 volts	5 volts	5 volts
Mode de Tx.	async	syn. COP	syn. BOP	asyn/syn	syn. BOP	asyn/syn	asyn/syn	asyn/syn	syn. BOP,..	asyn/syn
N° de canaux	1	1	1	1	1	2	2	2		2
Débit max de tx.	1 Mb/s	1.5 Mb/s	1.5 Mb/s	64 Kb/s	1 Mb/s	1Mb/s	800Kb/s	1Mb/s	2Mb/s	4Mb/s
Cycle d'horloge min.	500 ns	500 ns	500 ns	250 ns	250 ns	200 ns	250 ns	165 ns		
Oscillateur incorporé	-	-	-	-	-	oui	-	oui	-	oui
PLL digitale	-	-	-	-	oui	-	-	oui (2)	-	oui (2)
Encodage de données	-	-	NRZ, NRZI	-	NRZI	-	-	NRZ,NRZI,..	-	NRZ,NRZI,..
Générateurs de bauds	-	-	-	-	-	-	-	2	-	4
Horloges externes	2	2	2	2	2	3	3	1	2	2
Contrôle de modem	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake
Protection des erreurs	Parité	Parité	CRC	Parité	CRC	CRC,CCITT	CRC	CRC,CCITT	CRC,CCITT..	CRC,CCITT..
N° de tampons de Tx/Rx	2/2	4/4	4/4	2/2	2/2	2/4	2/4	2/4	2/2	4/4
Interface microprocesseur	6800, 6809	6800, 6809	Fam. 6800	8080, 8085	Fam. 80XX	68000	Z 80	Fam:Z,I,MC	2650,Z,MC	68000
Bus de données	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8,16 bits	8 bits
Adressage reg. internes	Direct	Direct	Direct	Pointeur	Direct	Direct	Pointeur	Direct/Pointe	Direct	Direct
Vecteur d'interruptions	-	-	-	-	-	oui	oui	oui	-	oui
Priorité interruptions	-	-	oui	-	-	oui	oui	oui	-	oui
Prog. interruptions	non	oui	non	non	oui	oui	oui	oui	-	oui
DMA, N° de canaux	-	-	2	-	2	4	4	4	2	4
Contrôleur nécessaire	-	-	MC 6844	-	8257	MC 68450	Z80 DMA	68450, 8257	-	68450
Boucle SDLC	-	-	oui	-	oui	non	non	oui	non	oui
Multipoint	-	-	-	-	-	-	-	-	-	oui
Détection d'erreurs	oui	oui	oui	oui	oui	oui	oui	oui	oui	oui
Test miroir	-	-	oui	-	oui	-	oui	oui	oui	oui
Test à distance	-	-	-	-	-	-	oui	-	-	oui

TABLEAU I

PRINCIPALES FONCTIONS REALISEES POUR QUELQUES CIRCUITS DANS UNE TRANSMISSION ASYNCHRONE

CONSTRUCTEUR	Motorola	Intel	Mostek	Zilog	Zilog	Signetics
Circuit	6850 ACIA	8251 PCI	68564 SIO	Z80 SIO	Z8030Z8530	68562
Longueur caractères	7, 8 bits	5 à 8 bits	5 à 8 bits	5 à 8 bits	5 à 8 bits	5 à 8 bits
Longueur bits d'arrêt	1,2	1,1 1/2, 2	1,1 1/2, 2	1,1 1/2, 2	1,1 1/2, 2	1,1 1/2, 2
Bit de parité	oui	oui	oui	oui	oui	oui
Générateur de break	-	oui	oui	oui	oui	oui
Horloges externes	:1,:16,:64	:1,:16,:64	:1,:16,:32:64	:1,:16,:32:64	:1,:16,:32:64	:1,:16
Dét. erreur parité	oui	oui	oui	oui	oui	oui
Dét. mauvais bit de dép.	oui	oui	oui	oui	oui	oui
Dét. erreur bourrage	oui	oui	oui	oui	oui	oui

Tableau II

PRINCIPALES FONCTIONS REALISEES PAR QUELQUES CIRCUITS DANS UNE TRANSMISSION ORIENTEE CARACTERE

CONSTRUCTEUR	Motorola	Intel	Mostek	Zilog	Zilog	RTC	Signetics
Circuit	6852 SSDA	8251 PCI	68564 SIO	Z80 SIO	Z8030Z8530	2652 MPCC	68562
Protocole	BSC*	BSC*	BSC	BSC	BSC	BSC	BSC
Prog. longueur cars.	7,8 ou 9 bits	5 à 8 bits	1 à 8 bits	5 à 8 bits	5 à 8 bits	5 à 8 bits	5 à 8 bits
Prog. car. de synchro	oui	oui	oui	oui	oui	oui	oui
Synchro sur 1 et 2 car.	oui/oui	oui/oui	oui/oui	oui/oui	oui/oui	non/oui	oui/oui
Insertion/sup. de syn	oui	oui	oui	oui	oui	oui	oui
Indication dét. synchro	oui	oui	oui	oui	oui	oui	oui
Synchro externe	oui	oui	oui	oui	oui	non	oui
Contrôle d'erreurs	parité 9° bit	parité	CRC:16,CCITT	CRC:16,CCITT	CRC:16,CCITT	CRC16	CRC16
Dét. erreur parité	oui	oui	-	-	-	oui	-
Dét. erreur bourrage	oui	oui	oui	oui	oui	oui	oui
Dét.erreur sur-cadence	oui	oui	oui	oui	oui	oui	oui

TABLEAU III

PRINCIPALES FONCTIONS REALISEES POUR QUELQUES CIRCUITS DANS UNE TRANSMISSION SYNCHRONE ORIENTEE BIT

CONSTRUCTEUR	Motorola	Intel	Mostek	Zilog	Zilog	RTC	Signetics
Circuit	6854 ADLC	8273 PPC	MK68564	Z80 SIO	Z8030Z8530	2652 MPCC	68562
Protocoles	HDLC,SDLC,ADCCP	HDLC,SDLC	HDLC,SDLC	HDLC,SDLC	HDLC,SDLC	HDLC,SDLC,ADCCP	HDLC,SDLC,ADCCP
Compatible X25	-	oui	-	oui	oui	-	oui
Mode boucle SDLC	oui	oui	-	oui	oui	-	oui
Stockage adr. dest.	-	oui	oui	oui	oui	oui	oui
Adressage étendu	oui	oui	oui	oui	oui	-	oui
Contrôle étendu	oui	oui	-	-	-	-	oui
Comp. adr.secondaire	-	-	-	-	oui	oui	oui
Rec. adr. de groupe	-	-	-	-	-	-	oui
Rec. adr. globale	-	oui	oui	oui	oui	-	oui
Traite.de car.residuels	oui	-	oui	oui	oui	-	oui
Insertion/suppre.de"0"	oui	oui	oui	oui	oui	oui	oui
Gén/dét auto. fanions	oui	oui	oui	oui	oui	oui	oui
Contrôle d'erreurs FCS	CRC-CCITT	CRC	CRC:16,CCITT	CRC	CRC:16,CCITT	CRC-CCITT	CRC-CCITT
Gén.:abort. idle	oui	oui	oui	oui	oui	oui	oui
Reset trames courtes	oui	oui	-	-	-	-	oui

Tableau IV

ANNEXE C

OUTILS D'AIDE A LA CONCEPTION

PRESENTATION DES OUTILS D'AIDE A LA CONCEPTION UTILISES

Les outils CAO de base utilisés pour l'implantation de la bibliothèque de cellules et puis pour les opérateurs flexibles, ont été : CASSIOPEE et CALMA.

1 CASSIOPEE

CASSIOPEE est un système intégré de conception de circuits intégrés développé par le CNET, la figure C.1 montre sa structure générale. Il inclut:

. une base de données,

. un éditeur graphique SHEDIR,

. plusieurs types de simulateurs pour les différents niveaux d'abstraction; simulateurs: électrique, logique et fonctionnel.

. plusieurs types de vérificateurs: de DRC et de connectique,

. une chaîne de génération de PLAs, et

. plusieurs programmes utilitaires.

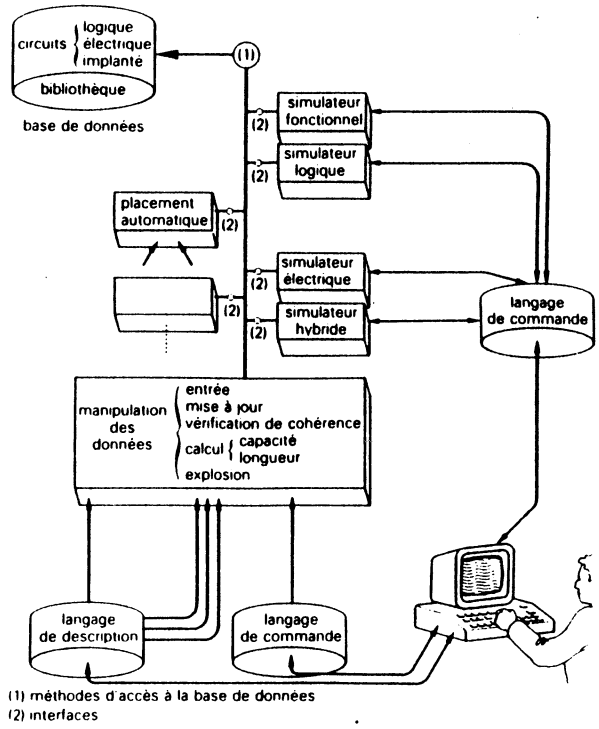


Figure C.1 Structure générale de CASSIOPEE.

Annexe C

1.1 * La base de données

Dans la base de données, un circuit est défini par une entité type à laquelle sont attachées deux descriptions :

. une description fonctionnelle qui rassemble toutes les caractéristiques du circuit (nombre des entrées/sorties, paramètres logiques, électriques, graphiques,.....),

. une description structurelle qui permet de définir le circuit en terme de blocs et de fils interconnectant ces blocs. Les blocs et les fils sont alors représentés respectivement dans la base de données par des entités composant et des entités équipotentielles.

Chaque entité composant (ou bloc) possède son propre type qui est défini, de la même façon que le circuit par une entité type.

Un circuit est donc complètement décrit dans la base de données par un ensemble hiérarchique d'entités types qui possèdent une description fonctionnelle et une description structurelle et ce pour plusieurs niveaux d'abstraction (correspondant aux différentes étapes du processus de conception).

1.2 *L'éditeur graphique SHEDIR*

L'entrée et la mise à jour des informations dans la base de données se font suivant un seul et même éditeur graphique SHEDIR. Il permet la prise en compte des différents niveaux de description: fonctionnel, logique, électrique, symbolique (bâton NMOS et CMOS, MDMOS), "micron". La figure C.2 montre l'implantation d'une cellule à l'aide d'un symbolisme bâton NMOS en utilisant l'éditeur SHEDIR.

Une des particularités de cet éditeur est qu'il maintient en permanence une cohérence entre les objets graphiques visualisés sur l'écran et les informations contenues dans la base de données (structurelles et graphiques). Il vérifie au fur et à mesure si les modifications demandées sont acceptables et provoque si nécessaire une correction immédiate.

1.3 *Les simulateurs*

CASSIOPEE inclut plusieurs types de simulateurs suivant les différents niveaux de description d'un circuit intégré :

. des simulateurs électriques: IMAG 3, SPICE 2, et ELDO ,

. un simulator logique EPILOG,

. un simulateur fonctionnel FIDEL.

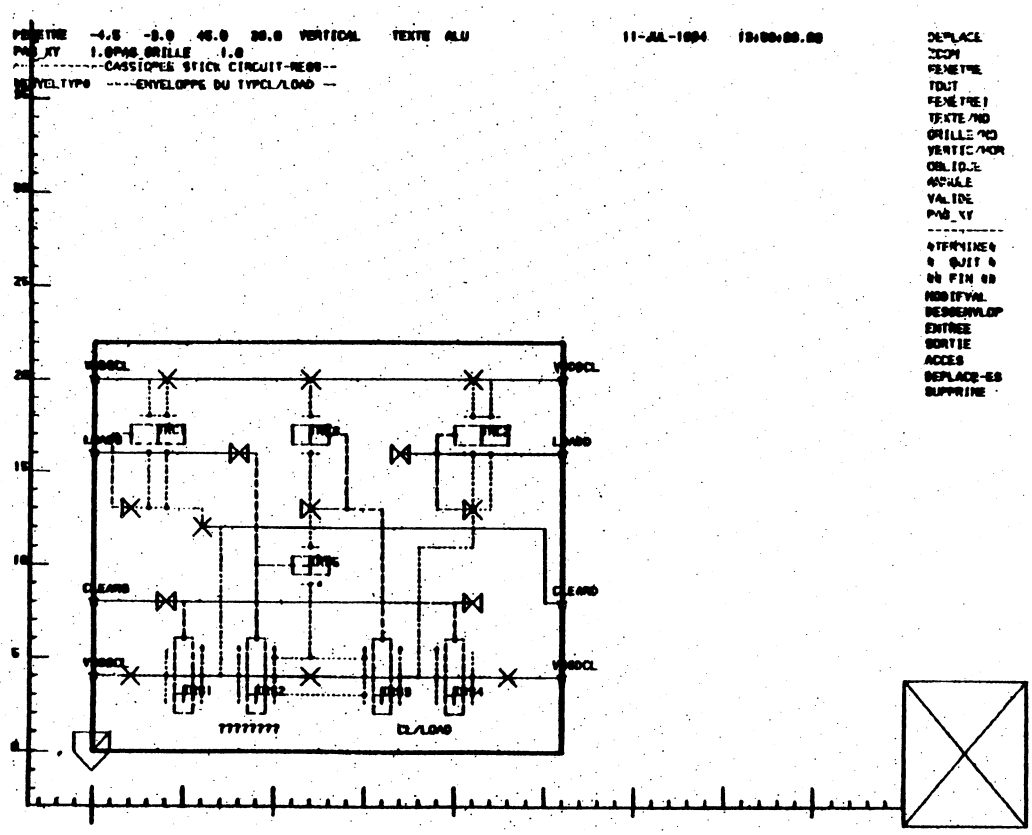


Figure C.2 Symbolisme bâton

* Le simulateur électrique ELDO

Le simulateur ELDO est utilisé pour vérifier le bon fonctionnement électrique des cellules et en général pour les parties critiques des circuits. Le fichier d'entrée de ce simulateur est directement généré par CASSIOPEE, à partir du schéma implanté en CRASH, par un programme d'extraction GENELDO. La figure C.3 montre le fichier d'entrée correspondant à la cellule de la figure C.2.

Cette méthode offre la possibilité de considérer dans le modèle du simulateur les capacités parasites introduites par les fils d'interconnexion et par concéquent d'avoir des simulations plus proches de la réalité.

Avant de réaliser cette simulation électrique un DRC sera effectué sur chacune des cellules ou des blocs dessinés.

```
(* 13-DEC-1984   DESCRIPTION ELDO DE ICLLOAD  *)
 %INCLUDE 'ELDOISTICK.PAS'
circuit(

 LOADG          ,LOADD          ,CLEARD         ,CLEAR

,n66            ,n71            )
(*TRS1     *)   tra(N,3, 1.50000E+01, 3.50000E+00,0                  ,n71          ,CLEARD       ,-2)
(*TRS2     *)   tra(N,3, 1.50000E+01, 3.50000E+00,CLEARD             ,LOADD        ,0            ,-2)
(*TRS3     *)   tra(N,3, 1.50000E+01, 3.50000E+00,0                  ,LOADD        ,LOADD        ,-2)
(*TRS4     *)   tra(N,3, 1.50000E+01, 3.50000E+00,LOADD             ,CLEARD       ,0            ,-2)
(*TRS5     *)   tra(N,3, 7.50000E+00, 3.50000E+00,n71               ,CLEAR        ,0            ,-2)
(*TRC1     *)   tra(N,1, 7.25000E+00, 7.50000E+00,-1                ,CLEARD       ,CLEARD       ,-2)
(*TRC2     *)   tra(N,1, 7.25000E+00, 7.50000E+00,-1                ,LOADD        ,LOADD        ,-2)
(*TRC3     *)   tra(N,1, 3.50000E+00, 7.50000E+00,-1                ,n71          ,n71          ,-2)
(*EQUIO    *)   cap(LOADG          ,0, 1.29100E-02)
(*EQUIVB   *)   cap(LOADD          ,0, 5.49356E-02)
(*EQUIDD   *)   cap(CLEARD         ,0, 5.42659E-02)
(*EQUIED   *)   cap(CLEAR          ,0, 1.12995E-03)
(*EQUITC   *)   cap(n71            ,0, 2.44480E-02)
(*--------------          10----------------------*)
(*FIN*)


(*********************************************)
```

Figure C.3 Description ELDO

* Le simulateur logique EPILOG

Il permet l'analyse du fonctionnement logique d'un bloc ou d'un circuit complet. Ce simulateur sera utilisé pour vérifier le bon fonctionnement logique des opérateurs en tranches à générer.

* Le simulateur fonctionnel FIDEL

Ce simulateur, comme tous ceux de son genre, permet la description d'un modèle fonctionnel d'un c.i., afin d'en faire une simulation qui permet de valider le ou les algorithmes proposés et d'obtenir des informations générales sur les performances du circuit modèlisé (c'est-à-dire l'évolution dans le temps des sorties en fonction des entrées).

Ce simulateur, en combinaison avec EPILOG, a été utilisé pour vérifier chacun des modules qui constituent le circuit FIP-VLSI, ainsi que le fonctionnement du circuit complet.

1.4 *La chaine de génération des PLAs*

Pour compléter les outils intégrés dans le système CASSIOPEE, il reste à mentionner une chaîne totalement intégrée qui permet de manipuler des blocs de type PLA. Partant d'une description fonctionnelle avec une entrée graphique ou textuelle (à partir de GASP), elle prend en compte les contraintes au niveau logique (disponibilité ou non du signal et du signal inverse,....) et au niveau implanté sur la topologie (position des entrées/sorties, rappel de masse, ...).

Remarque sur la génération du dessin des masques

La génération du dessin de masques "au micron", à partir de l'implantation symbolique, est réalisée par le système CALMA. Cette génération demande auparavant un transfert des informations sous bande magnétique à l'aide du programme GENCAL entre CASSIOPEE et CALMA.

2 LE SYSTEME CALMA

CALMA est un système graphique de CAO composé d'un calculateur et de plusieurs stations de travail (écran graphique plus table à numériser), qui gèrent la base de donnée graphique du circuit (ensemble de figures géométriques dans plusieurs niveaux). L'entrée des données peut se faire par menu à l'écran, ou par programme.

Dans le système CALMA du CNET-Grenoble plusieurs types de bibliothèques sont disponibles :

. pour le dessin "au micron" en technologie NMOS et CMOS,

. pour le dessin en symbolique bâton NMOS et CMOS,

. des programmes de génération automatique de dessin de masques à partir du symbolique bâton,

. des programmes de vérification de CRC

. l'implantation automatique des PLAs,

. il contient une bibliothèque d'amplificateurs compilables en NMOS(voir 3.5.2)
lm 10;

Enfin, le système permet la génération des bandes de masques dans le format GDS II propre à CALMA et dispose des logiciels de traitement pour les différents types de masqueurs (optiques ou électroniques). Avant ce traitement final une migration sélective des masques du circuit est réalisée ("resize")si nécessaire.

ANNEXE D

BIBLIOTHEQUE DE CELLULES

LA BIBLIOTHEQUE DE CELLULES

Toutes les cellules ont été implantées en symbolique bâton à grille fixe de pas 3.75 um et en respectant les règles de dessin définies pour une technologie NMOS 3um du CNET-Grenoble.

Critères d'implantation

1.- Nous avons décidé que les cellules implantées seraient du type statique.

2.- Plusieurs cellules ont été implantées avec une logique distribuée (COMPAR et les portes NOR). Pour la partie de precharge deux options ont été choisies:

. l'inclusion de la précharge dans la première cellule de la chaîne, lorsque les cellules sont différentes,

. dans le dessin des cellules, il a été prévu de laisser la place pour une cellule de précharge (dans ce cas, un transistor MOS à déplétion).

3.- Enfin, il reste à indiquer que la cellule de mémorisation utilisée est un point de stockage "LATCH". Cette cellule de registre a été choisi d'une part pour bien correspondre au type d'opérateurs à implanter et d'autre part par le nombre faible de transistors à dessiner. Vu la répétitivité de la cellule cette dernière considération peut être traduite par une optimisation de la surface à utiliser.

Présentation de la bibliothèque de cellules

Chacune des cellules qui vont être présentées ci-dessous comporte les informations suivantes:

. une brève description sur la fonction réalisée,

. le nom implanté de la cellule,

. les schémas : logique, électrique, implanté bâton et implanté "micron",

Remarque: Les cellules présentées sont un sous-ensemble des cellules dessinées.

Cellule : Amplificateur

nom : AMP1

Schéma logique

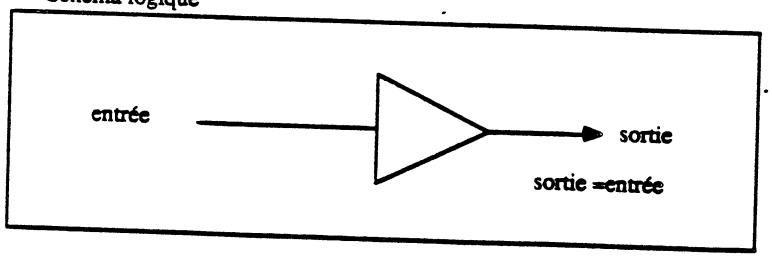
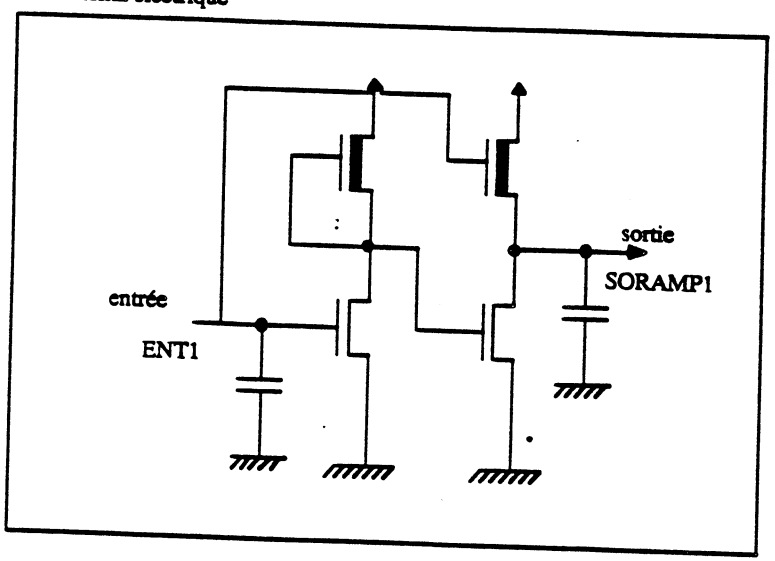
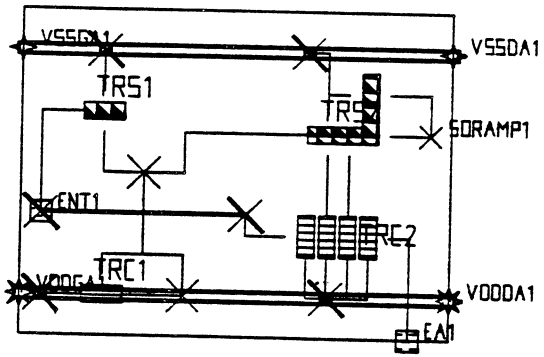
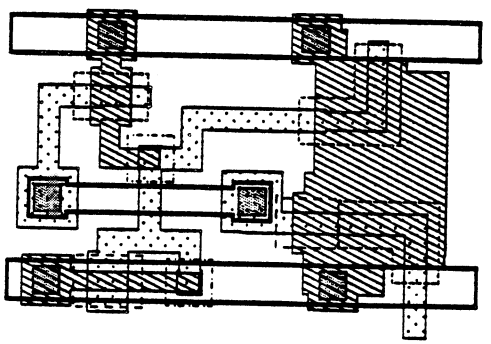
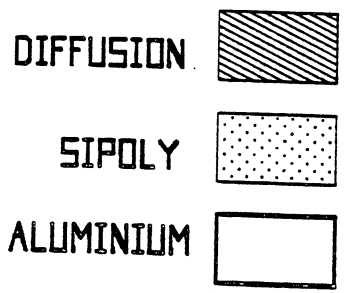


Schéma électrique









Cellule : Amplificateur inverseur

nom : AMPINV1

Schéma logique

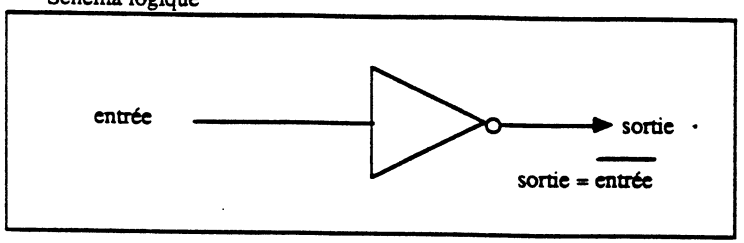
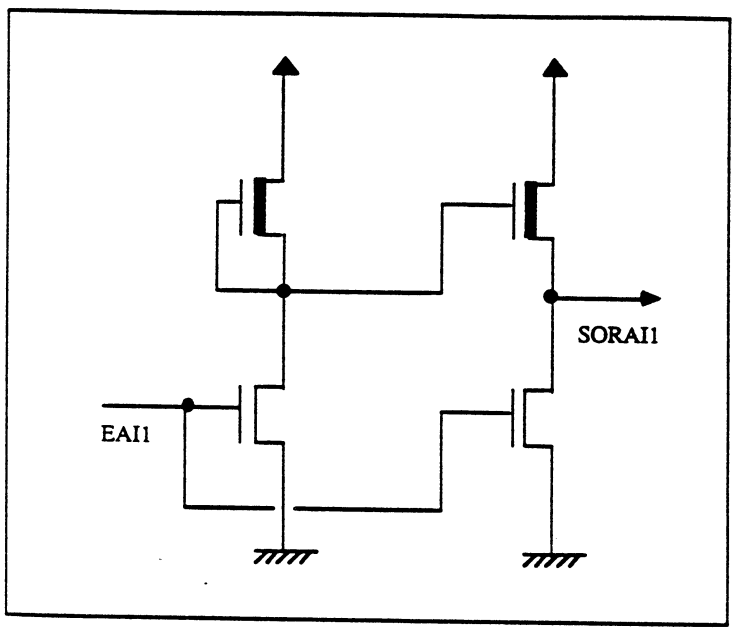
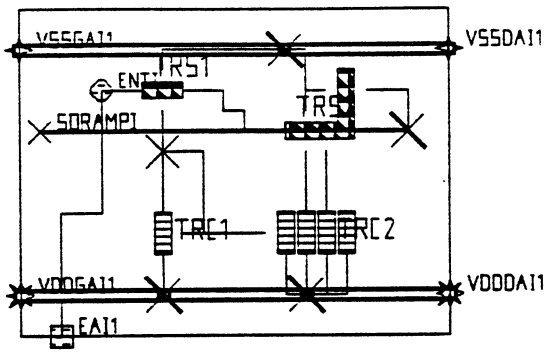
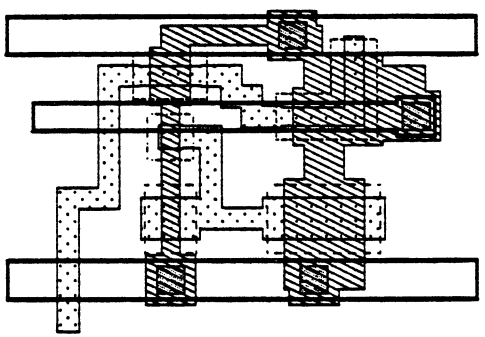


Schéma électrique







Cellule : Porte ET

nom : AND

schéma logique

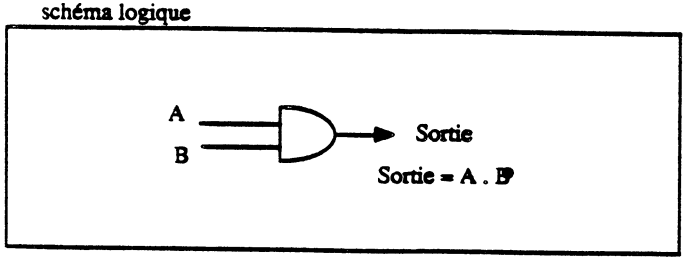
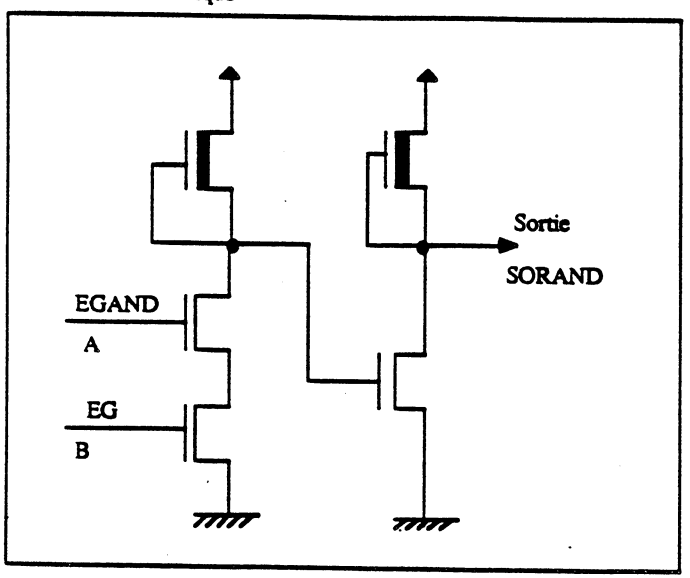
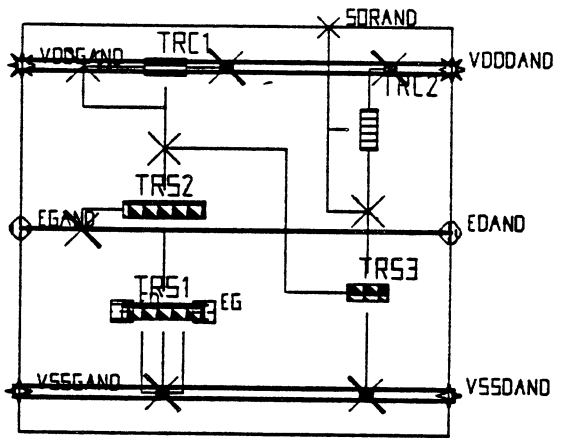
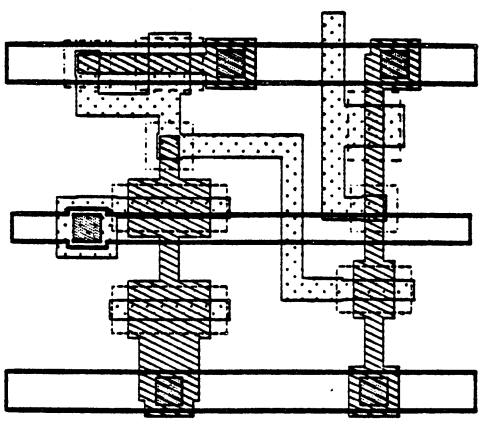


schéma électrique



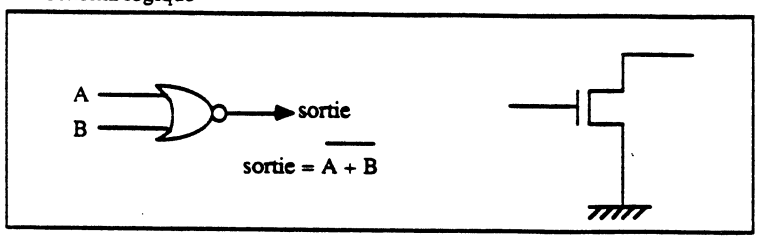


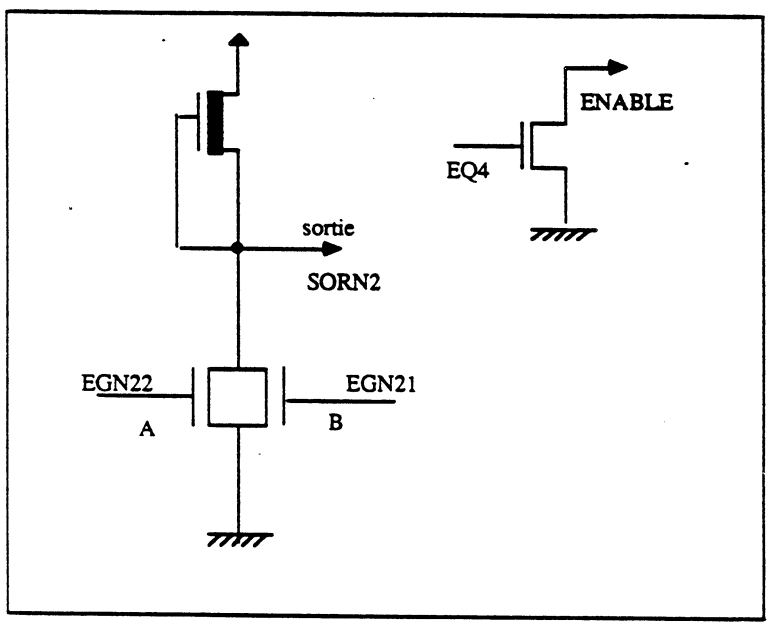


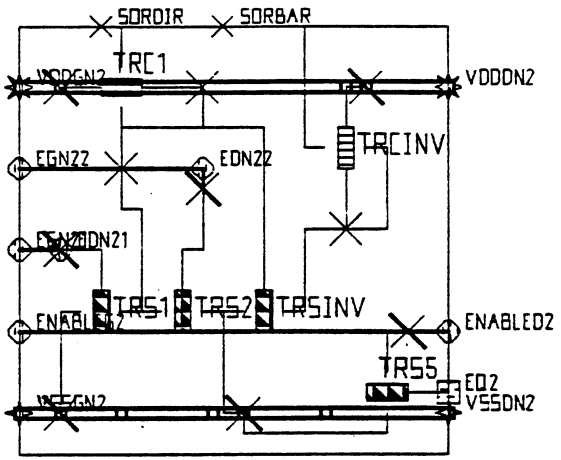
Cellule : Porte non-ou (2 entrées)+ un transistor signal

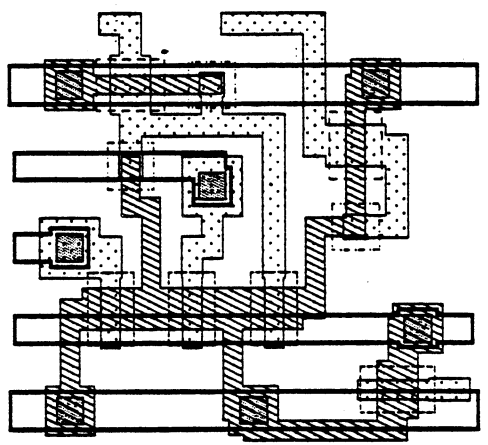
nom : NOR2

Schéma logique









Cellule : Porte non-ou (3 entrées)+ un transistor signal

nom : NOR3

Schéma logique

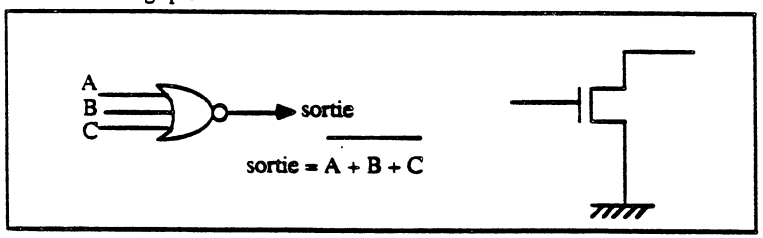
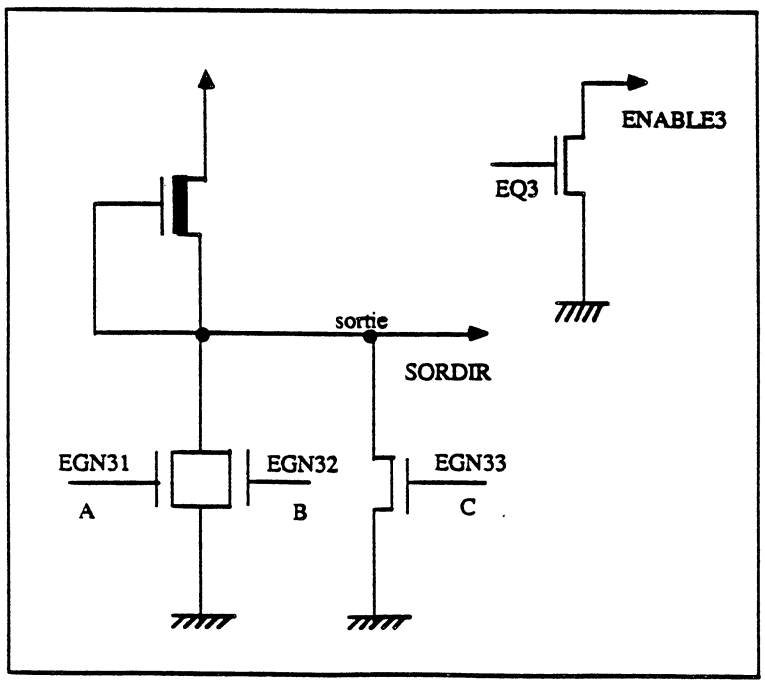
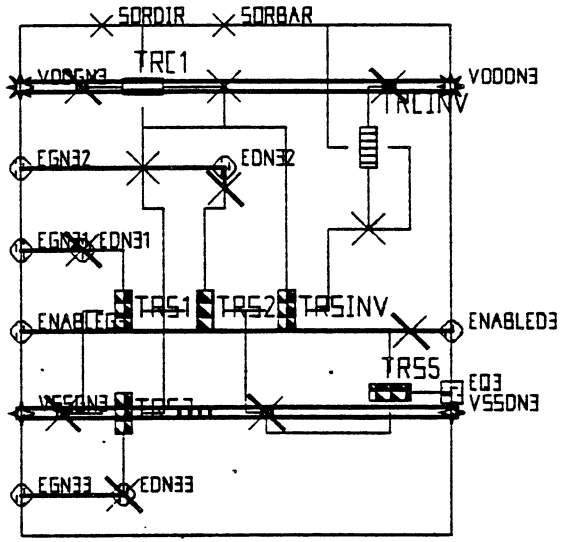
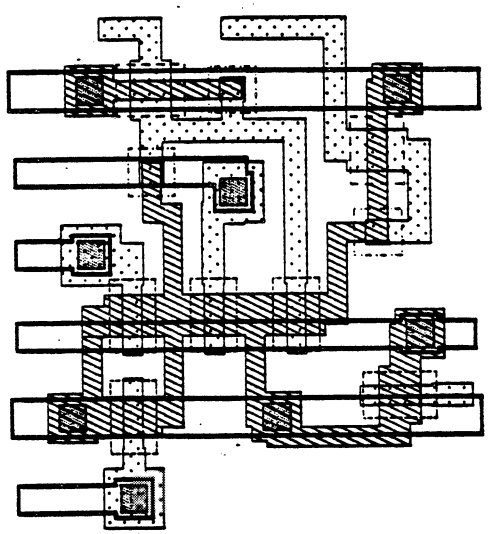


Schéma électrique



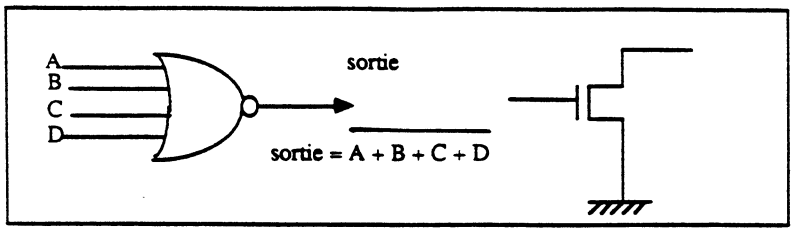


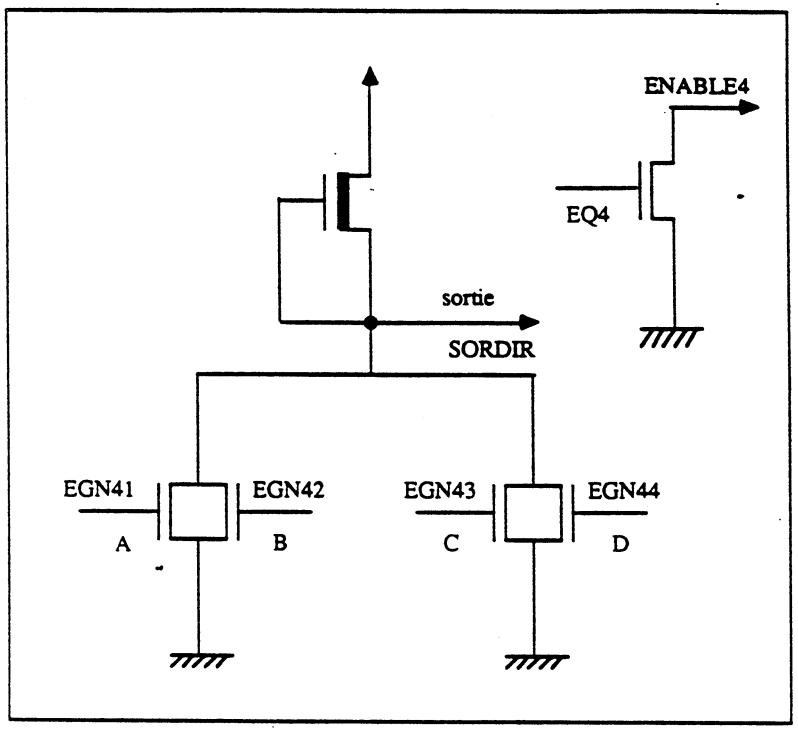


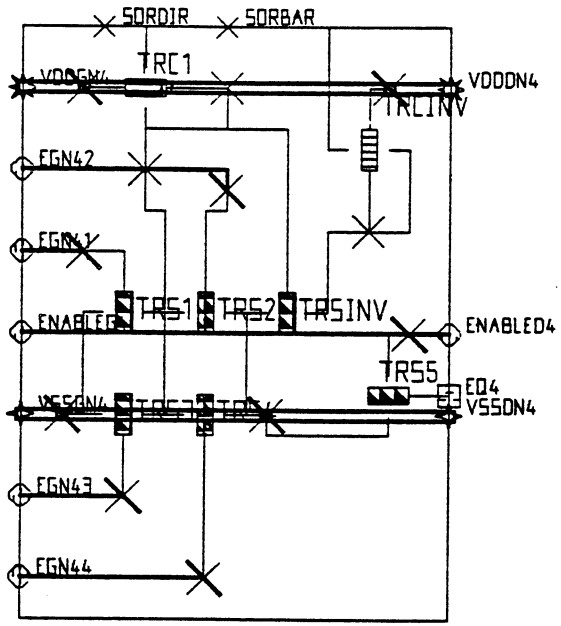
Cellule : Porte non-ou (4 entrées)+ un transistor signal

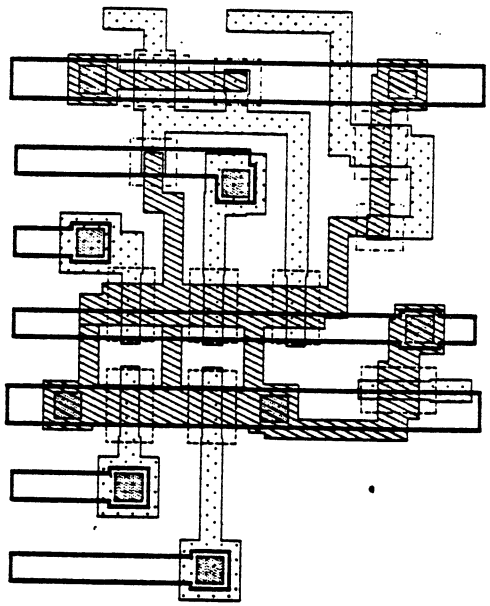
nom : NOR4

Schéma logique









Cellule : Selecteur (ou Multiplexeur 2 vers 1)

nom : SEL

Schéma logique

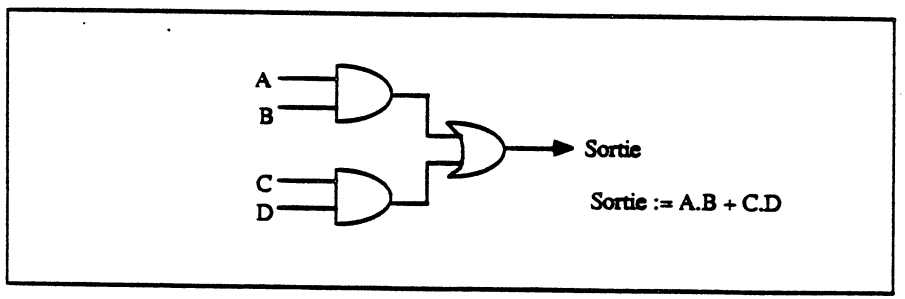
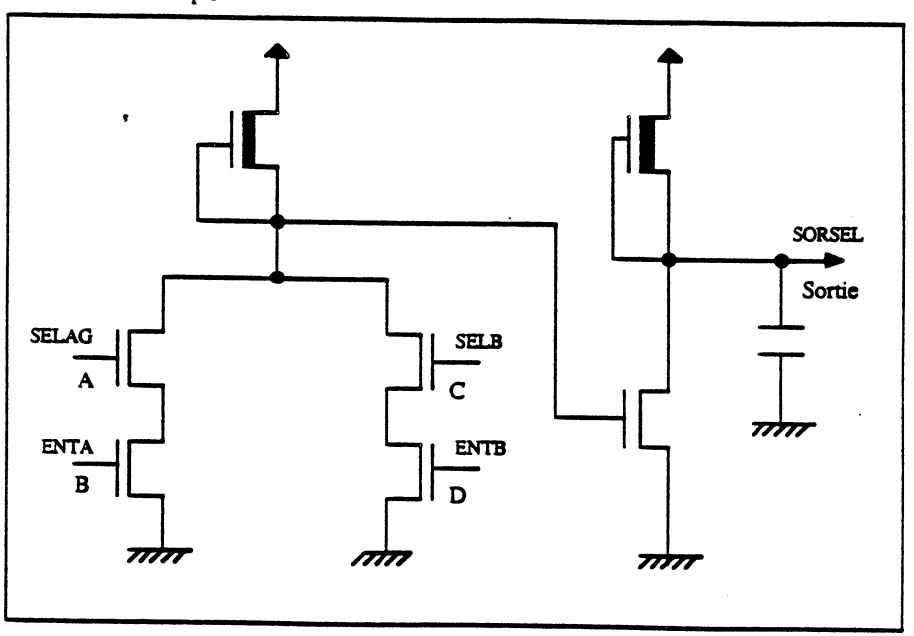
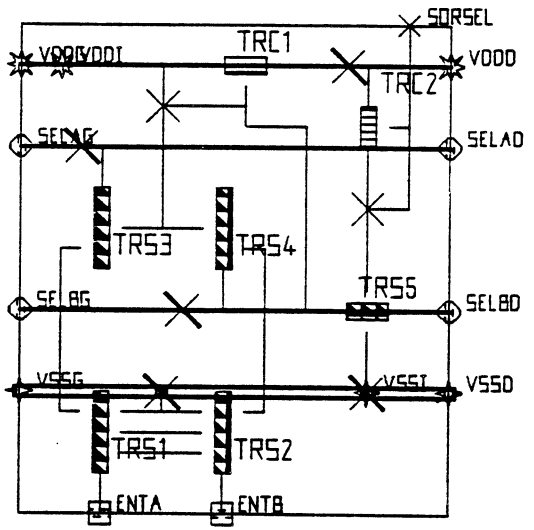
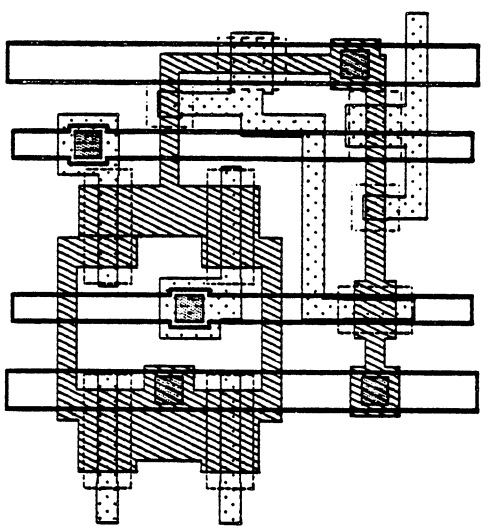


schéma électrique







Cellule : Porte ou-exclusif

nom : OREXC1

Schéma logique

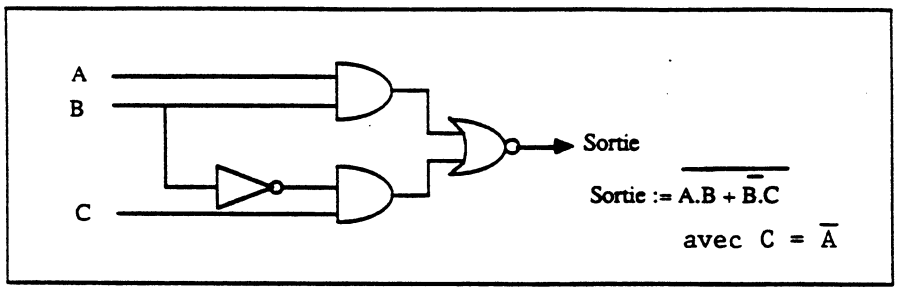
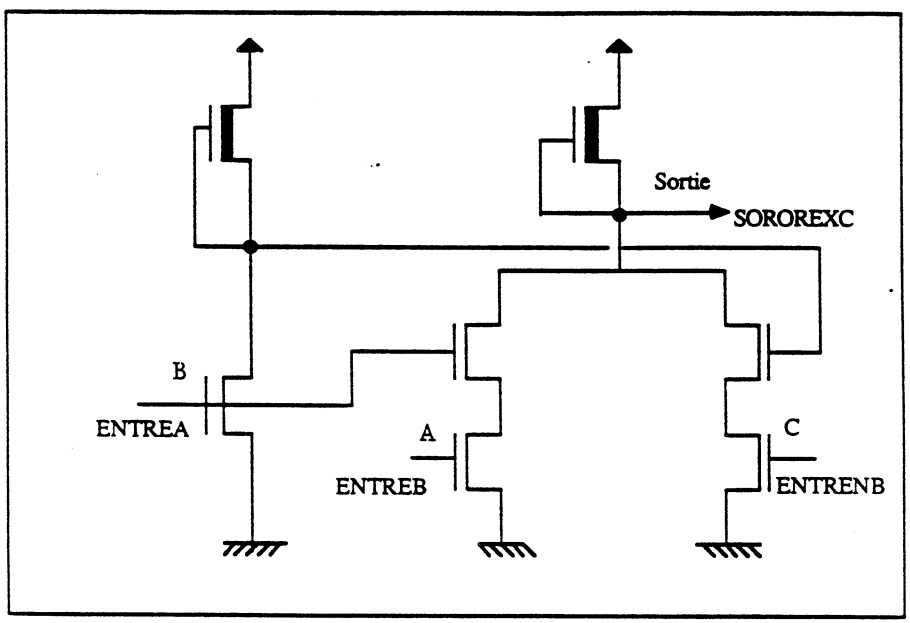
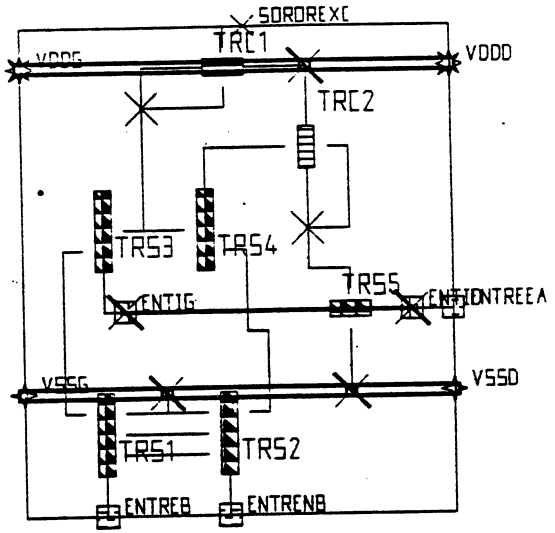
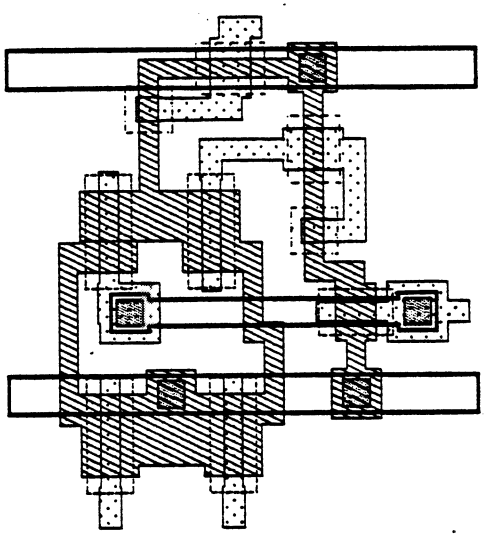


Schéma électrique







Cellule : Porte "égalité" (non-ou-exclusif)

nom : NOREXC

Schéma logique

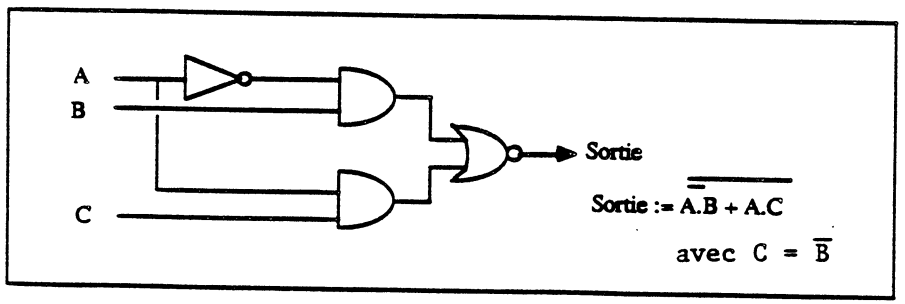
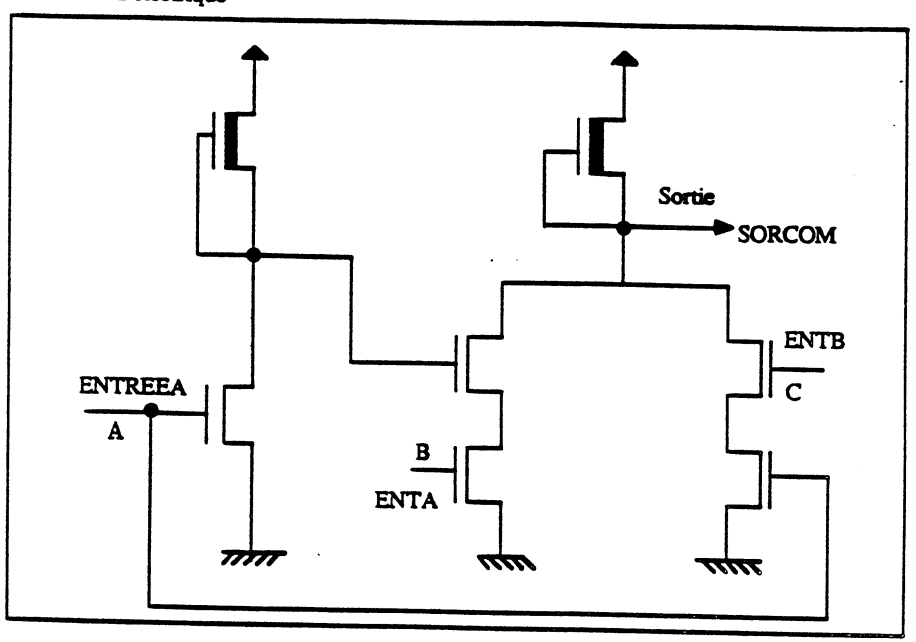
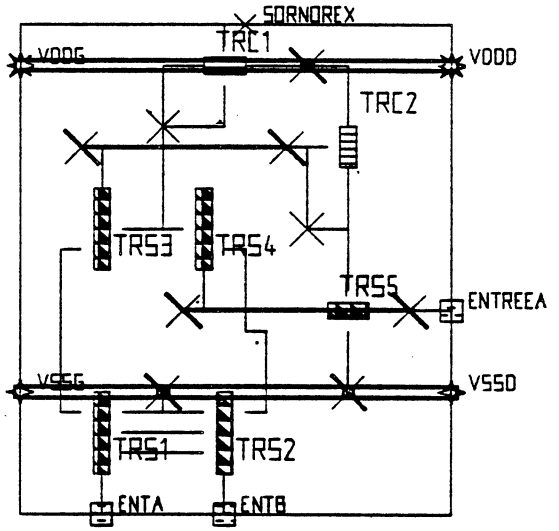
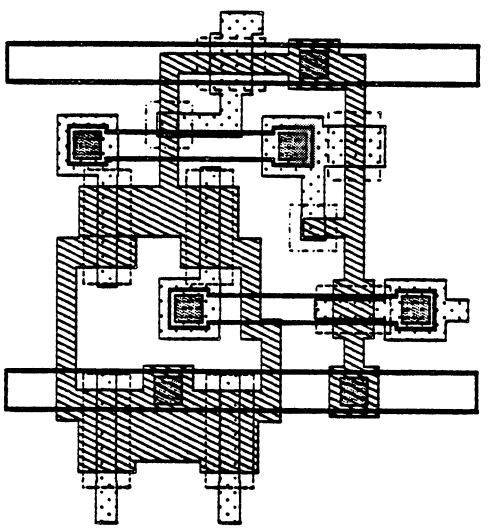


Schéma électrique







Cellule : Comparateur

nom : COMPAR

Schéma logique

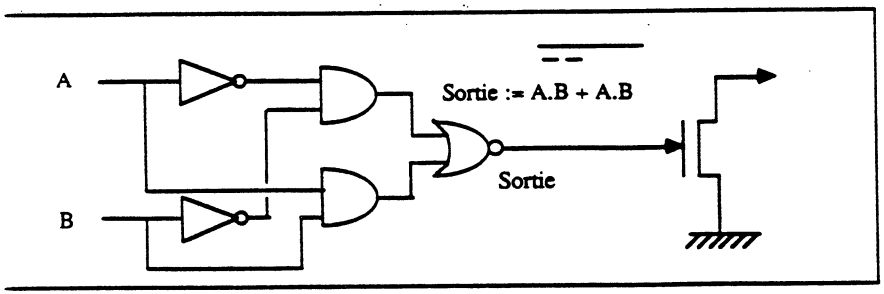
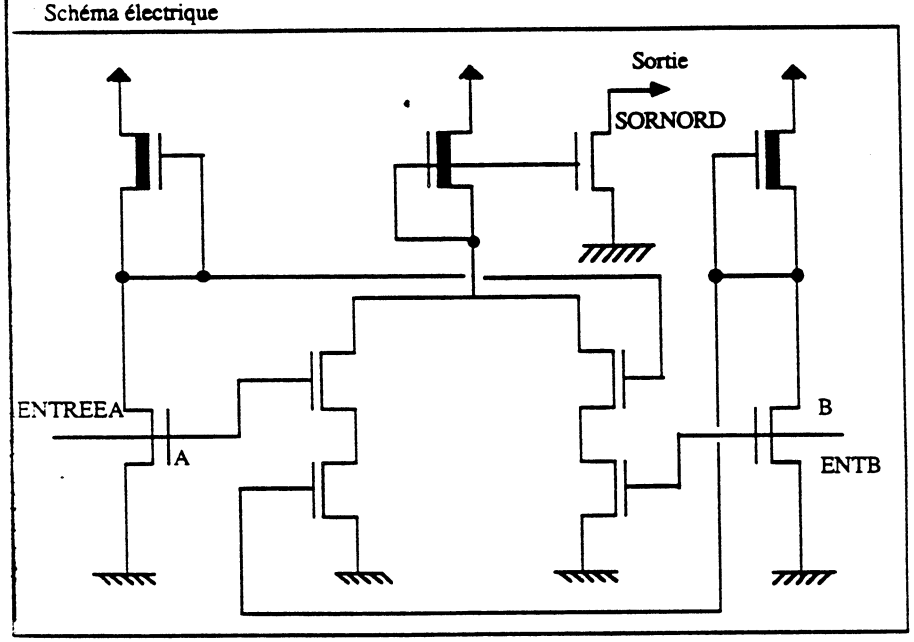
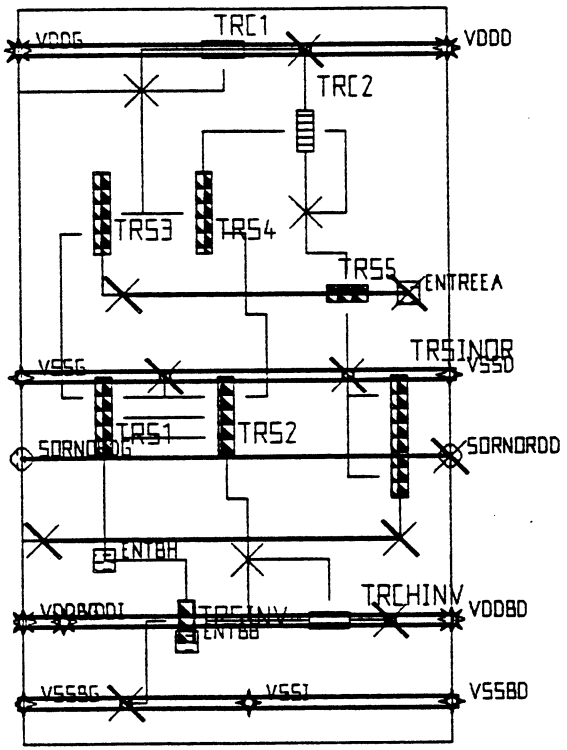
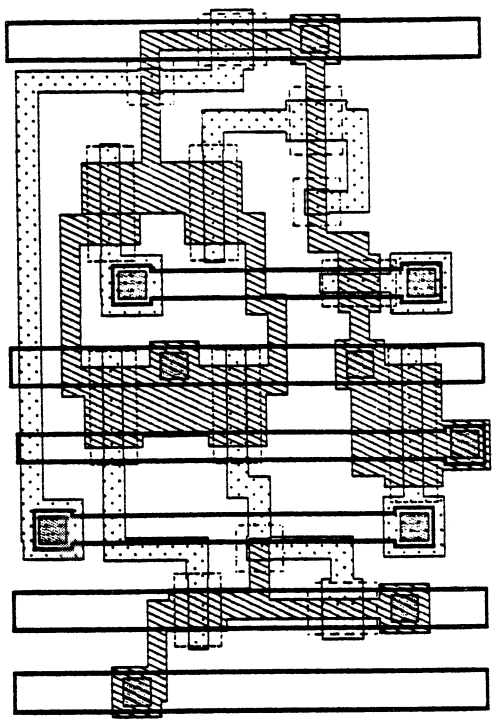


Schéma électrique



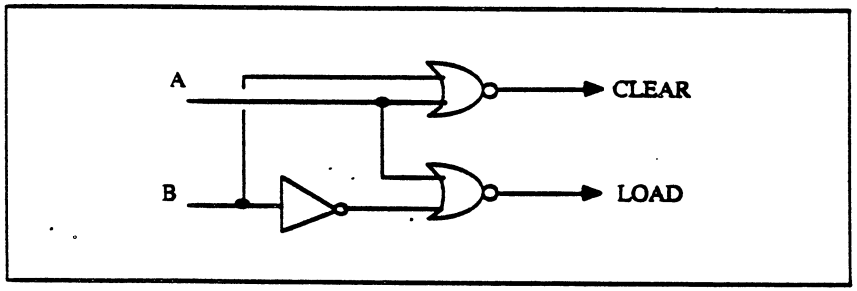




Cellule : Logique de contrôle

nom : CL/LOAD

Schéma logique



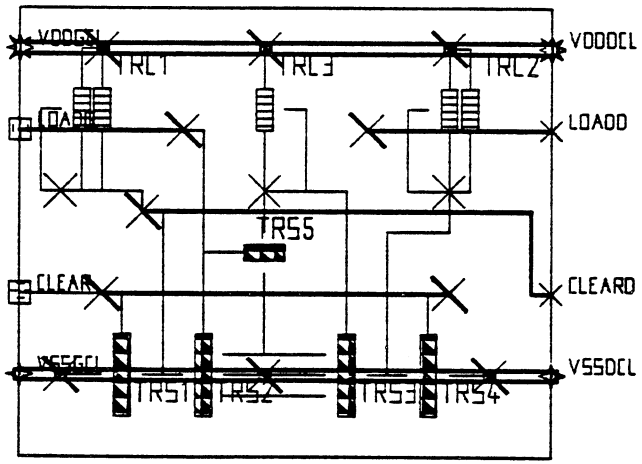
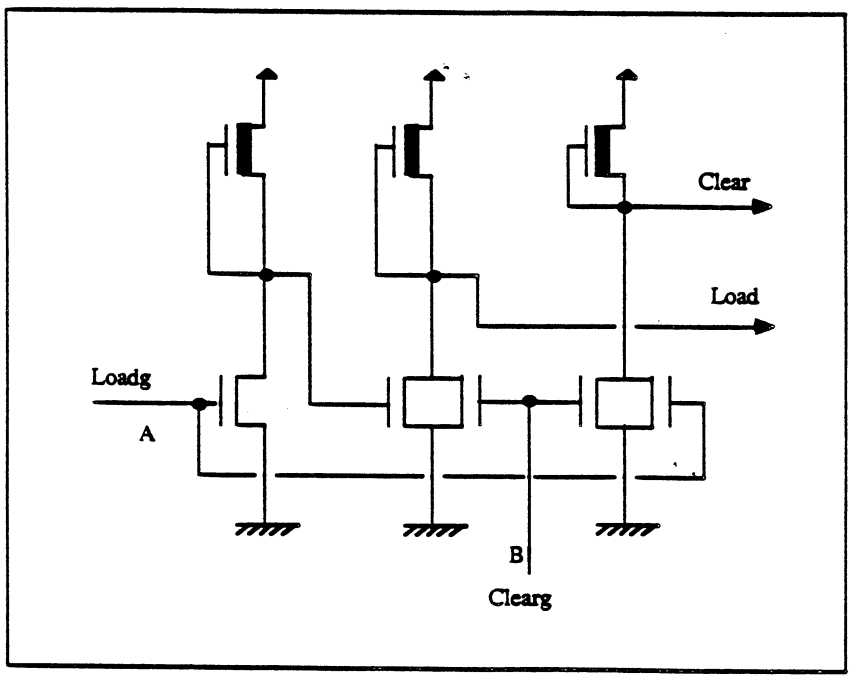
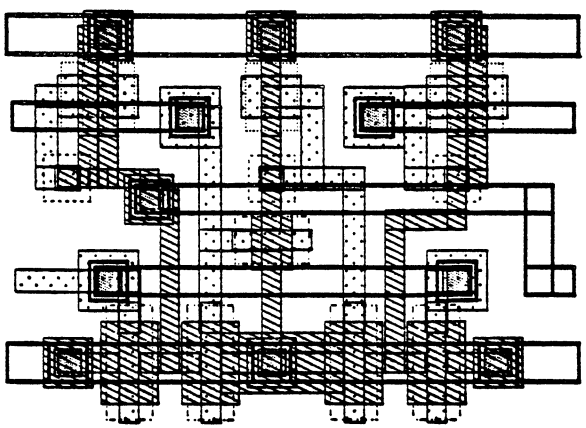


Schéma électrique





Cellule : Bascule Maître -esclave

nom : BASCULED

Schéma logique

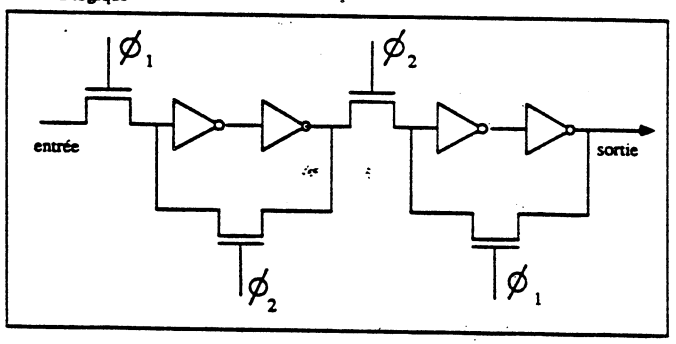
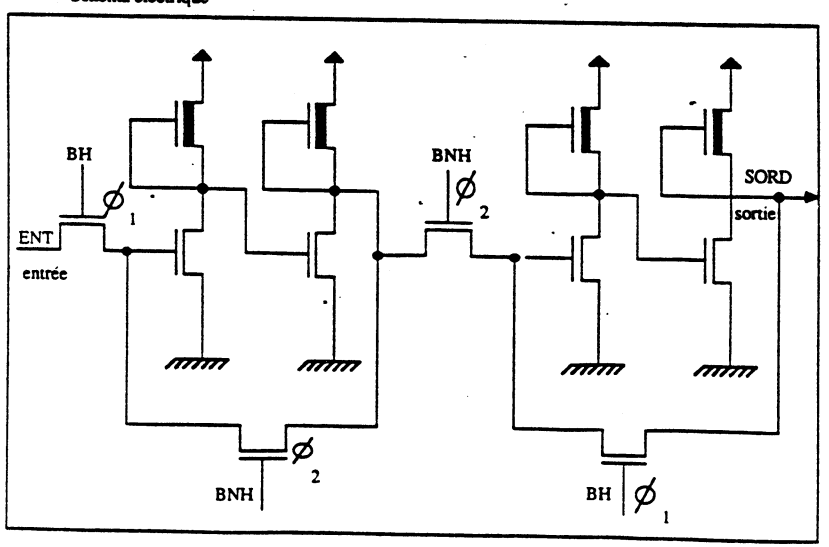
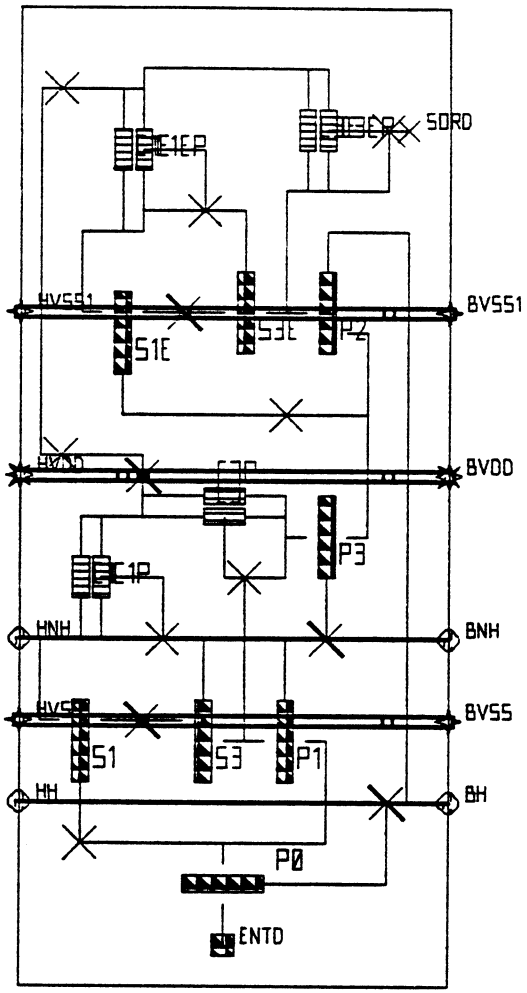
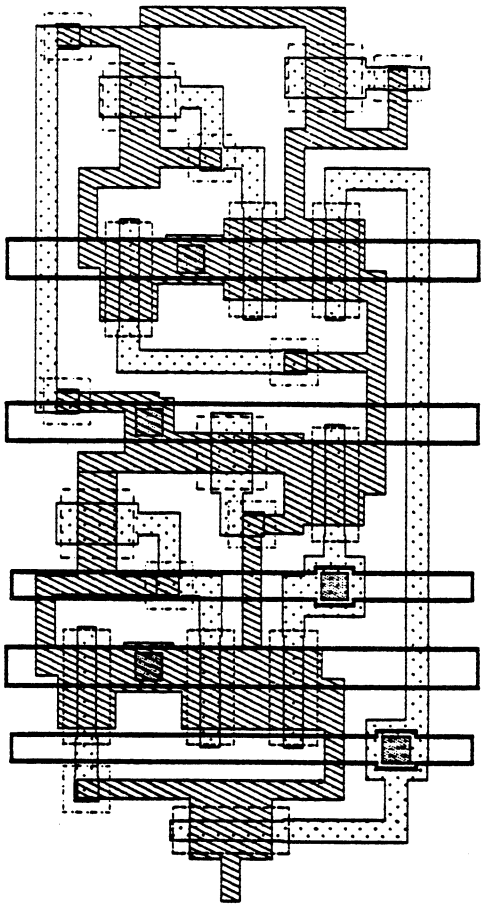


Schéma électrique







Cellule : Bascule Maître-esclave avec remise a 1 synchrone

nom : BAS

Schéma logique

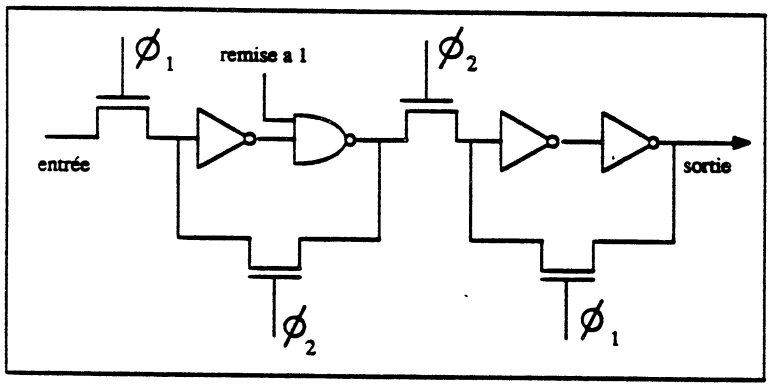
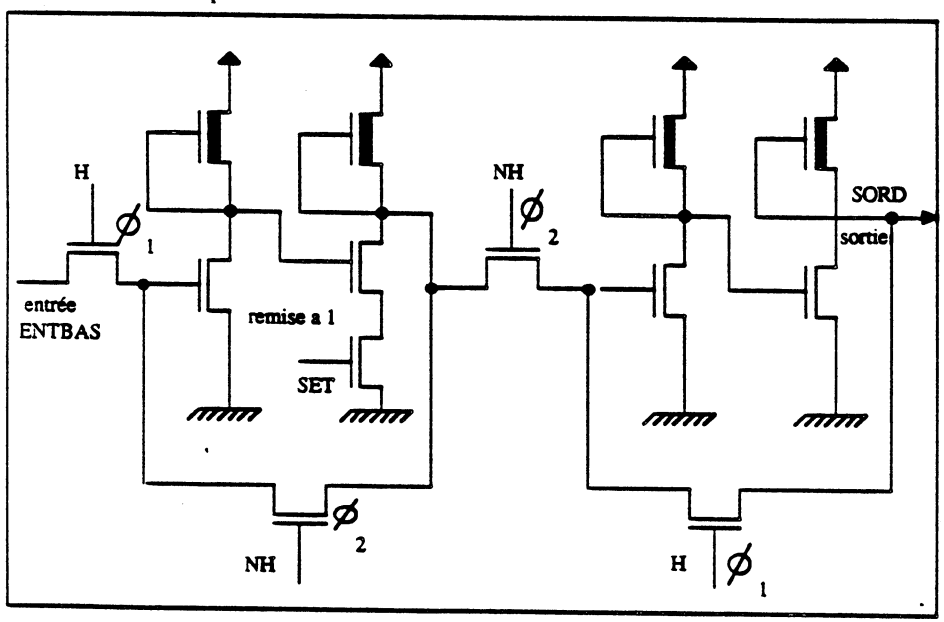
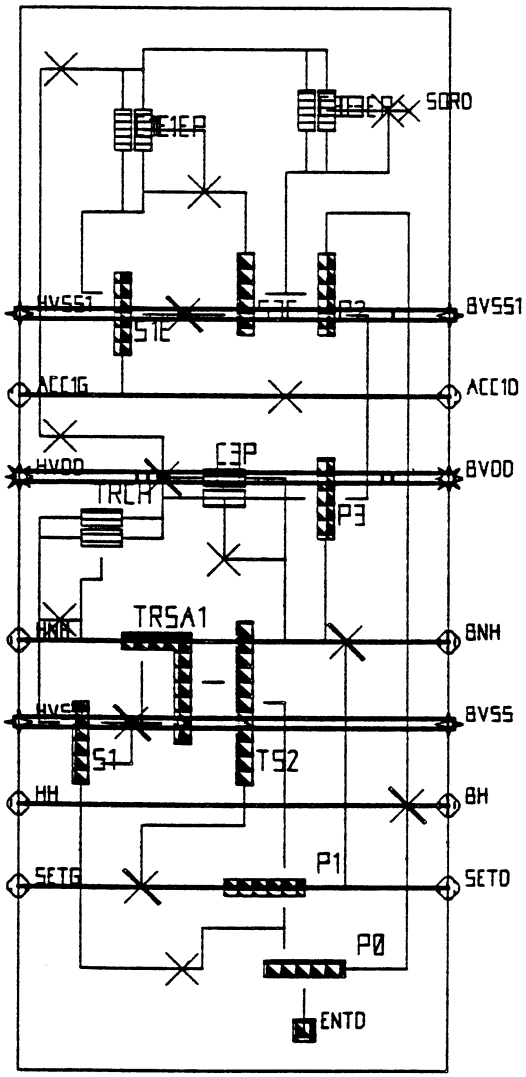
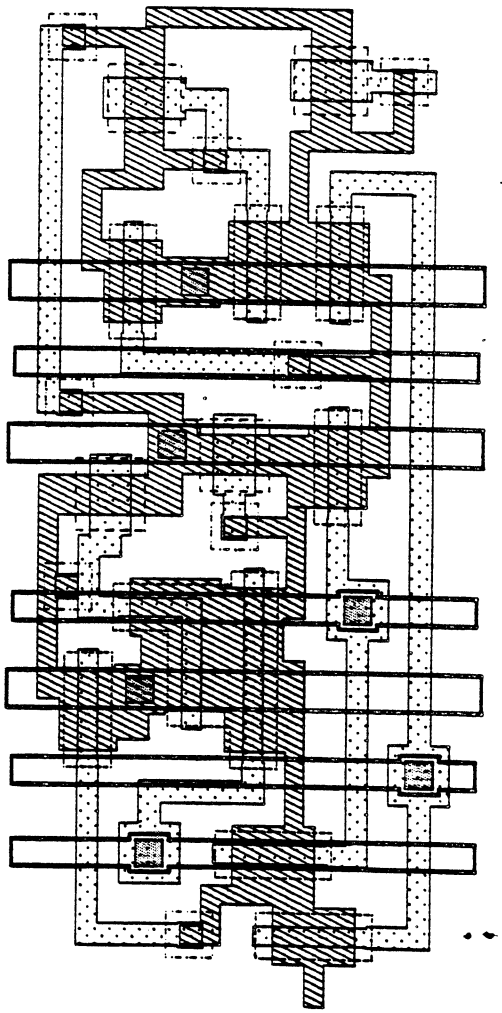


Schéma électrique







Cellule : Bascule Maître-esclave avec remise a 0 asynchrone

nom : BASAC

Schéma logique

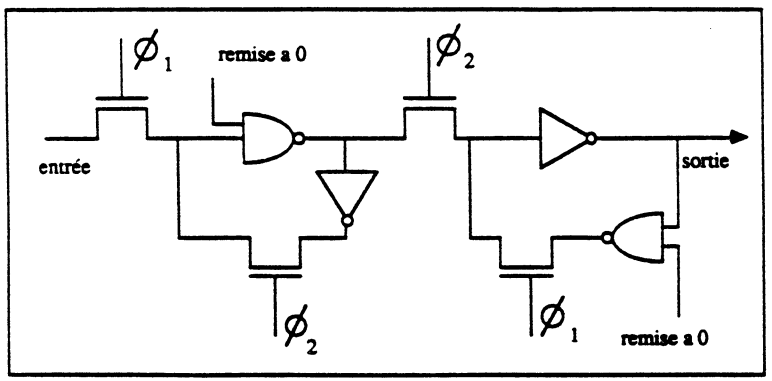
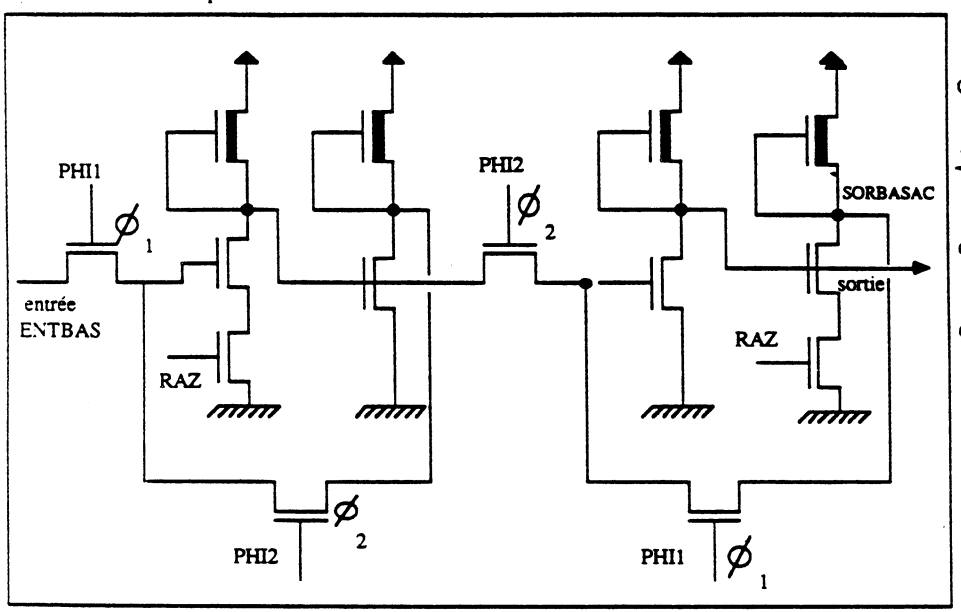
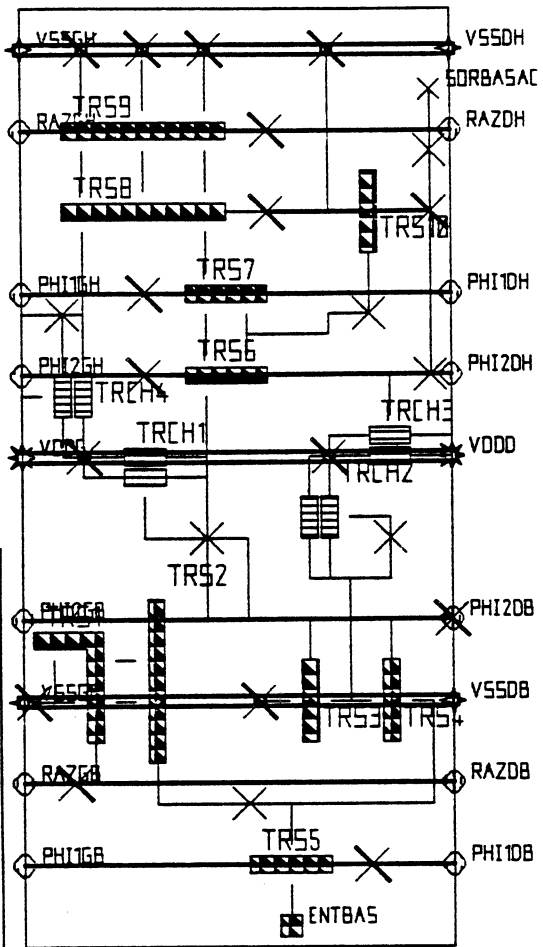
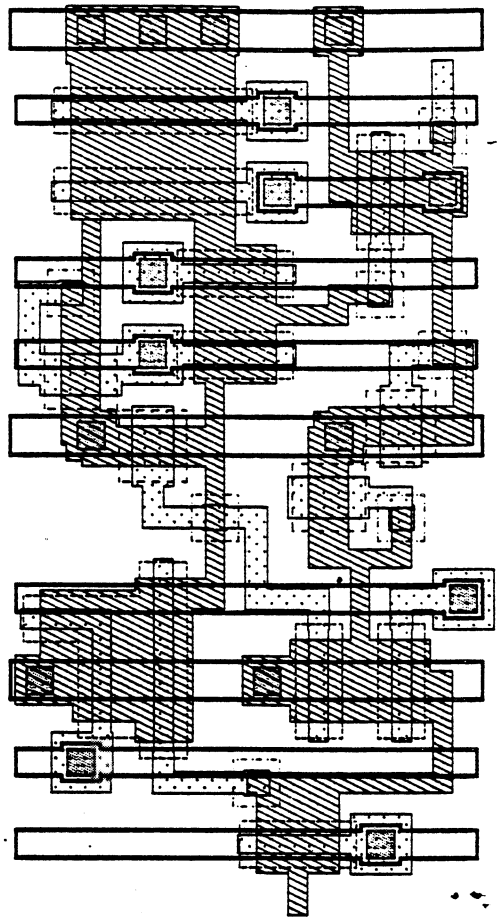


Schéma électrique







ANNEXE E

EXEMPLES D'OPERATEURS EN TRANCHES

EXEMPLES D'OPERATEURS EN TRANCHES

Les opérateurs présentés sont :

1.- Registre à décalage.

2.- Désérialisateur

3.- Désérialisateur avec remise à zéro synchrone

4.- Sérialisateur

5.- Registre Parallèle

6.- Compteur synchrone avec remise à zéro synchrone

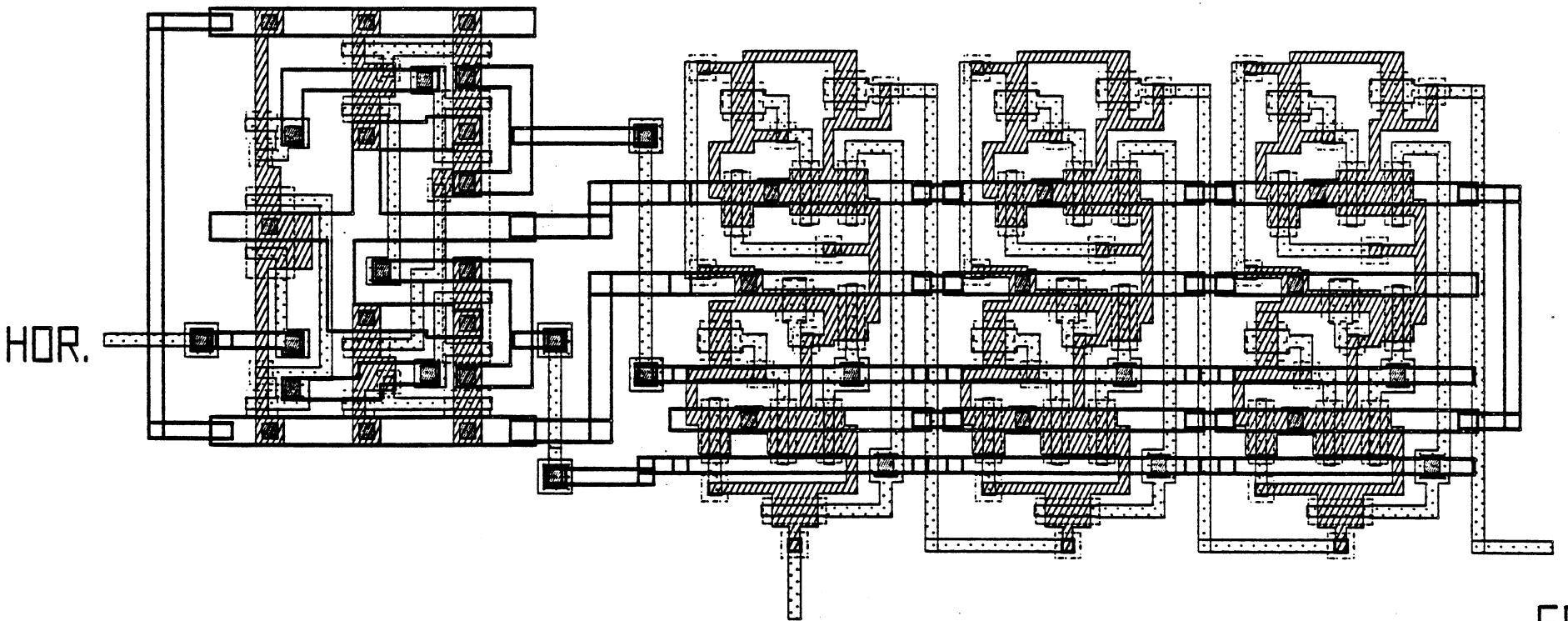
7.- Compteur synchrone avec affichage

8.- Décompteur synchrone avec affichage

9.- Générateur de CRC

REGISTRE A DECALAGE

3 bits

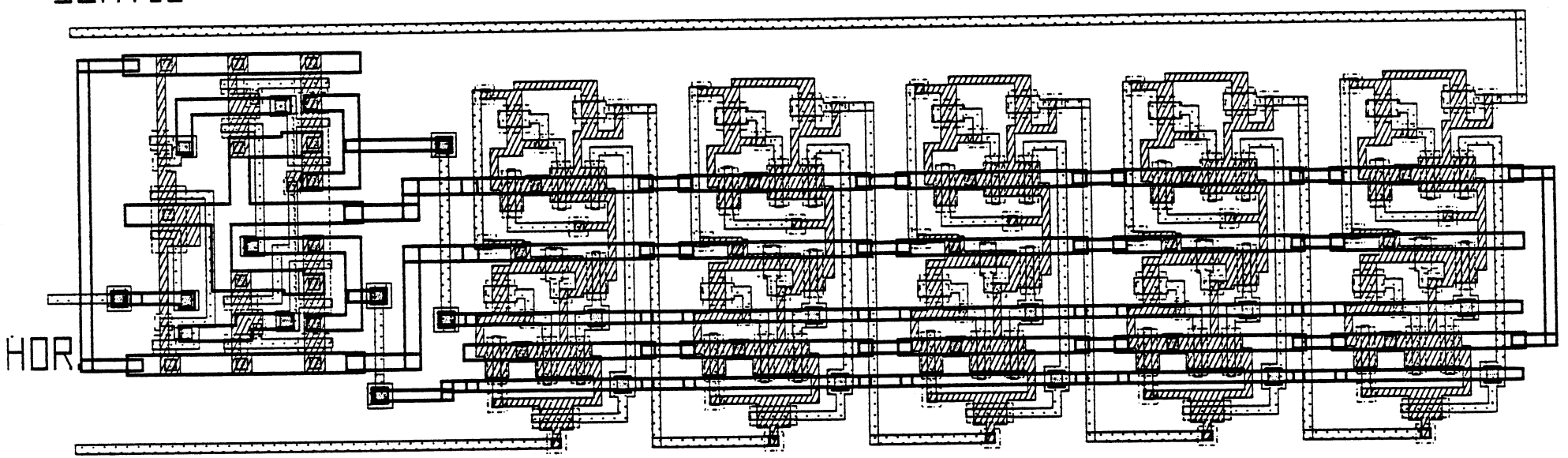


ecartement
entre bits 3

REGISTRE A DECALAGE

5 bits

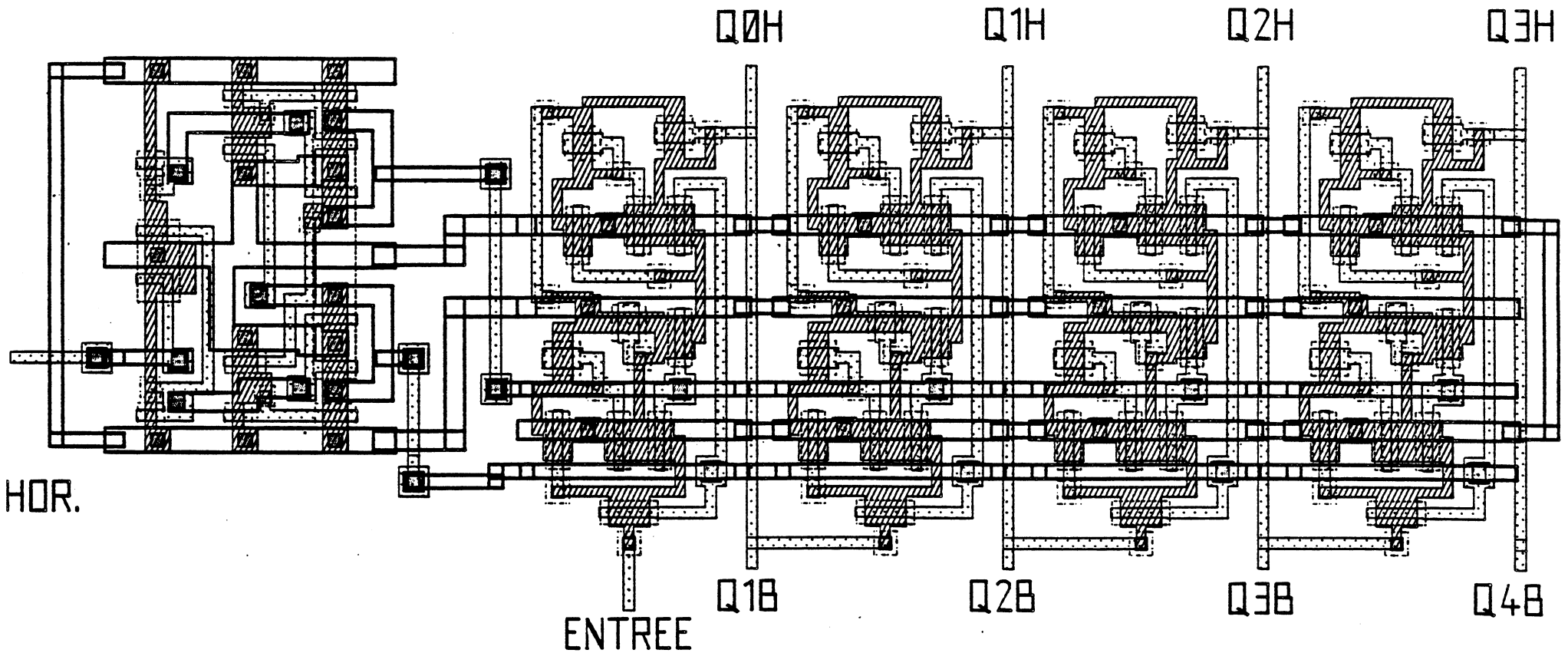
SORTIE



ENTREE

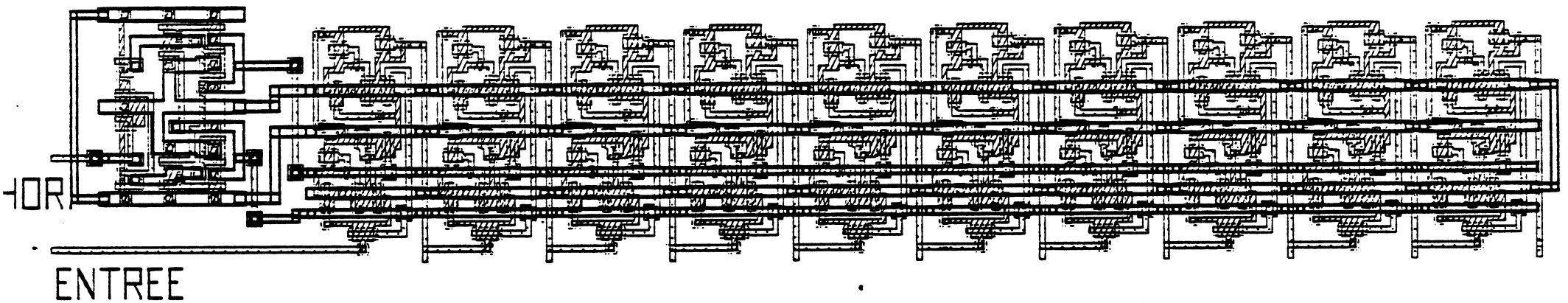
ecartement
entre bits 5

DESERIALISATEUR 4 bits



ecartement
entre bits 4

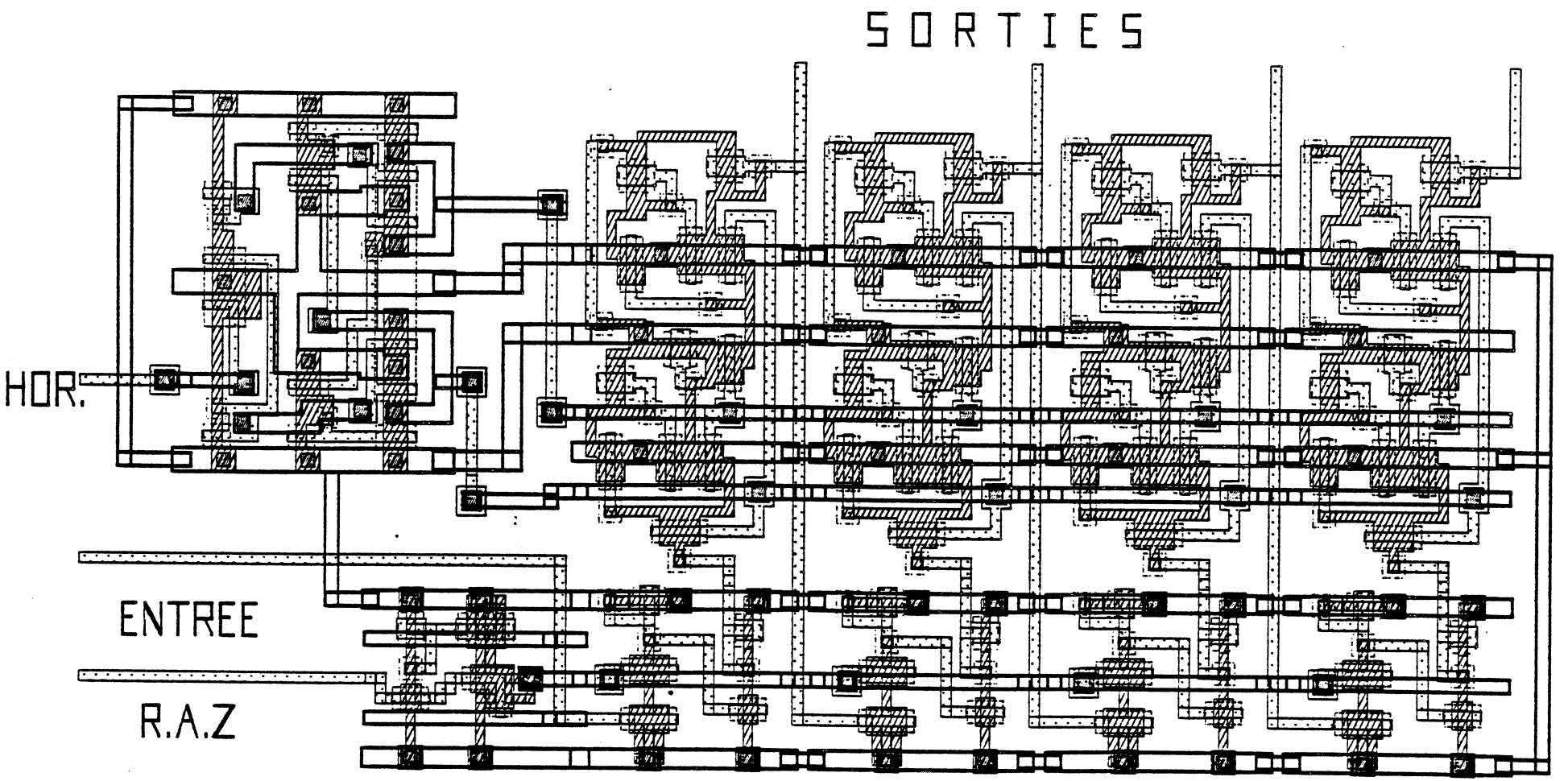
DESERIALISATEUR 10 bits



SORTIES

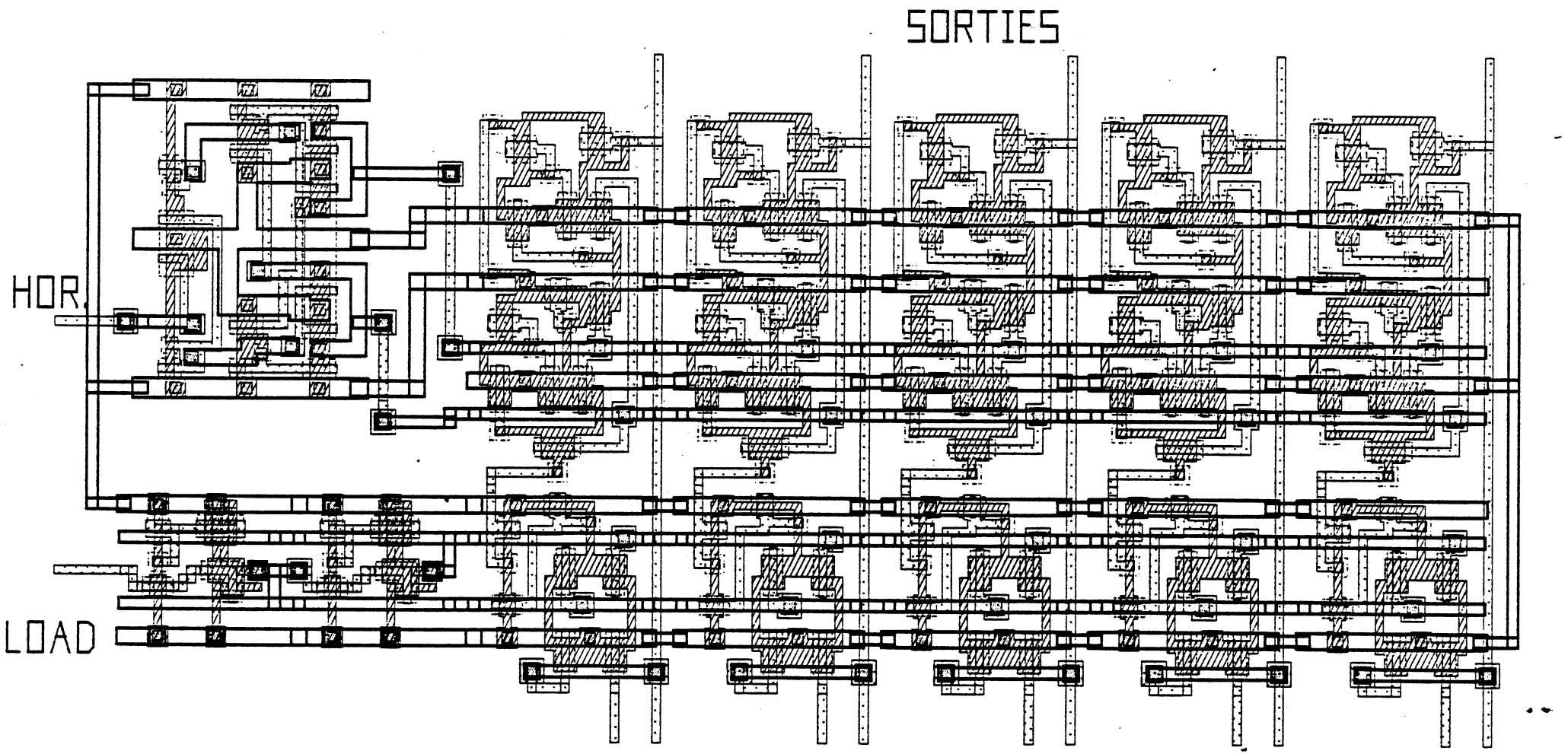
ecartement
entre bits 3

DESERIALISATEUR AVEC R.A.Z. SYNCHRONE

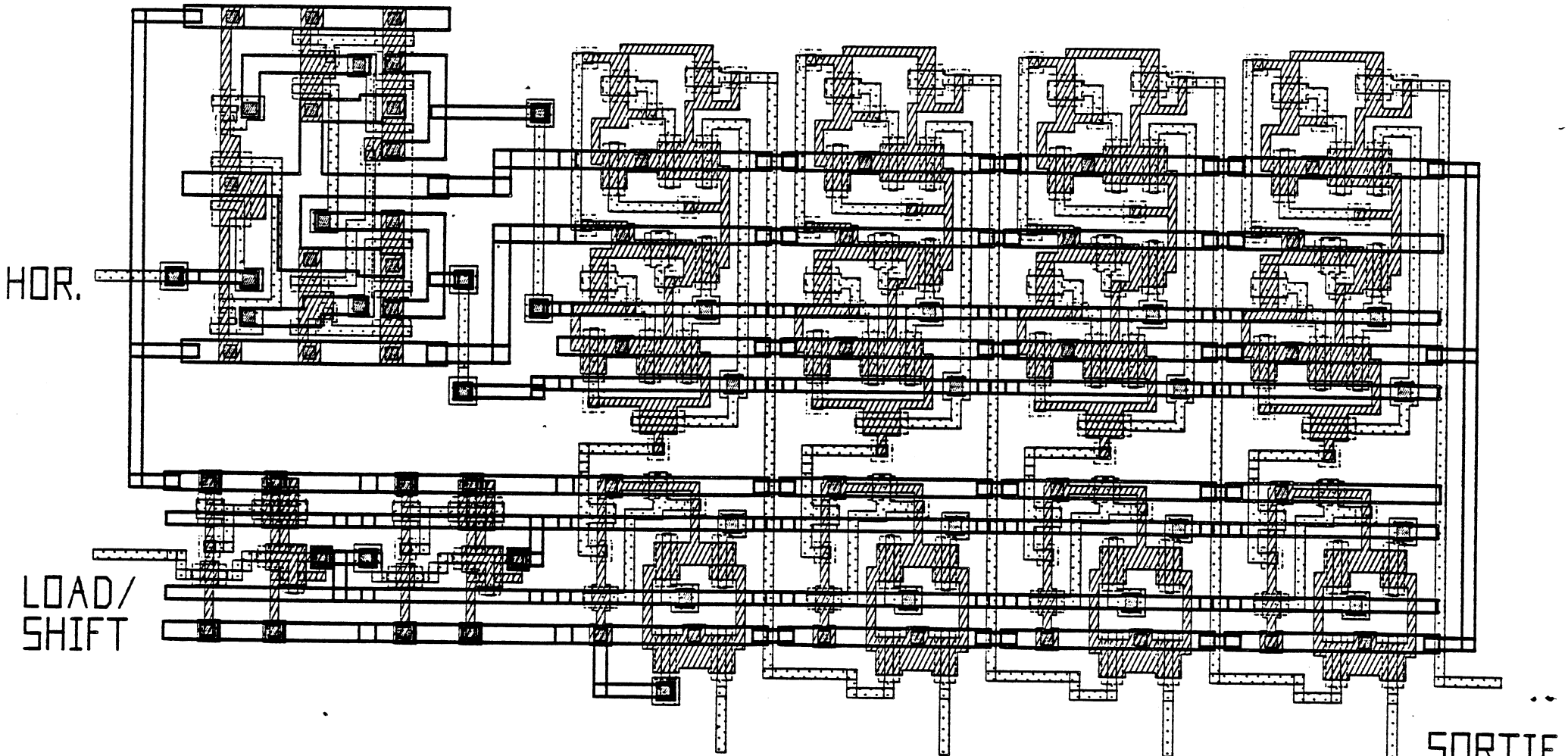


ecartement
entre bits 3

REGISTRE PARALLELE 5 bits

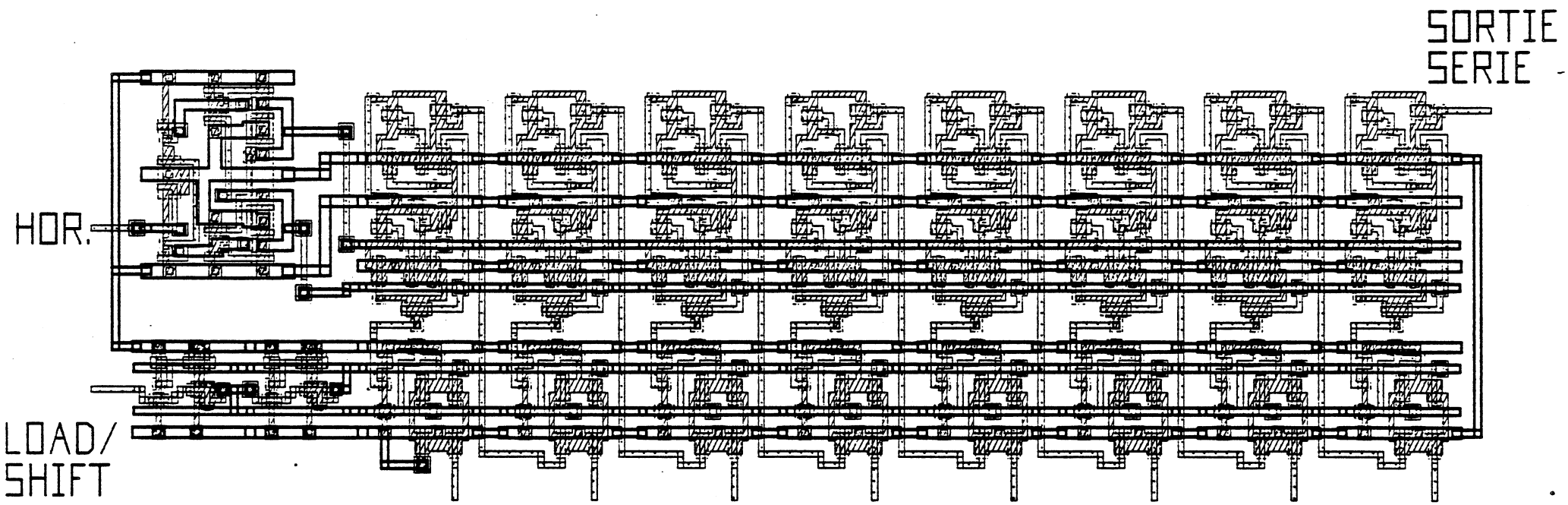


SERIALISATEUR 4 bits



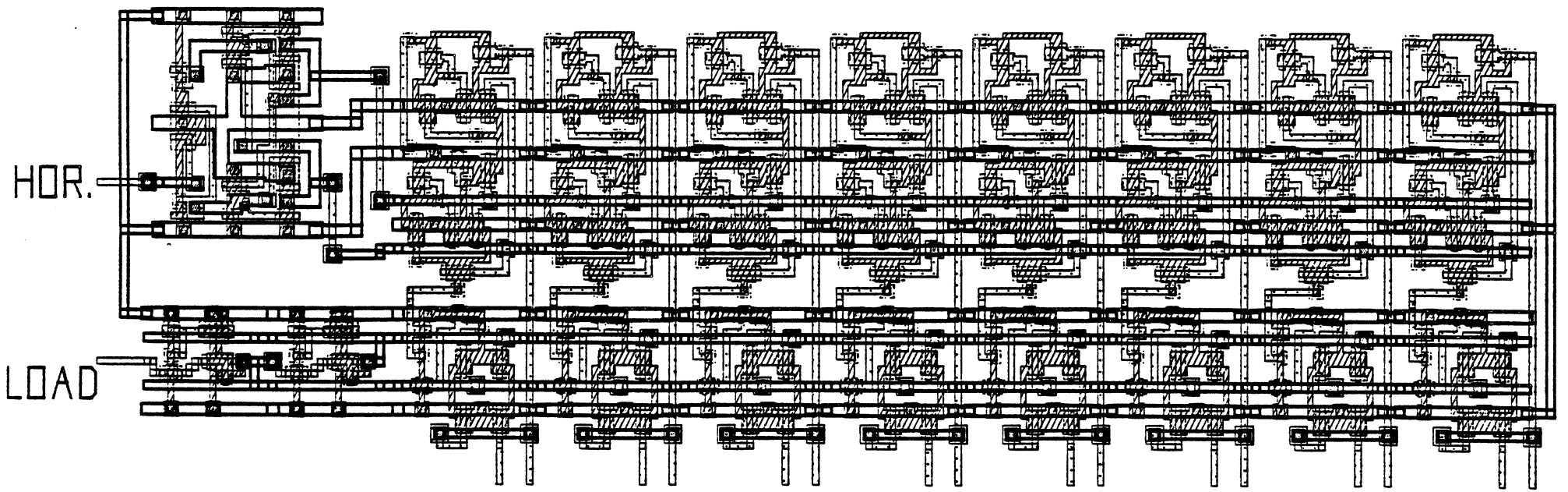
ecartement
entre bits 3

SERIALISATEUR 8 bits



ecartement
entre bits 4

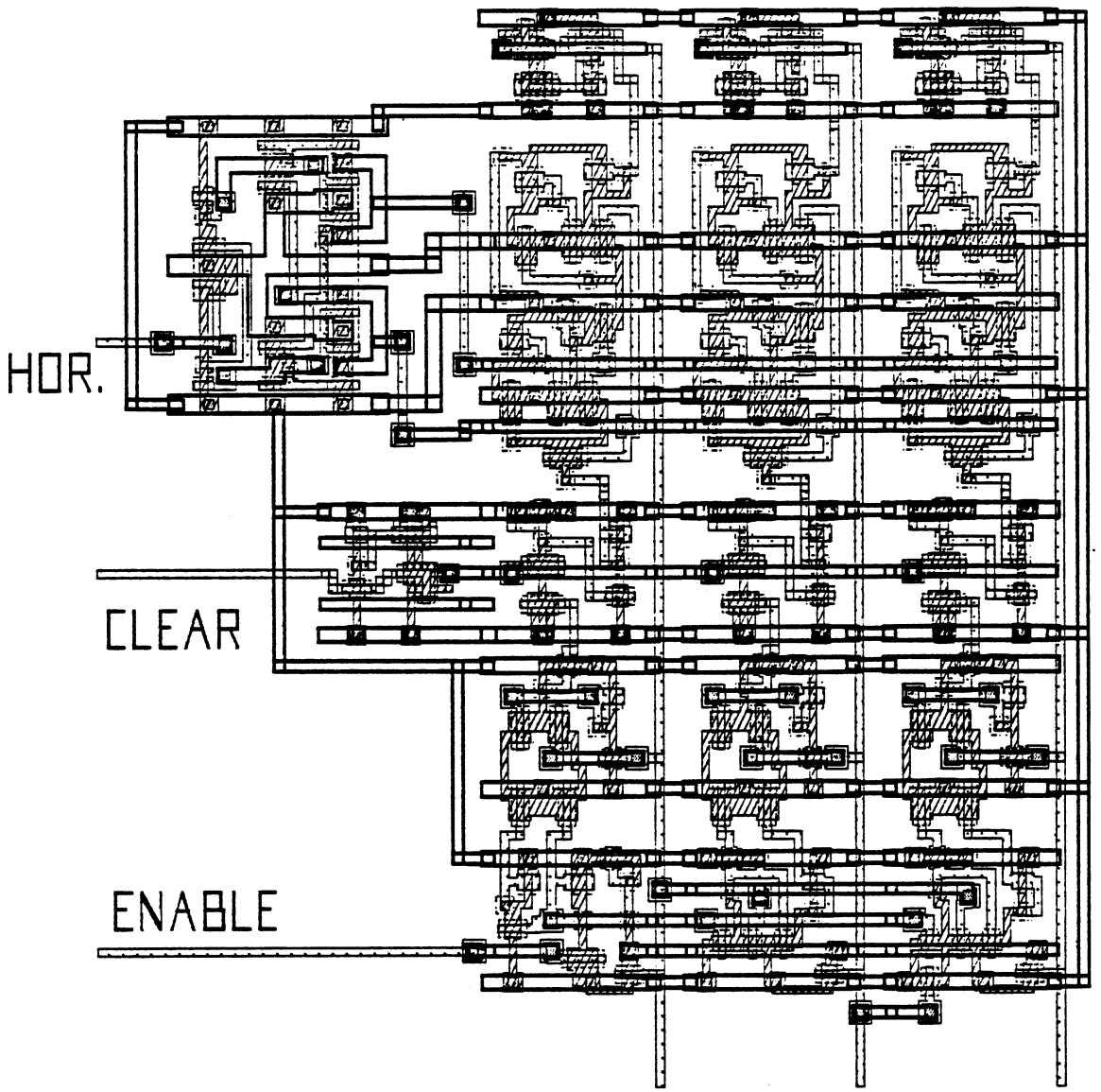
REGISTRE PARALLELE 8 bits



ENTREES ET SORTIES

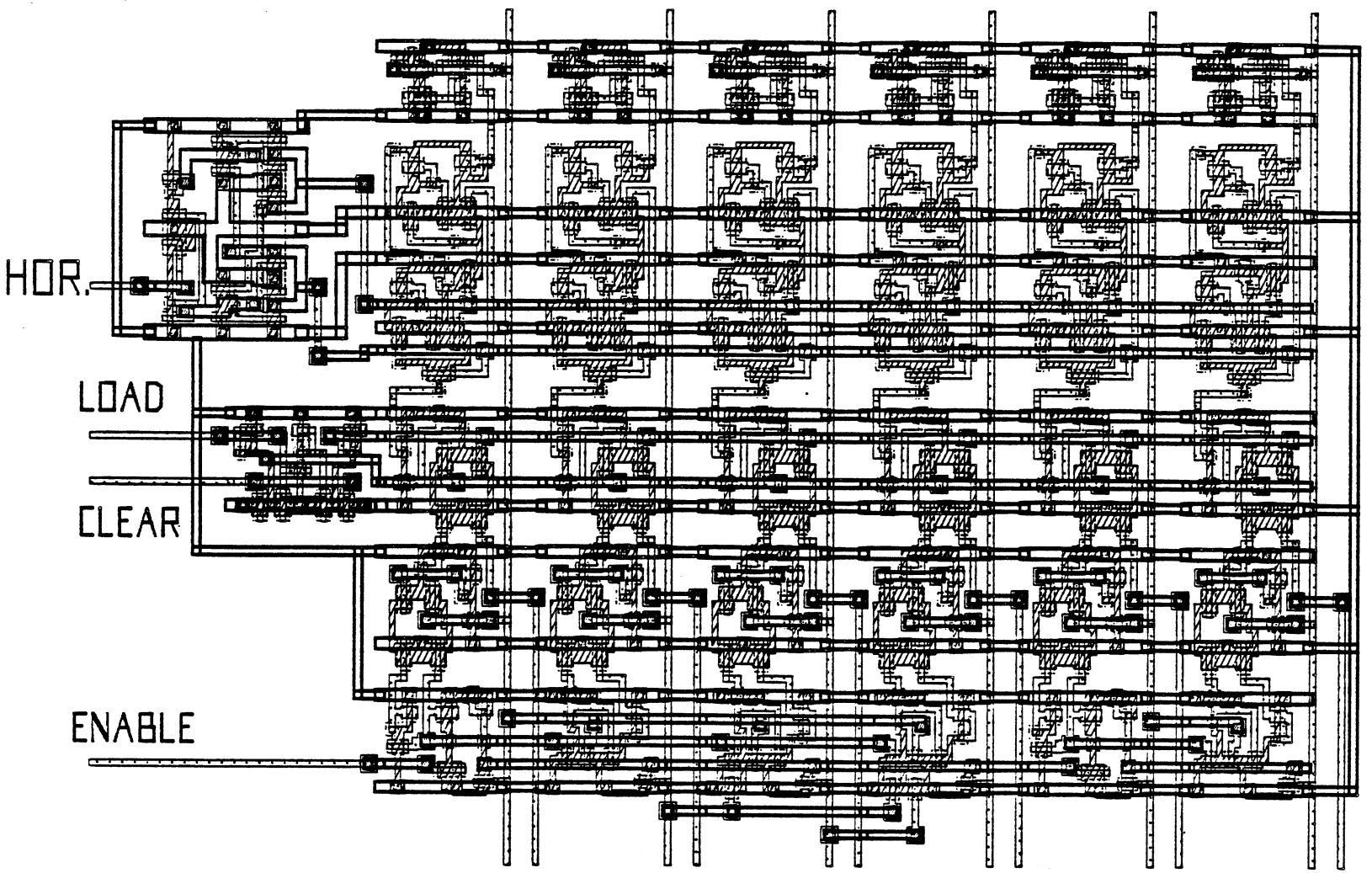
ecartement
entre bits 3

COMPTEUR SYNCHRONE
AVEC R.A.Z. SYNCHRONE



COMPTEUR SYNCHRONE
AVEC AFFICHAGE ET R.A.Z. SYNCHRONE 6 bits

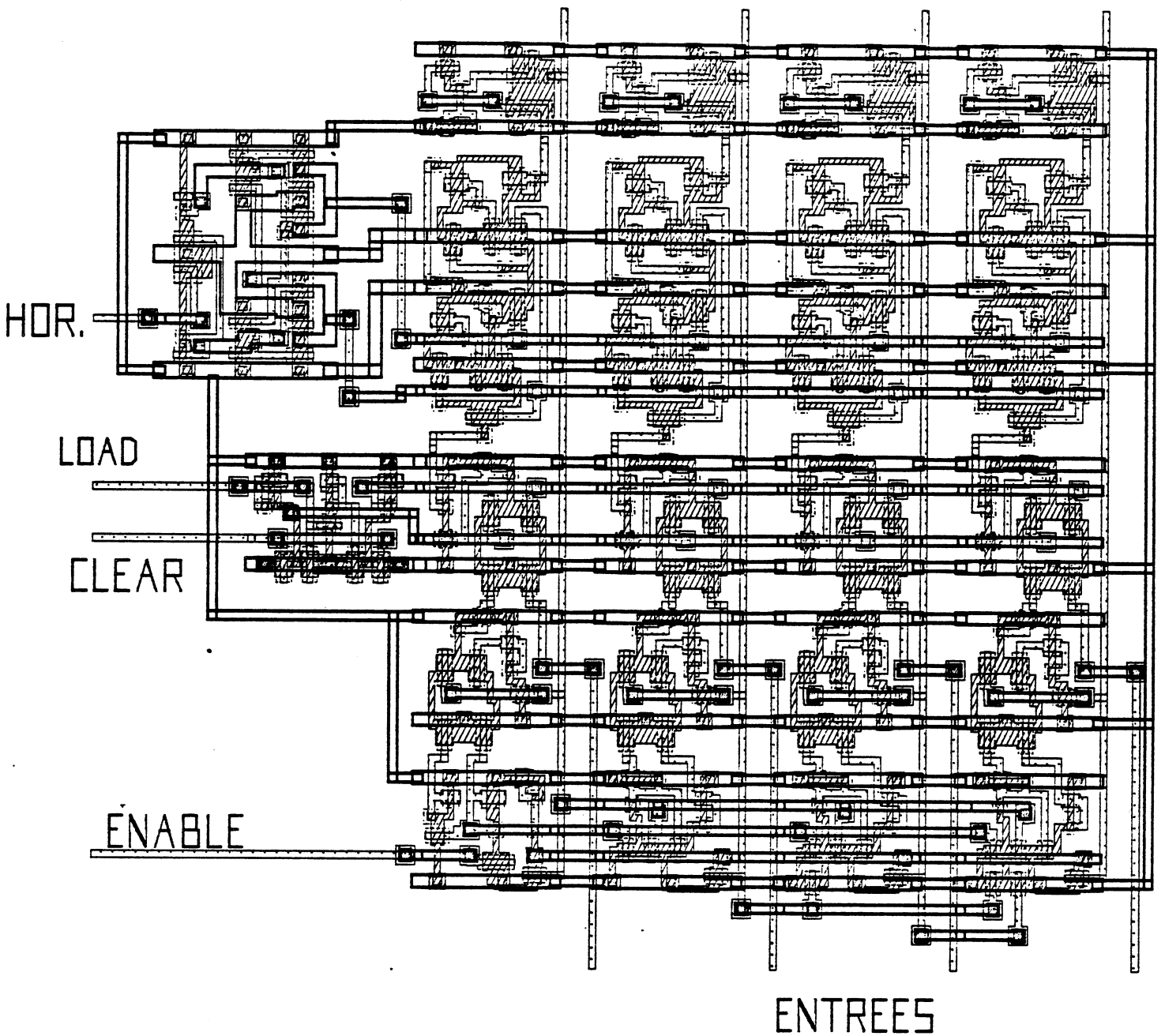
SORTIES



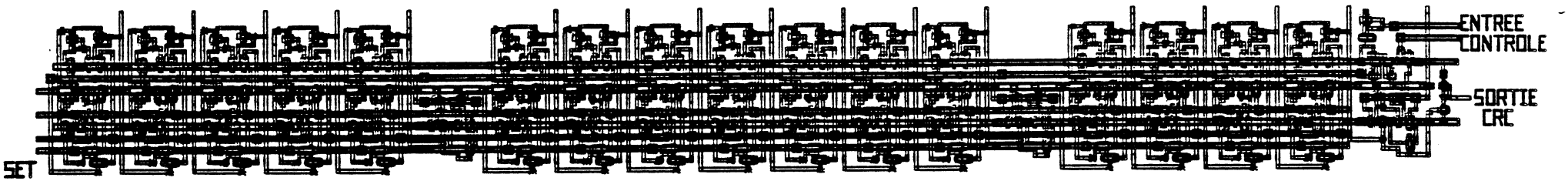
ENTREES ET SORTIES

ecartement
entre bits 5

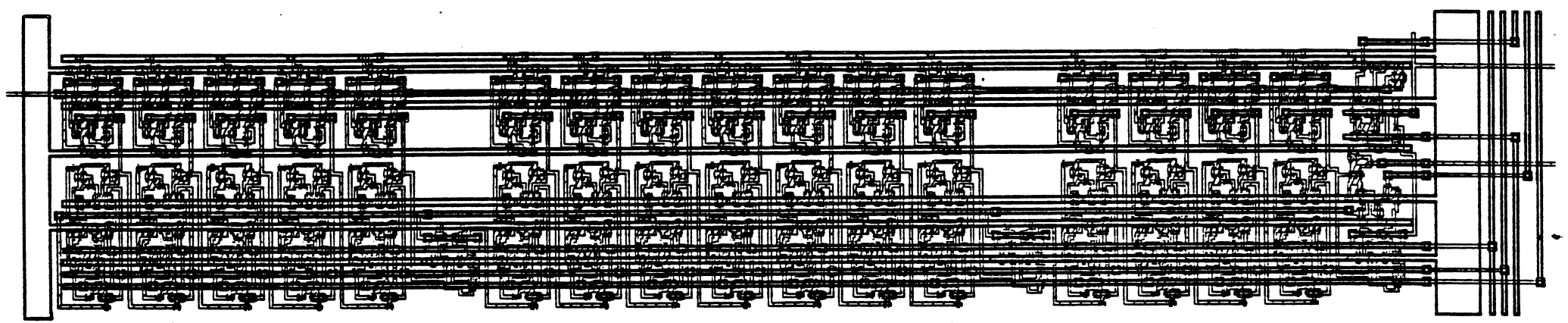
DECOMPTEUR 4 bits



GENERATEUR CRC (CODEUR)



GENERATEUR CRC (CODEUR/DECODEUR)



POLYNOME GENERATEUR $X^{16} + X^{12} + X^{5} + 1$

ANNEXE F

DEFINITION DES BROCHES

DU CIRCUIT FIP – VLSI

DEFINITION DES BROCHES DU CIRCUIT FIP-VLSI

La figure F.1 montre l'organisation externe du circuit, laquelle est formée de 39 broches classifiées en plusieurs groupes.

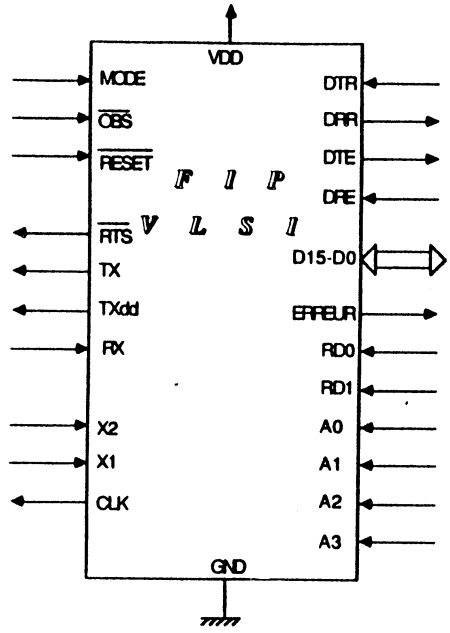


Figure F.1 Organisation externe du circuit communicateur FIP-VLSI

Les groupes de broches qui forment le circuit sont les suivants:

Côté station.

D0-D15	Bus de données bidirectionnel 16 bits,
DTR	Donnée prise, en réception,
DRR	Donnée prête, en réception,
DTE	Donnée prise, en émission,
DRE	Donnée prête, en émission,
ERREUR	Erreur dans la réception d'une trame,

Côté médium.

RTS	Request To Send. Au niveau "0", demande la validation du transmetteur; actif pendant la durée d'émission d'une trame
Tx	Transmission de données en série, niveau "1" en repos

Txdd	Tx retardée d'environ 100ns (pour faciliter l'"accentuation" avec certains types de transmetteurs)
Rx	Réception de données en série.

Signaux d'horloges.

X1, X2	Entrées de l'oscillateur. Un quartz à 20 Mhz donne une base de temps = 8 x le débit (2.5 Mbits/s), et
CLK	Horloge : 2; (10 MHz) pour usage externe.

Signaux de contrôle.

MODE	Au niveau haut cette broche détermine le mode appareil, au niveau bas, le mode distributeur
OBS	Au niveau bas cette broche détermine le mode observateur
A0-A3	Entrées pour définir l'adresse de la station,
RD0-RD1	Entrées pour définir les sens de transfert des fils qui constituent le bus de données
RESET	Remise à zéro du circuit.

Alimentation.

VDD	Valeur typique +5 volts
GND	la masse, 0 volts.

ANNEXE G

ALGORITHME DE CONTROLE

DU CIRCUIT FIP–VLSI

ALGORITHME DE CONTROLE DU CIRCUIT FIP

```
prog FIP(RESET,MODE,OBSERVATEUR,PORT,a,b,c,d,e,RD1,TRAME,DONNEES_REC,
        DONNEE_PRETE_E,DONNEE_PRET_R);

{ Les paramètres externes au circuit FIP sont:
        RESET: initialise le circuit lorsqu'il est à zéro,
         MODE: définit le mode de fonctionnement de la station, mode=1 =>
               station DISTRIBUTRICE, autrement => station APPAREIL ou
               secondaire,
   OBSERVATEUR: une station peut être employée pour surveiller le trafic sur
               le réseau et vérifier son bon fonctionnement, OBSERVATEUR=0,
          PORT: représente l'interface de données côté station, il est
               constitué de 16 voies,
DONNEE_PRETE_E: la station signale au circuit la présence d'une donnée prête
               sur le PORT,
DONNEE_PRETE_R: le circuit indique à la station la présence d'une donnée prête
               sur le PORT,
       a,b,c,d: sont des fils choisis, parmi les différents bits du port, pour
               définir l'adresse physique du circuit.Ces fils sont rebouclés
               sur les entrées A0,A1,A2,A3,
         e,RD1: sont deux fils qui permettent définir les sens de transfert des
               voies qui forment le bus de données avec la station. e connecté
               sur un des voies du bus et l'entrée RD0 du circuit définit les
               nombre de voies en entrée et celles en sortie. RD1 permet de
               définir le sens de transfert en entrée du circuit de toutes les
               voies qui forment le bus, sa valeur est "1" lorsqu'il est actif,
        TRAME: représente la trame codées et à émettre sur la ligné série,
   DONNEES_REC: c'est les données recues de la ligne série. }

const
       ID_CRC= $1C0F;      { reste unique pour l'identification du CRC en
                             réception, exprimé en hexadécimale }
type
       bit= 0..1;
        TA= tableau [0..3] de bit;
      TR11= tableau [0..10] de bit;
      TR15= tableau [0..14] de bit;
      TR16= tableau [0..15] de bit;
   adresse= struct
                cas  REG: booleen de
                     faux: A0,A1,A2,A3: TA
                     vrai: RA: TR16
                fincas;
            fin;

  adresse1= struct
                cas  nom : booleen de
                     faux: A.15: TR15
                     vrai: A.16: TR16
                fincas;
            fin;

    status= struct
                cas  état : booleen de
                     faux : ERREUR: string
                     vrai : MOTE: TR11
                fincas;
            fin;
var
     DECOMPTEUR_M16: entier;   { décompteur }
   MODE,OBSERVATEUR: bit;
   DONNEES_DECODEES: bit;
       CONTROLE_CRC: bit;       { validation du générateur de CRC }
           SONNERIE: booleen;   { indicateur reveil }
              ERROR: booleen;   { indicateur d'erreur }
           REG_ETAT: status     { registre d'état de la transmission }
               PORT: TR16;
        ADR_STATION: adresse;
        REG_ADRESSE: adresse1;

{**********************************************}
{************ PROGRAMME PRINCIPAL *************}
{**********************************************}

 { Il présente les principales procédures ou fonctions que le circuit
FIP-VLSI réalise en prenant en compte son mode de fonctionnement programmé. Les
différentes procédures sont:
  INITIALISATION: décrit l'initialisation du circuit,
 EMISSION_TRAMES: émission d'une trame de nom ou de données selon le mode
                  de fonction du circuit,
 RECEPTION_TRAME: permet de décrire la réception des trames de NOM et de
                  données,
          REVEIL: décrit la fonction de chien de garde utilisée par le
                  distributeur pour vérifier la transmission d'une trame de
                  données derrière une trame de nom. }

début;
INITIALISATION (RESET,a,b,c,d,e,f);
tantque  RESET <> 0 faire
   si MODE= 1  {il s'agit d'une station DISTRIBUTRICE de nom }
      alors EMISSION_TRAMES(MODE,ETAT_EMISSION_TRAM  ;
            REVEIL(SONNERIE);
            si non SONNERIE alors RECEPTION_TRAMES; fsi;
   fsi;
   { si MODE= 0 il s'agit d'une station secondaire }
   tantque MODE = 0 ou OBSERVATEUR= 0 faire
           RECEPTION_TRAMES  { réception d'une trame de nom ou de données}
   ftq;
ftq;
f FIP;  { fin programme de description }
```

```
{*************************************************}
{***************  INITIALISATION  ****************}
{*************************************************}

 { Elle décrit essentiellement la génération de la séquence d'initialisation
utilisée pour charger l'adresse de la station, ainsi comme pour définir le sens
de transfert des voies qui forment le bus de données. }

proc INITIALISATION (RESET,a,b,c,d,e,f)
type
       bit= 0..1;
        TA= tableau [0..3] de bit;
      TMUX= tableau [0..15] de bit;
TPLA2_INIT= tableau [0..15,0..3] de bit;
  SENS_DIR= struct
                  cas sens: booleen de
                      faux: TRD0: TA;
                      vrai: TRD1: bit;
                  fin;
            fin;
var
      a,b,c,d,e,i,j: entier;
          RESET,RD1: bit;
            MUXPORT: TMUX;
          PLA2_INIT: TPLA2_INIT;  {PLA2_INIT génère la séquence pour
                                   l'initialisation de l'adresse }
                RD0: TA;
     REG_DIRECTION: SENS_DIR; {définit le sens de transfert des voies
                               qui forment les bus de données }
début;
tantque RESET faire
  ATTENTE_RESET       {lorsque  RESET= 0 le circuit sera initialisé}
ftq;
MUXPORT:= 1;          {positionnement de tous les bits du port en sortie}
REG_ETAT.MOTE:= 0;    {initialisation du registre d'état}
{génération de la séquence pour le chargement de l'adresse physique du circuité}
pour i:= 0 à 3 faire
 pour j:= 0 à 15 faire
   PORT[j,i]:= PLA2_INIT[j,i];
   cas j de
      a: ADR_STATION.A0[i]:= PORT[a,i];
      b: ADR_STATION.A1[i]:= PORT[b,i];
      c: ADR_STATION.A2[i]:= PORT[c,i];
      d: ADR_STATION.A3[i]:= PORT[d,i];
      e: RD0[i]:= PORT[e,i];
   fcas;
 fpour;
fpour;
REG_ADRESSE.A16:= ADR_STATION.RA {stockage de l'adresse physique du circuité
REG_DIRECTION.TRD0:= RD0;        {stockage de la valeur définissant le nombre
REG_DIRECTION.TRD1:= RD1;           de voies en entrée et/ou en sortie du circuité
f INITIALISATION;
```

```
{***********************************************}
{***************** EMISSION TRAMES ***************}
{***********************************************}

 (Cette procédure décrit la transmission d'une trame de nom lorsqu'il
s'agit d'une station distributrice ou d'une trame de données dans le cas
d'une station secondaire)
 ( Une trame peut être formée d'une ou plusieurs champs d'information. C'est
la station à travers la variable DONNEE_PRETE_E qui indique la transmission d'une
nouvelle données ou non. Lorsqu'elle est vrai signifie qu'il y a une donnée
à émettre, dans le cas contraire indique fin des données et le circuit devra
insérer le CRC )
proc EMISSION_TRAMES(MODE,ETAT_EMISSION_TRAME);
const
    taille_trame= ...; ((n+1).(m nombre de mots ou adresses) +preambule +
                        synchro + (crc+1) + séquence de fin; n=crc=16)
type
    bit= 0..1;
    VAL_PREAMBULE= tableau [0..5] de bit;
    VAL_SYN_NOM= tableau [0..2] de bit;
    VAL_SYN_DON= tableau [0..2] de bit;
    VAL_FIN_TRAME= tableau [0..1] de bit;
    infos= (champ, bits);
    tchamp= tableau [0..15] de bit;
    trames=  struct
               cas champs de
                 champ: ( TPREAMBULE: VAL_PREAMBULE;
                          TSYN_NOM: VAL_SYN_NOM;
                          TSYN_DON: VAL_SYN_DON;
                   TDONNEES_EMISSION: tchamp;
                             TCRC: tchamp;
                        TFIN_TRAME: VAL_FIN_TRAME; )
                  bits: (contenu: tableau [1..taille_trame] de bit)
               fin;
           fin;
var
       DONNEE_PRETE_E: booleen;
                 ^RTS: bit;
           PREAMBULE: VAL_PREAMBULE;
             SYN_NOM: VAL_SYN_NOM:
             SYN_DON: VAL_SYN_DON:
           FIN_TRAME: VAL_FIN_TRAME:
               TRAME: trames;
        SERIALISATEUR: tchamp;
             GEN_CRC: tchamp;
  ETAT_EMISSION_TRAME: string;

début;
DONNEE_PRETE_E:= "vrai";  ( variable externe au circuit provenant de la station )
ATTENTE;                  ( propagation de la trame de nom sur le réseau )
si DONNEE_PRETE_E
   alors REG_ETAT.MOTE:= 0; ( remise à zéro du registre d'état)
         ^RTS:=0;          {validation circuit de transmission sur la ligne sérieè
         TRAME.PREAMBULE:= PREAMBULE;            {émission PREAMBULE}
         CODAGE(TRAME);    écodage des données binaires en code Manchester}
         si MODE= 1 alors TRAME.SYN_NOM:= SYN_NOM{émission SYNCHRO_NOM}
                          VCODE(TRAME);
         sinon TRAME.SYN_DON:= SYN_DON       { émission SYNCHRO_DONNEES}
               VCODE(TRAME); fsi;
         CONTROLE_CRC:=1;
         tantque DONNEE_PRETE_E faire
            DONNEE_PRETE_E:= faux;           { acquitemment)
            TRAME.BIT_PARITE:= 1;             { émission BIT_PARITE}
            CODAGE (TRAME)
            SERIALISATEUR:= PORT;
            DECOMPTEUR_M16:= 16;
            tantque DECOMPTEUR_M16>0 faire
               TRAME.DONNEES_EMISSION:= SERIALISATEUR [16-DECOMPTEUR_M16];
               CODAGE (TRAME)
               DECOMPTEUR_M16:= DECOMPTEUR_M16 - 1;
            ftq;                 ( la transmission des données est faite
          ftq;                     à partir du bit de poids le plus faible)
          TRAME.BIT_PARITE:=1;
          CODAGE (TRAME)
          DECOMPTEUR_M16:= 16;
          tantque DECOMPTEUR_M16 > 0 faire
             TRAME.CRC:= GEN_CRC[ DECOMPTEUR_M16];{ la transmission du CRC commence
             CODAGE (TRAME)                        par le coefficient du terme le
             DECOMPTEUR_M16:= DECOMPTEUR_M16 - 1;   plus élevé}
          ftq;
          TRAME.BIT_PARITE:= 0;
          CODAGE (TRAME)
          TRAME.FIN_TRAME:= VAL_FIN_TRAME;       {émission FIN_TRAME}
          ^RTS:= 1;                    {fin validation circuit de transmission}
          ETAT_EMISSION_TRAME:= "correct";
sinon REG_ETAT.ERREUR:= "BOURRAGE";
      ETAT_EMISSION_TRAME:= "incorrect";
      SIGNAL_ERREUR:= 1;
fsi;
f EMISSION_TRAME;
{***********************************************}
{******************** REVEIL *******************}
{***********************************************}

 ( La station distributrice arme une temporisation pour surveiller si une
trame  de données est transmise.
    Si la sonnerie du temporisateur sonne avant que le PREAMBULE de la trame de
données soit détecté, le DISTRIBUTEUR considère qu'il y a eu un problème
dans la transmission de la trame de nom et propose une nouvelle trame.
    Dans le cas contraire, le DISTRIBUTEUR surveille la transmission de la trame
de données, et à la fin de celle-ci propose une nouvelle trame de nom )

proc REVEIL(SONNERIE);
début;
SONNERIE:= faux;
DECOMPTEUR_M16:= 32;
tantque DECOMPTEUR_M16 >0 et non D_PREAMBULE faire
   DECOMPTEUR_M16:= DECOMPTEUR_M16 - 1;
ftq;
si DECOMPTEUR_MOD16 = 0 et non D_PREAMBULE
   alors SONNERIE:= vrai; fsi;
fREVEIL;
```

```
{***********************************************}
{*************** RECEPTION TRAMES **************}
{***********************************************}

 { Elle permet de décrire la détection des trames qui circulent sur le
réseau et identifie s'il s'agit d'une trame de nom ou une trame de données.
Elle appele la procédure correspondente:
      RECEPTION_TRAME_NOM ou RECEPTION_TRAME_DONNEES}

proc RECEPTION_TRAMES  { réception d'une trame de nom ou de données}
var
            D_PREAMBULE: booleen;{identificateur de détection du préambule}
              D_SYN_NOM: booleen;{identificateur de détection du SYNCHRO_DON}
              D_SYN_DON: booleen;{identificateur de détection du SYNCHRO_DON}
   ETAT_RECEPTION_TRAME: string;
                STATION: string; {indique le rôle de la station: émettrice,
                                  réceptrice, neutre}
début;
D_PREAMBULE:= faux;
D_SYNCHRO_NOM:= faux;
D_SYNCHRO_DONNEES:= faux;
STATION:= "NEUTRE"
ETAT_RECEPTION_TRAME:= "quelconque";

tantque non D_SYNCHRO_NOM et non D_SYNCHRO_DONNEES faire
   tantque non D_PREAMBULE faire
      ATTENTE_PREAMBULE;          { détection du préambule}
   ftq;
      ATTENTE_SYNCHRO;            { détection synchros}
      si D_SYNCHRO_NOM
         alors RECEPTION_TRAME_NOM(STATION,ETAT_RECEPTION_TRAME);
               si ETAT_RECEPTION_TRAME="correct" et STATION="EMETTRICE"
                  alors EMISSION_TRAME(MODE,ETAT_EMISSION_TRAME);
                        D_SYNCHRO_NOM:= faux;
               sinon D_SYNCHRO_NOM:= faux;
               fsi;
      fsi;
      si D_SYNCHRO_DONNEES et (STATION= "RECEPTRICE" ou MODE=1)
         alors RECEPTION_TRAME_DONNEES;
      fsi;
   ftq;
f RECEPTION_TRAMES;
```

```
{**********************************************}
{********* RECEPTION D'UNE TRAME DE NOM *********}
{**********************************************}

 ( Lorsque le début d'une trame de nom a été détectée, cette procédure décrit
l'analyse des adresses contenues dans la trame pour savoir si l'une entre elles
corresponde à l'adresse de la station.
 Elle indique au procédure RECEPTION_TRAMES le type de rôle que la station devra
jouer dans le reste de la communication:

  "EMETTRICE": la station devra émettre la trame de données suivante,
 "RECEPTRICE": le circuit doit recevoire la trame de données suivante et le
               transferer vers la station, et
     "NEUTRE": la station n'est pas concerne dans la communication, elle devra
               attendre une nouvelle trame de nom.

    Cette procédure décrit aussi les types d'erreur qui peuvent survenir dûs
à la transmission. En absence d'erreur la trame est validée, dans le cas contraire
elle devra être annulée.)


proc RECEPTION_TRAME_NOM(STATION,ETAT_RECEPTION_TRAME);
type
     TD.15= tableau [0..14] de bit;
     TD.16= tableau [0..15] de bit;
     deser= struct
                 cas des : booleen de
                      faux: D.15: TD15
                      vrai: D.16: TD16
                 fincas;
            fin;
var
          adresses: booleen;  { indique la réception de l'adresse de l'émetteur}
        D_FIN_TRAME: booleen;  { indique la détection de la fin d'une trame }
              D_CRC: booleen;  { indicateur pour signaler la réception de la trame
                               sans erreur de transmission}
            CRC_REC: TD16;     { gén'rateur et identificateur de CRC}
    DESERIALISATEUR: deser;    { transforme les données série en mot 16 bits}

début;
ADRESSES:= faux;
D_FIN_TRAME:= faux;
ERROR:= faux;
D_CRC:= faux;
CONTROLE_CRC:= 1;
DECOMPTEUR_M16:=16;    { détermine les différents champs de la trame}

tantque non D_FIN_TRAME et non ERROR faire
 lire (DONNEES_REC)
 DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
 si DONNEES_DECODEES = 1    { indique l'existance du bit de parité lié à chaque
                              mot 16 bits }
    alors
      tantque DECOMPTEUR_M16>0  faire
         lire (DONNEES_REC)
         DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
         DESERIALISATEUR.D16[16-DECOMPTEUR_M16] := DONNEES_DECODEES;
         CRC_REC[16-DECOMPTEUR_M16] := DONNEES_DECODEES;
         DECOMPTEUR_M16:= DECOMPTEUR_M16 - 1;
      ftq;
     {lorsque decompteur_ m16= 0}
      si CRC_REC <> ID_CRC
         alors  si non ADRESSES
                   alors si DESERIALISATEUR.D16= REG_ADRESSE.A16 ** comparaison entre
                            alors STATION:= "EMETTRICE";          l'adresse de la station
                                  ADRESSES:= vrai;                 et l'adresse recue)
                                  DECOMPTEUR_M16:=16;
                         sinon si DESERIALISATEUR.D15= REG_ADRESSE.A15
                                  alors STATION:= "RECEPTRICE";
                                        ADRESSES:= vrai;
                                        DECOMPTEUR_M16:=16;
                                sinon STATION:= "NEUTRE";
                                      ADRESSES:= vrai;
                                      DECOMPTEUR_M16:=16;
                                  fsi;
                            fsi;
                   sinon si DESERIALISATEUR.D16 = REG_ADRESSE.A16
                            alors STATION:= "RECEPTRICE";
                                  DECOMPTEUR_M16:=16;
                                sinon STATION:= "NEUTRE";
                                      DECOMPTEUR_M16:=16;
                                fsi;
                         fsi;
       sinon D_CRC:= vrai;
             CONTROLE_CRC:= 0;
             lire (DONNEES_REC)
             DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
             si DONNEES_DECODEES= 0     { indique la détection d'un bit à zéro avant le
                                          drapeau de FIN_TRAME}
                alors ATTENTE_FIN_TRAME { réception du fin de la trame}
                      si D_FIN_TRAME
                         alors ETAT_RECEPTION_TRAME:= "correct";
                               REG_ETAT.ERREUR:= "MANQUE FLAG FIN";
                               ERROR:= vrai;
                      fsi;
              sinon REG_ETAT.ERREUR:= "BIT DE PARITE";
                    ERROR:= vrai;
              fsi;
         fsi;
   sinon REG_ETAT.ERREUR:= "BIT DE PARITE";
         ERROR:= vrai;
   fsi;
ftq;
si D_FIN_TRAME et non D_CRC alors REG_ETAT.ERREUR:= "ERREUR DE CRC";
                                  ERROR:=vrai;
fsi;
si ERROR alors ETAT_RECEPTION_TRAME:= "incorrect";
               SIGNAL_ERREUR:= 1;
fsi;
f RECEPTION_TRAME_NOM;
```

```
{***********************************************}
{******* RECEPTION D'UNE TRAME DE DONNEES *******}
{***********************************************}

 { Cette procédure décrit la réception des trames de données et le transfert
des données vers la station. Elle valide la réception des informations si
la trame a été recue correctement, dans le cas contraire indique une erreur de
transmission.
    S'il s'agit d'une station utilisée dans le mode OBSERVATEUR, elle transmet
vers la station le contenu du registre d'état}

proc RECEPTION_TRAME_DONNEES(ETAT_RECEPTION_TRAME);
var
  NEW_DATA_REC: booleen;
  DATA_PRETE_R: booleen;

début;
NEW_DATA_REC:= faux;   { indicateur: réception d'un mot de données}
D_FIN_TRAME:= faux;
ERROR:= faux;
CRC:= faux;
CONTROLE_CRC:= 1;
DECOMPTEUR_MOD16:=16;
DATA_PRETE_R:= faux;

tantque non D_FIN_TRAME et non ERROR faire
 lire (DONNEES_REC)
 DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
 si DONNEES_DECODEES = 1     { identification du bit de parité}
    alors
        tantque DECOMPTEUR_M16>0  faire
           lire (DONNEES_REC)
           DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
           DESERIALISATEUR[16-DECOMPTEUR_M16]:= DONNEES_DECODEES;
           CRC_REC[16-DECOMPTEUR_M16]:= DONNEES_DECODEES;
           DECOMPTEUR_M16:= DECOMPTEUR_M16 - 1;
        ftq;
       {lorsque decompteur_ m16= 0}
        si D_CRC <>ID_CRC
           alors  si non NEW_DATA_REC
                     alors si MODE= 0
                              alors PORT:= DESERIALISATEUR;
                                    NEW_DATA_REC:= vrai;
                                    DATA_PRETE_R:= vrai;
                                    DECOMPTEUR_MOD16:=16;
                           sinon si OBSERVATEUR=0
                                    alors PORT:= REGISTRES_ETAT;
                                          NEW_DATA_REC:= vrai;
                                          DATA_PRETE_R:= vrai;
                                          DECOMPTEUR_MOD16:=16;
                                 fsi;
                           fsi;
                  sinon REG_ETAT.ERREUR:= "BOURRAGE";
                        ERROR:= vrai;
                  fsi;
         sinon D_CRC:= vrai;
               CONTROLE_CRC:= 0;
               lire (DONNEES_REC)
               DONNEES_DECODEES:= DECODAGE(DONNEES_REC);
               si DONNEES_DECODEES= 0
                    alors ATTENTE_FIN_TRAME   { réception du fin de la trame}
                          si D_FIN_TRAME
                             alors ETAT_RECEPTION_TRAME:= "correct"
                          sinon REG_ETAT.ERREUR:= "MANQUE FLAG FIN";
                                ERROR:= vrai;
                          fsi;
               sinon REG_ETAT.ERREUR:= "BIT DE PARITE";
                     ERROR:= vrai;
               fsi;
          fsi;
 sinon REG_ETAT.ERREUR:= "BIT DE PARITE";
       ERROR:= vrai;
 fsi;
ftq;
si D_FIN_TRAME et non D_CRC alors REG_ETAT.ERREUR:= "ERREUR DE CRC";
                                  ERROR:=vrai;
fsi;
si ERROR alors ETAT_RECEPTION_TRAME:= "incorrect";
               SIGNAL_ERREUR:= 1;
fsi;
f RECEPTION_TRAME_DONNEES;
```

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 15 Titre III de l'arrêté du 5 juillet 1984 relatif aux études doctorales

VU les rapports de présentation de Messieurs

. R. GERBER, Professeur
. J. TAILLEBOIS, Directeur Crouzet

Monsieur DIAZ-NAVA Mario

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, Spécialité "Informatique".

Fait à Grenoble, le 17 juin 1986

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

RESUME:

Dans le cadre de ce travail, il a été spécifié et réalisé un circuit intégré de communication pour le réseau local FIP. Le réseau FIP, projet national, est destiné à permettre la communication entre automates réflexes, capteurs et actionneurs. Le circuit intégré a été spécifié pour permettre soit la connexion de capteurs simples soit la connexion de capteurs intelligents ou des automates au réseau.

La conception de ce circuit intégré "à la demande" a été réalisée à partir d'une méthodologie originale. Cette méthodologie est un autre objectif important de ce travail. Elle est orientée vers la conception de circuits VLSI de communication à partir d'une bibliothèque d'opérateurs flexibles, d'une part pour réduire le temps de conception, et d'autre part pour donner la possibilité aux ingénieurs non-spécialistes en conception de concevoir eux-même leur circuit.

MOTS-CLES:

Réseaux locaux temps réel, circuits intégrés communicateurs, méthodologie de conception, opérateurs flexibles, bibliothèque de cellules.