N° d'ordre : 1288

50376
1985
143

UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

THÈSE

Présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE 3ème CYCLE

(Informatique)

par

Saïd GOURRAM

RESEAU MULTI-NIVEAUX : NOUVEL OUTIL DE MODELISATION DES ORDINATEURS. DEFINITION ET IMPLEMENTATION.

Thèse soutenue le mardi 9 juillet 1985 devant la Commission d'Examen

Membres du Jury	Président	M. LATTEUX
	Rapporteur	P. BAKOWSKI
	Examinateurs	V. CORDONNIER
		M. MERIAUX
		B. TOURSEL

P R O F E S S E U R S CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	I.E.E.A.
M. FOURET René	Physique
M. GABILLARD Robert	I.E.E.A.
M. MONTREUIL Jean	Biologie
M. PARREAU Michel	Mathématiques
M. TRIDOT Gabriel	Chimie
M. VIVIER Emile	Biologie
M. WERTHEIMER Raymond	Physique

P R O F E S S E U R S 1ère classe

M. BACCHUS Pierre	Mathématiques
M. BEAUFILS Jean-Pierre (dét.)	Chimie
M. BIAYS Pierre	G.A.S.
M. BILLARD Jean (dét.)	Physique
M. BOILLY Bénoni	Biologie
M. BOIS Pierre	Mathématiques
M. BONNELLE Jean-Pierre	Chimie
M. BOUGHON Pierre	Mathématiques
M. BOURIQUET Robert	Biologie
M. BREZINSKI Claude	I.E.E.A.
M. CELET Paul	Sciences de la Terre
M. CHAMLEY Hervé	Biologie
M. COEURE Gérard	Mathématiques
M. CORDONNIER Vincent	I.E.E.A.
M. DEBOURSE Jean-Pierre	S.E.S.
M. DYMENT Arthur	Mathématiques

PROFESSEURS 1ère classe (suite)

M. ESCAIG Bertrand	Physique
M. FAURE Robert	Mathématiques
M. FOCT Jacques	Chimie
M. GRANELLE Jean-Jacques	S.E.S.
M. GRUSON Laurent	Mathématiques
M. GUILLAUME Jean	Biologie
M. HECTOR Joseph	Mathématiques
M. LABLACHE COMBIER Alain	Chimie
M. LACOSTE Louis	Biologie
M. LAVEINE Jean Pierre	Sciences de la Terre
M. LEHMANN Daniel	Mathématiques
Mme LENOBLE Jacqueline	Physique
M. LHOMME Jean	Chimie
M. LOMBARD Jacques	S.E.S.
M. LOUCHEUX Claude	Chimie
M. LUCQUIN Michel	Chimie
M. MIGEON Michel Recteur à Grenoble	E.U.D.I.L.
M. MIGNOT Fulbert (dét.)	Mathématiques
M. PAQUET Jacques	Sciences de la Terre
M. PROUVOST Jean	Sciences de la Terre
M. ROUSSEAU Jean-Paul	Biologie
M. SALMER Georges	I.E.E.A.
M. SEGUIER Guy	I.E.E.A.
M. SIMON Michel	S.E.S.
M. STANKIEWICZ François	S.E.S.
M. TILLIEU Jacques	Physique
M. VIDAL Pierre	I.E.E.A.
M. ZEYTOUNIAN Radyadour	Mathématiques

P R O F E S S E U R S 2ème classe

M. ANTOINE Philippe	Mathématiques (Calais)
M. BART André	Biologie
Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mathématiques
M. BELLET Jean	Physique
M. BERZIN Robert	Mathématiques
M. BKOUCHE Rudolphe	Mathématiques
M. BODARD Marcel	Biologie
M. BOSQ Denis	Mathématiques
M. BRASSELET Jean-Paul	Mathématiques
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie
M. CARREZ Christian	I.E.E.A.
M. CAYATTE Jean-Louis	S.E.S.
M. CHAPOTON Alain	C.U.E.E.P.
M. COQUERY Jean-Marie	Biologie
Mme CORSIN Paule	Sciences de la Terre
M. CORTOIS Jean	Physique
M. COUTURIER Daniel	Chimie
M. CROSNIER Yves	I.E.E.A.
M. CURGY Jean-Jacques	Biologie
Mle DACHARRY Monique	Géographie
M. DAUCHET Max	I.E.E.A.
M. DEBRABANT Pierre	E.U.D.I.L.
M. DEGAUQUE Pierre	I.E.E.A.
M. DELORME Pierre	Biologie
M. DELORME Robert	S.E.S.
M. DE MASSON D'AUTUME Antoine	S.E.S.
M. DEMUNTER Paul	C.U.E.E.P.

PROFESSEURS 2ème classe (Suite 1)

M. DENEL Jacques	I.E.E.A.
M. DE PARIS Jean-Claude	Mathématiques (Calais)
Mle DESSAUX Odile	Chimie
M. DEVRAINNE Pierre	Chimie
M. DHAINAUT André	Biologie
Mme DHAINAUT Nicole	Biologie
M. DORMARD Serge	S.E.S.
M. DOUKHAN Jean-Claude	E.U.D.I.L.
M. DUBOIS Henri	Physique
M. DUBRULLE Alain	Physique (Calais)
M. DUBUS Jean-Paul	I.E.E.A.
M. FAKIR Sabah	Mathématiques
M. FONTAINE Hubert	Physique
M. FOUQUART Yves	Physique
M. FRONTIER Serge	Biologie
M. GAMBLIN André	G.A.S.
M. GLORIEUX Pierre	Physique
M. GOBLOT Rémi	Mathématiques
M. GOSSELIN Gabriel (dét.)	S.E.S.
M. GOUDMAND Pierre	Chimie
M. GREGORY Pierre	I.P.A.
M. GREMY Jean-Paul	S.E.S.
M. GREVET Patrice	S.E.S.
M. GUILBAULT Pierre	Biologie
M. HENRY Jean-Pierre	E.U.D.I.L.
M. HERMAN Maurice	Physique
M. JACOB Gérard	I.E.E.A.
M. JACOB Pierre	Mathématiques
M. JEAN Raymond	Biologie
M. JOFFRE Patrick	I.P.A.

PROFESSEURS 2ème classe (suite 2)

M. JOURNEL Gérard	E.U.D.I.L.
M. KREMBEL Jean	Biologie
M. LANGRAND Claude	Mathématiques
M. LATTEUX Michel	I.E.E.A.
Mme LECLERCQ Ginette	Chimie
M. LEFEVRE Christian	Sciences de la Terre
Mle LEGRAND Denise	Mathématiques
Mle LEGRAND Solange	Mathématiques (Calais)
Mme LEHMANN Josiane	Mathématiques
M. LEMAIRE Jean	Physique
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique
M. LOSFELD Joseph	C.U.E.E.P.
M. LOUAGE Francis(dét.)	E.U.D.I.L.
M. MACKE Bruno	Physique
M. MAIZIERES Christian	I.E.E.A.
M. MESSELYN Jean	Physique
M. MESSERLIN Patrick	S.E.S.
M. MONTEL Marc	Physique
Mme MOUNIER Yvonne	Biologie
M. PARSY Fernand	Mathématiques
Mle PAUPARDIN Colette	Biologie
M. PERROT Pierre	Chimie
M. PERTUZON Emile	Biologie
M. PONSOLLE Louis	Chimie
M. PORCHET Maurice	Biologie
M. POVY Lucien	E.U.D.I.L.
M. RACZY Ladislas	I.E.E.A.
M. RAOULT Jean François	Sciences de la Terre
M. RICHARD Alain	Biologie

PROFESSEURS 2ème Classe (suite 3)

M. RIETSCH François	E.U.D.I.L.
M. ROBINET Jean-Claude	E.U.D.I.L.
M. ROGALSKI Marc	Mathématiques
M. ROY Jean-Claude	Biologie
M. SCHAMPS Joël	Physique
Mme SCHWARZBACH Yvette	Mathématiques
M. SLIWA Henri	Chimie
M. SOMME Jean	G.A.S.
Mle SPIK Geneviève	Biologie
M. STAROSWIECKI Marcel	E.U.D.I.L.
M. STERBOUL François	E.U.D.I.L.
M. TAILLIEZ Roger	Institut Agricole
Mme TJOTTA Jacqueline (dét.)	Mathématiques
M. TOULOTTE Jean-Marc	I.E.E.A.
M. TURRELL Georges	Chimie
M. VANDORPE Bernard	E.U.D.I.L.
M. VAST Pierre	Chimie
M. VERBERT André	Biologie
M. VERNET Philippe	Biologie
M. WALLART Francis	Chimie
M. WARTEL Michel	Chimie
M. WATERLOT Michel	Sciences de la Terre
Mme ZINN JUSTIN Nicole	Mathématiques

CHARGES DE COURS

M. ADAM Michel	S.E.S.

CHARGES DE CONFERENCES

M. BAFCOP Joël	I.P.A.
M. DUVEAU Jacques	S.E.S.
M. HOFLACK Jean	I.P.A.
M. LATOUCHE Serge	S.E.S.
M. MALAUSSENA DE PERNO Jean-Louis	S.E.S.
M. NAVARRE Christian	I.P.A.
M. OPIGEZ Philippe	S.E.S.

Je tiens à remercier,

Monsieur M. LATTEUX, Professeur à l'Université de Lille 1, qui a bien voulu me faire l'honneur de présider le Jury de cette Thèse.

Monsieur P. BAKOWSKI, Maître-assistant à l'Université de Lille 1, qui est à l'origine de ce travail et avec qui les discussions furent enrichissantes.

Monsieur V. CORDONNIER, Professeur à l'Université de Lille 1, qui m'a accueilli dans son Laboratoire de Recherche et qui m'a fait une confiance exigeante.

Monsieur M. MERIAUX, Chargé de Recherche au CNRS, qui accepte de juger ce travail malgré de nombreuses autres activités.

Monsieur B. TOURSEL, Professeur à l'Université de Lille 1, d'avoir manifesté beaucoup de curiosité à l'égard de ce travail et pour sa participation à ce Jury.

Les amis de l'équipe dont la bonne humeur et la sympathie ne faillirent jamais.

Madame B. VANDROEMME et Monsieur H. GLANT qui ont assuré avec beaucoup de compétence la frappe et le tirage de cette Thèse.

A mes parents,
A ma femme,
A mes frères et sœurs, leurs époux
et enfants,
A mes proches et amis.

PLAN

INTRODUCTION GENERALE

La conception des systèmes logiques, à très grande échelle d'intégration, pose de plus en plus de problèmes, à cause de leur complexité qui ne cesse pas de croître. Les problèmes sont de trois ordres :

- Description : le cahier des charges doit être clair, les aspects structurels et fonctionnels de primitives de base dans la description doivent être pris en compte.

- Simulation : les outils dont on dispose doivent permettre la vérification fonctionnelle.

- Conception : la structure d'implémentation doit être régulière, afin de faciliter les procédés de synthèse, de tests et de correction d'erreurs.

Il devient impératif de disposer de langage de description multi-niveaux, permettant la description d'un système logique du niveau architecture système jusqu'au niveau le plus détaillé, où le transistor est la primitive de base.

La définition d'un outil de modélisation pour chaque niveau rend difficile le passage entre les niveaux à cause de l'incompatibilité de ces outils. D'où la nécessité d'unifier d'abord les concepts de modélisation puis de définir le langage multi-niveaux.

Notre travail se situe donc dans le cadre de la modélisation et de la description des systèmes logiques et particulièrement les systèmes informatiques.

Nous proposons un outil de modélisation appelé REMUN : REseaux MUlti-Niveaux. L'accent est mis sur :

- la décomposition fonctionnelle

- et la hiérarchie structurelle.

Nous montrerons les applications des REMUN, à différents niveaux de description ; nous détaillerons celle du niveau transfert de registre ; un langage de description de systèmes logiques est défini et implémenté permettant la simulation des systèmes décrits en ce niveau.

Dans le premier chapitre de cette thèse, nous nous intéresserons à l'évolution technologique ; nous mettrons en évidence certaines contraintes que posent les systèmes VLSI, telles que la régularité topologique ou structurelle et la décomposition fonctionnelle. Nous présenterons l'élément actif de base qu'est le transistor, les méthodes de conception des systèmes VLSI ainsi que les topologies de quelques systèmes complexes.

Quelques outils de modélisation seront présentés au second chapitre. Nous formulerons des critiques à leurs sujets tant en ce qui concerne leur réalité matérielle qu'en ce qui concerne leur organisation structurelle.

Dans le troisième chapitre, nous exposerons les notions de base, qui nous ont servi pour la définition des REMUN, à savoir : état de structure et état de contenu. Un REMUN sera défini par ces deux états et par les différentes transitions entre-eux. Nous donnerons les liens existants entre les REMUN et les niveaux de description. Un langage de description des REMUN, LREMUN, est proposé.

Les REMUN sont qualifiés de multi-niveaux pour deux raisons, la première est qu'ils offrent un moyen de décomposition fonctionnelle en plusieurs niveaux systémiques, la seconde est qu'ils permettent la modélisation à différents niveaux de description.

Le niveau transfert de registre est choisi pour l'application détaillée des REMUN. Un langage de description LIDO (Langage Interprété de Description des Ordinateurs) est présenté au quatrième chapitre ; il utilise la décomposition fonctionnelle en plusieurs niveaux : "HORLOGE", "CONDITION", "SEQUENCEUR", "DECODEUR" et "OPERATION", cette décomposition permettra de faciliter l'interprétation d'une description LIDO en LREMUN pour effectuer la simulation du système décrit.

CHAPITRE I

SYSTEMES VLSI : COMPLEXITE, CONCEPTION ET EVOLUTION

PLAN

I. INTRODUCTION

La densité d'intégration de systèmes logiques double tous les ans. Cette densité conduit à des complexités de conception à l'aide d'outils classiques.

Les chiffres montrent que le coût de conception avec les techniques actuelles croît de plus en plus. Ainsi un système de 10^5 transistors nécessite 40 Homme/année pour sa conception [NEM].

L'évolution des systèmes à très grande échelle d'intégration (VLSI*) représente un défi pour les informaticiens et les architectes des ordinateurs ; en effet leur évolution influe sur plusieurs domaines, à savoir les méthodes de conception des systèmes informatiques, leurs structures ainsi que leurs fonctionnements.

Nous présenterons dans ce chapitre l'élément de base, d'un système VLSI, qu'est le transistor ; ses caractéristiques physiques et fonctionnelles.

Quelques méthodes de conception de systèmes VLSI, seront exposées, elles sont basées sur la décomposition fonctionnelle en deux parties globales, partie contrôle et partie opérative. Nous discuterons, le long de cet exposé les avantages que présentent ces méthodes par rapport à d'autres.

Enfin, nous présentons quelques structures de systèmes à structures répétitives dont l'intégration est facile à atteindre.

II. TECHNOLOGIE VLSI

La communauté des concepteurs de systèmes logiques, à très grande échelle d'intégration (Very Large Scale Integration : VLSI), utilise les structures de commutation comme moyen d'expression des unités fonctionnelles, en disposant de transistors MOS comme éléments de base de ces structures [MEA].

---

* VLSI : Very Large Scale Integration.

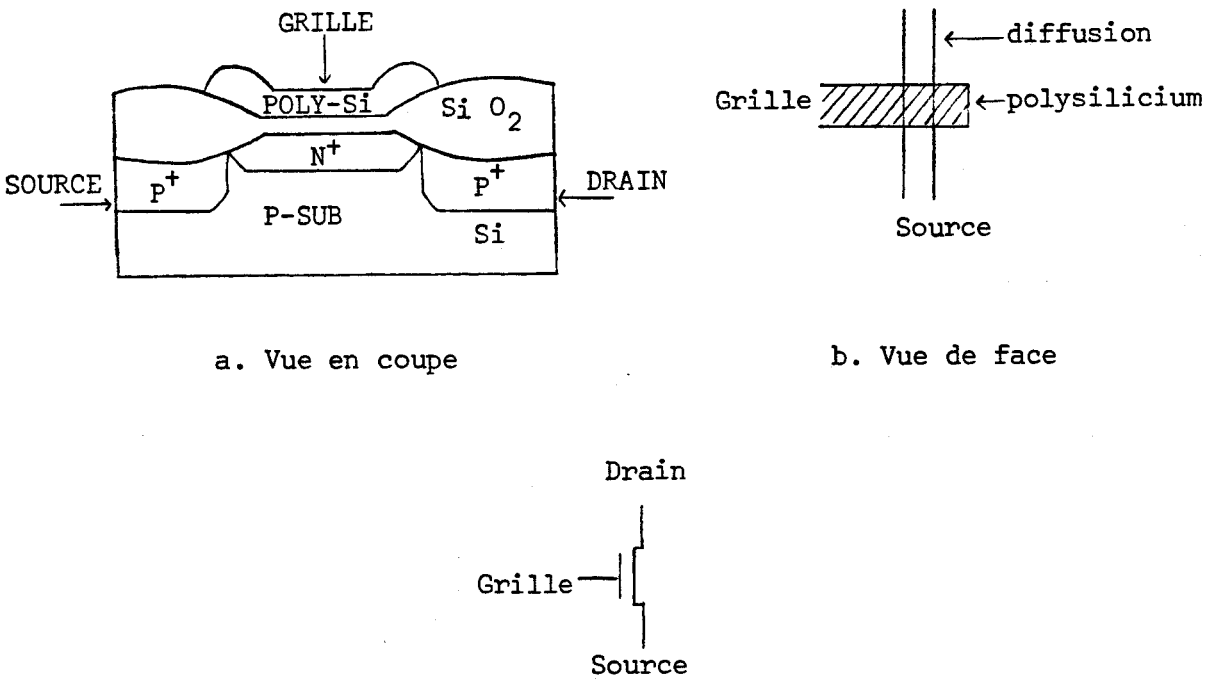
II.1. Transistor MOS

Un transistor à effet de champ (FET), à canal négatif (N), en Métal-Oxyde-Semiconducteur (MOS), est un dispositif physique présentant deux états structurels stables.

II.1.1. Génération physique

La figure 1.a donne une vue en coupe d'un transistor N-MOS. Sur un substrat semi-conducteur de type P, deux régions de type N sont diffusées, appelées **source** et **drain** et entre lesquelles se trouve une couche isolante de silice ($Si\ O_2$) surmontée d'un conducteur en silicium polycristallin (POLY-Si) et appelée **grille** [DUR].

La génération d'un transistor MOS correspond, alors, au croisement d'une liaison en polysilicium et d'une liaison de diffusion, comme le montre la figure 1.b.



a. Vue en coupe

b. Vue de face

c. Représentation symbolique

Figure 1 : Le transistor MOS-VLSI.

II.1.2. Etats structurels

Lorsqu'une tension de 5 volts ("1" logique) est appliquée à la grille, un canal induit se forme entre la source et le drain ce qui rend le transistor MOS conducteur.

Lorsqu'une tension de 0,2 volt ("0" logique) est appliquée à la grille, le transistor MOS se bloque ce qui le rend non conducteur.

Le transistor MOS peut être, ainsi, fonctionnellement assimilé à un interrupteur [ANC 1] présentant les deux états structurels suivants :

- état structurel ouvert qui correspond à G = "0",

- état structurel fermé qui correspond à G = "1".

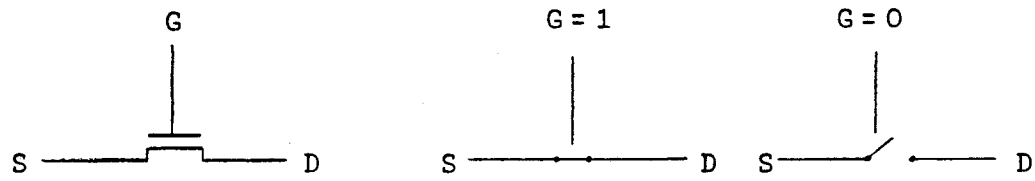


Figure 2 : Etats structurels d'un transistor MOS-VLSI.

II.2. Circuits MOS de base

Quelques circuits combinatoires de base seront présentés, pour chacun seront décrits la structure d'implémentation, le symbole logique et la fonction.

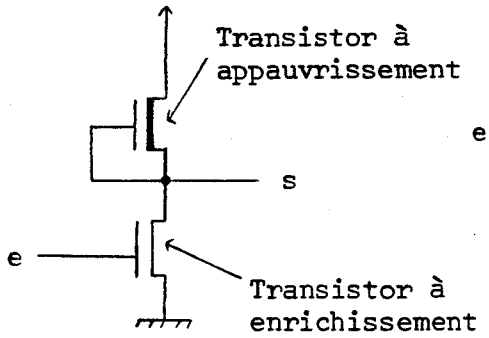
II.2.1. Inverseur

La structure d'implémentation d'un inverseur est composée de deux transistors de deux types différents :

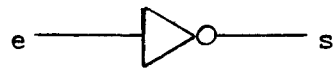
- L'un du type transistor à enrichissement jouant le rôle d'interrupteur, exposé ci-dessus.

- L'autre appelé transistor à appauvrissement jouant le rôle d'une résistance.

La figure 3 montre la structure d'implémentation, le symbole logique et la fonction d'un transistor ; cette fonction est présentée par une table de vérité.



a. Représentation électrique.



b. Représentation logique.

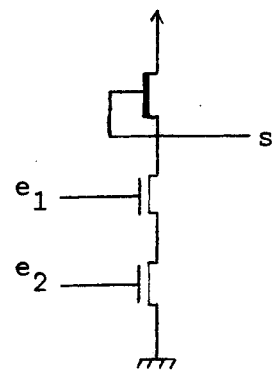
e	s
0	1
1	0

c. Représentation fonctionnelle.

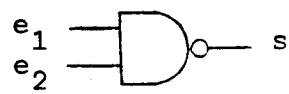
Figure 3 : Représentation d'un inverseur.

II.2.2. Opérateur NON-ET

La représentation électrique d'un opérateur NAND correspond à la disposition sérielle de deux transistors à enrichissement comme le montre la figure 4.a.



a. Représentation électrique.



b. Représentation logique.

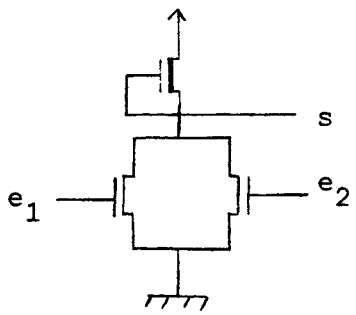
e_1	e_2	s
0	0	1
0	1	1
1	0	1
1	1	0

c. Représentation fonctionnelle.

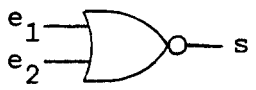
Figure 4 : Opérateur NAND.

II.2.3. Opérateur NON-OU

La représentation électrique d'un opérateur NOR correspond à la disposition parallèle de deux transistors à enrichissement comme le montre la figure 5.a.



a. Représentation électrique.



b. Représentation logique.

e_1	e_2	s
0	0	1
0	1	0
1	0	0
1	1	0

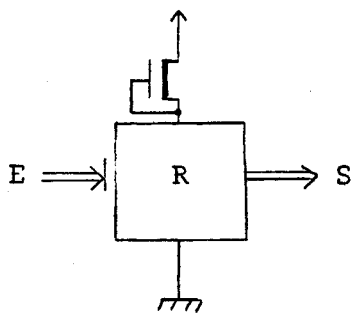
c. Représentation fonctionnelle.

Figure 5 : Opérateur NOR.

II.3. Circuits MOS complexes

La représentation électrique d'un système logique, en général, est obtenue en établissant des interconnexions sérielles et/ou parallèles entre les transistors.

Le réseau de ces transistors peut être représenté par une "boîte-noire" dont la structure interne découle de la fonction logique qu'elle réalise [MEA]. La représentation de cette "boîte-noire" est montrée dans la figure 5.



R : Réseau de transistors
E : Ensemble fini d'entrées
S : Ensemble fini de sorties
⟹| : Ensemble de liaisons entre les entrées et les grilles des transistors MOS.

Figure 5 : Représentation de circuits en technologie MOS.

II.3.1. Réalisation classique

La réalisation classique d'un tel circuit s'effectue à l'aide d'opérateurs capiteux (NAND ou NOR) ou d'opérateurs de base (ET, OU, NON).

Cette solution, utilisant des opérateurs logiques, présente plusieurs inconvénients structurels, temporels et de consommation. Elle entraîne :

- La non régularité de la structure.
- L'utilisation abondante de la surface d'implémentation.
- L'augmentation du temps de propagation.
- La dissipation importante de puissance.

II.3.2. Réalisation VLSI

La réalisation VLSI, utilisée actuellement dans le domaine de conception des circuits intégrés, applique l'ancienne logique de commutation à relais utilisant la commutation de contacts dont les éléments sont uni-directionnels.

On donnera comme exemple d'illustration, le schéma électrique d'un multiplexeur, à quatre entrées (e_0, e_1, e_2, e_3), deux commandes (c_0, c_1) et une sortie s définie par l'expression logique suivante :

$$s = \bar{c}_1\,\bar{c}_0\,e_0 + \bar{c}_1\,c_0\,e_1 + c_1\,\bar{c}_0\,e_2 + c_1\,c_0\,e_3$$

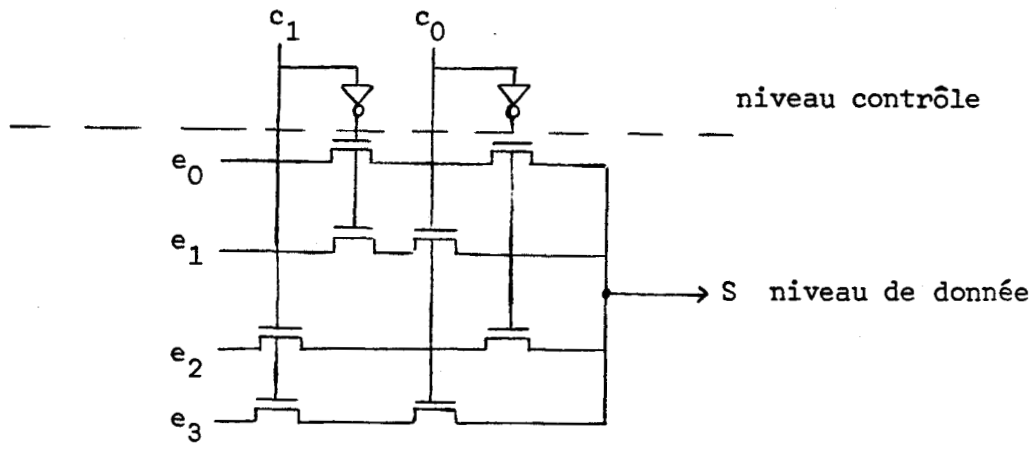


Figure 6 : Représentation électrique d'un multiplexeur.

Il a été largement montré la nécessité de décomposer fonctionnellement de tels systèmes en deux niveaux :

- niveau contrôle,
- niveau de données.

Nous représentons cette décomposition fonctionnelle de la façon suivante :

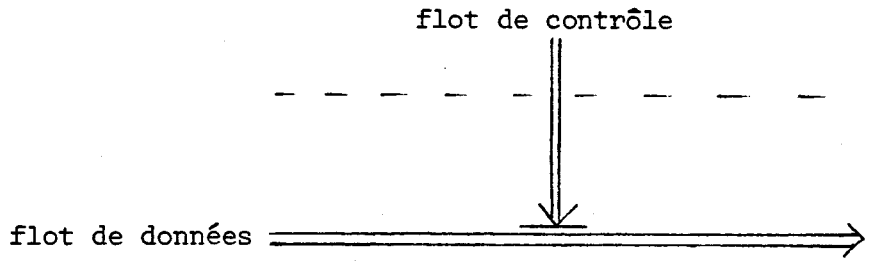


Figure 7 : Décomposition fonctionnelle.

Les méthodes de conception des systèmes VLSI, doivent tenir compte de leur décomposition fonctionnelle afin de réduire leur complexité [MAR]. Cette complexité sera définie dans le paragraphe suivant.

III. SYSTEME VLSI : COMPLEXITE ET EVOLUTION

La complexité d'un système VLSI est définie par évaluation du nombre de transistors assurant la fonctionnalité du système.

Cette complexité double tous les ans, comme le montre la loi de Moore (Figure 8) qui n'a jamais été démentie depuis 1959. Ceci entraîne le doublement de densité d'intégration aussi.

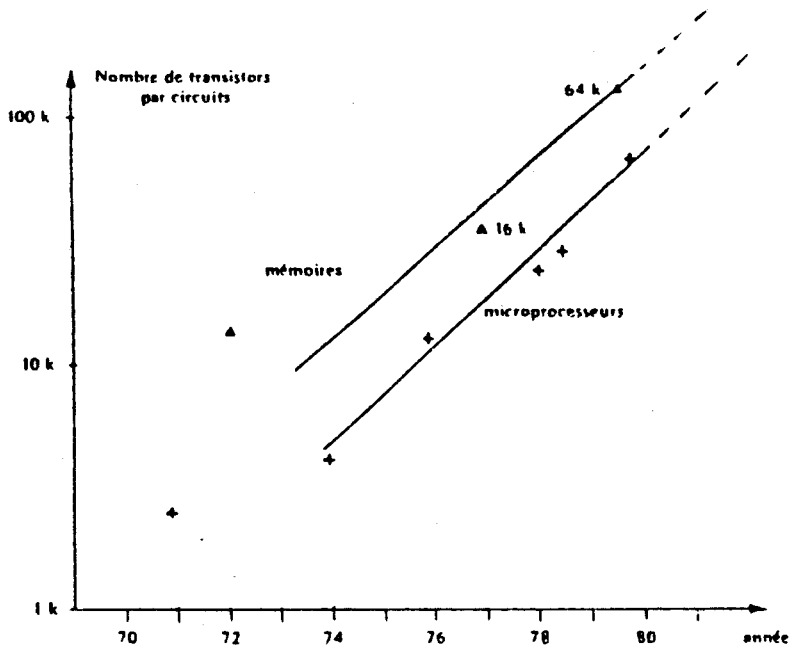


Figure 8 : Evolution de la complexité des systèmes logiques.

Cette loi ne risque pas d'être démentie dans les années futures puisque les principes de la physique indiquent que le taille d'un transistor peut être réduite, encore, à moins d'un centième, tout en gardant la propriété de commutation de base [NEM].

Les systèmes VLSI posent des problèmes pour les informaticiens et les architectes des ordinateurs [MEA] ; en effet leur évolution influe sur les domaines suivants :

- La conception des systèmes informatiques.
- La structure des systèmes informatiques.
- Le fonctionnement des systèmes informatiques.

Nous exposerons dans le paragraphe ci-après les influences de l'évolution de systèmes VLSI dans ces trois domaines.

IV. INFLUENCES DE L'EVOLUTION DE SYSTEMES VLSI

Nous nous intéressons le long de ce paragraphe aux deux notions fondamentales suivantes :

- Décomposition fonctionnelle : consiste à décomposer un système en plusieurs unités fonctionnelles afin de rendre claire sa représentation.

- Hiérarchie structurelle : consiste à structurer le système d'une manière hiérarchique de façon à faciliter son implémentation.

IV.1. Conception de systèmes logiques

Les systèmes logiques ont pour rôle l'interprétation algorithmique et l'exécution d'opérations élémentaires sur les données.

La représentation d'un système logique, introduite par Glushkov [GLU], consiste à décomposer le système en deux entités fonctionnelles distinctes et à spécifier leurs interconnexions.

Cette représentation connue sous le nom de "algorithmic state machine" définit alors deux parties, dont chacune a un rôle fonctionnel spécifique :

1. La partie opérative est chargée d'exécuter des opérations élémentaires et d'envoyer des indications sur le résultat de chaque exécution à l'autre partie.

2. La partie contrôle pilote la partie opérative en l'examinant d'abord, pour assurer le séquencement, puis, en envoyant des commandes d'activation des opérations élémentaires.

L'interaction entre ces deux parties est illustrée dans la figure ci-dessous.

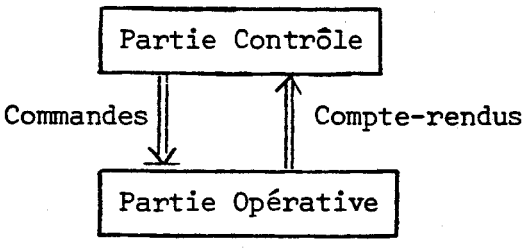


Figure 9 : Décomposition fonctionnelle d'un système logique.

Cette décomposition est largement utilisée dans le domaine de la conception, bien qu'elle ne soit pas toujours liée à la structure topologique réelle des systèmes.

L'intérêt principal, qu'offre cette décomposition, est de permettre la conception de chaque partie indépendamment de l'autre ; ce qui entraîne la spécification de méthodes de conception pour chaque partie.

Nous citerons ci-après quelques méthodes de conception pour la partie contrôle et pour la partie opérative.

IV.1.1. Méthodes de conception de parties contrôles

La réalisation matérielle de la partie contrôle a été l'objet de plusieurs travaux de recherche, ces dix dernières années [FAS], [OBR].

Actuellement, on trouve essentiellement trois structures matérielles de la partie contrôle qui sont exploitées :

- Structures cablées (ou anarchiques).
- Structures à algorithme enregistré (régulières).
- Structures mixtes (combinaison de ces deux structures).

Une étude comparative de différentes méthodes de conception de parties contrôles a été effectuée par M. Obrebska [OBR] ; les chiffres qui seront cités par la suite découlent de cette étude.

IV.1.1.1. Réalisation cablée

La réalisation cablée consiste à interpréter l'algorithme de contrôle en terme de portes logiques et de cellules de mémorisation élémentaires.

Cette solution, bien que représentant la solution la plus avantageuse pour l'implémentation de très grande densité, est actuellement de plus en plus délaissée en raison des complexités structurelles, ou topologiques, qu'elle introduit et qui rendent difficile le test.

En plus, la majorité des conceptions à l'aide de cette méthode se termine manuellement [ETI], ce qui la rend coûteuse en temps, puis en nombre d'erreurs humaines.

L'exemple le plus connu, de ce type de conception, est la réalisation de la partie contrôle du MC 6800 de Motorola, qui a été effectuée en 1976, représentant une complexité de l'ordre de 5000 transistors.

La représentation de cette réalisation est illustrée par la figure 10.

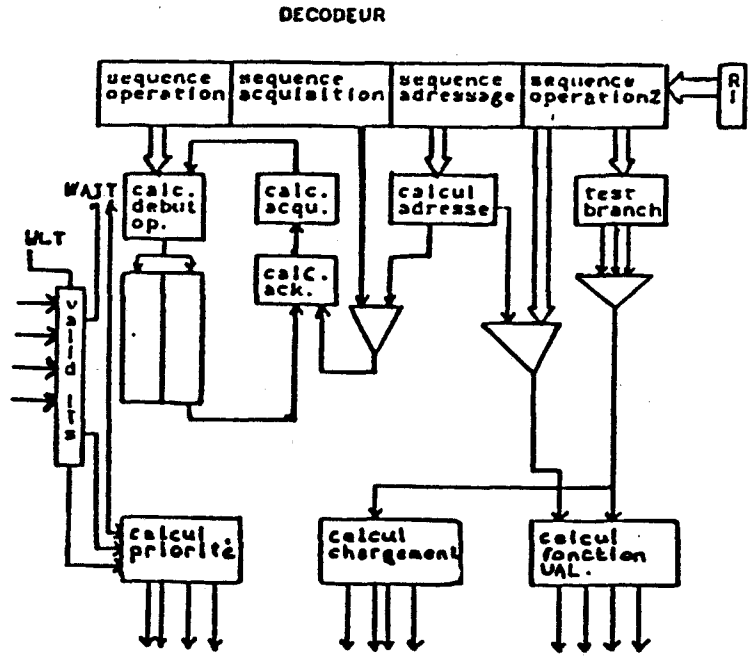


Figure 10 : Représentation de la partie contrôle du MC 6800.

On remarque que cette réalisation a entraîné la décomposition de la partie contrôle en plusieurs entités fonctionnelles :

- entité de séquencement,
- entité de décodage,... etc.

IV.1.1.2. Réalisation à algorithme enregistré

Ce type de réalisation présente comme caractéristiques principales :

- Régularité structurelle.
- Simplicité des méthodes de conception et d'implémentation.
- Facilité de correction d'erreurs.

En effet, l'algorithme de contrôle sera interprété sous forme d'une matrice de transistors ; cet aspect matriciel de la réalisation implique les avantages structurels et conceptuels.

L'usage de ce type de structure à algorithme enregistré nous amène à deux méthodes distinctes :

- Conception à l'aide de PLA's (Programmable Logic Arrays : réseau logique programmable).
- Conception microprogrammée.

IV.1.1.2.1. Conception à l'aide de PLA

La réalisation d'une fonction logique à l'aide de PLA, consiste à spécifier deux matrices de transistors, une matrice "ET" et une matrice "OU" (Figure 11).

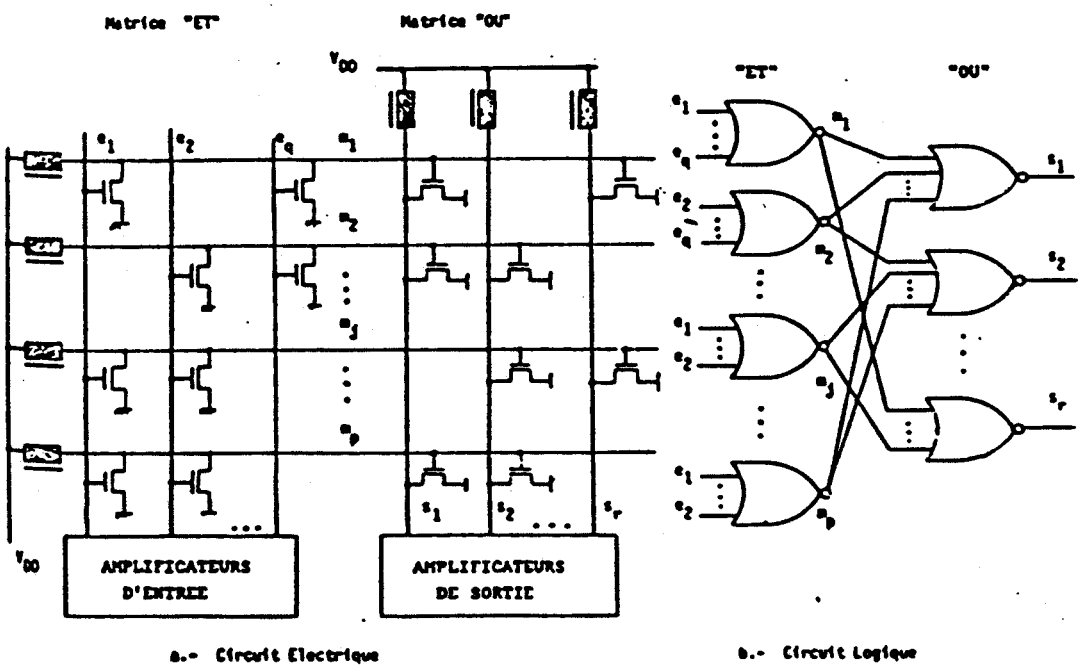


Figure 11 : Structure des PLA.

La décomposition fonctionnelle de l'algorithme de contrôle est pratiquée pour faciliter la conception de parties contrôles.

La conception de parties contrôles consistera, alors, à spécifier d'une part les PLA correspondants à chaque sous-algorithme et d'établir d'autre part les connexions entre-eux [FAS].

On donnera comme exemple la représentation de la partie de contrôle du MC 6800 à base de PLA, étudiée à Grenoble [ETI].

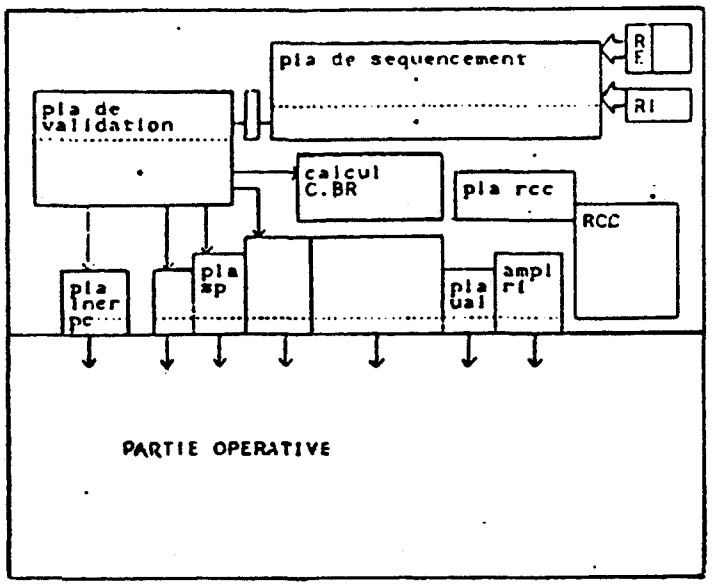


Figure 12 : Représentation de la partie contrôle du MC 6800 à l'aide de PLA.

L'inconvénient de cette réalisation reste encore l'importance de la surface occupée ; qui est dûe au fait que les matrices "ET" et "OU" sont creuses.

Bien que des taux d'optimisation en surface aient atteint entre 30 et 50 % de réduction, à l'aide de systèmes CAO tel que POALA [CHS], cette réalisation reste désavantageuse en occupation de surface par rapport à la réalisation câblée, mais elle est plus adaptée à la conception des systèmes VLSI, à cause de sa régularité structurelle et de sa facilité d'implémentation.

IV.1.1.2.2. Réalisation microprogrammée

Un microprogramme est défini par une suite de micro-instructions, dont chacune est composée de deux champs fonctionnels :

- un champ adresse spécifiant l'adresse de la prochaine micro-instruction à exécuter
- et un champ commande spécifiant le flot d'activation des opérations élémentaires de la partie opérative.

La réalisation microprogrammée consiste à établir le micro-programme correspondant à l'algorithme de contrôle, puis son implémentation en une matrice de transistors formant la mémoire morte (Figure 13).

La matrice de transistors est d'autant plus creuse que le nombre d'opérations activées d'une manière instantanée est minimal par rapport à la taille du champ de commande. Ceci entraîne le problème de l'occupation de la surface de silicium qui est souvent limitée.

En effet, pour un même algorithme de contrôle ; la réalisation microprogrammée occupe 66 % de la surface globale du processeur MC 6800, alors que la réalisation câblée n'en occupe que 57,5 %.

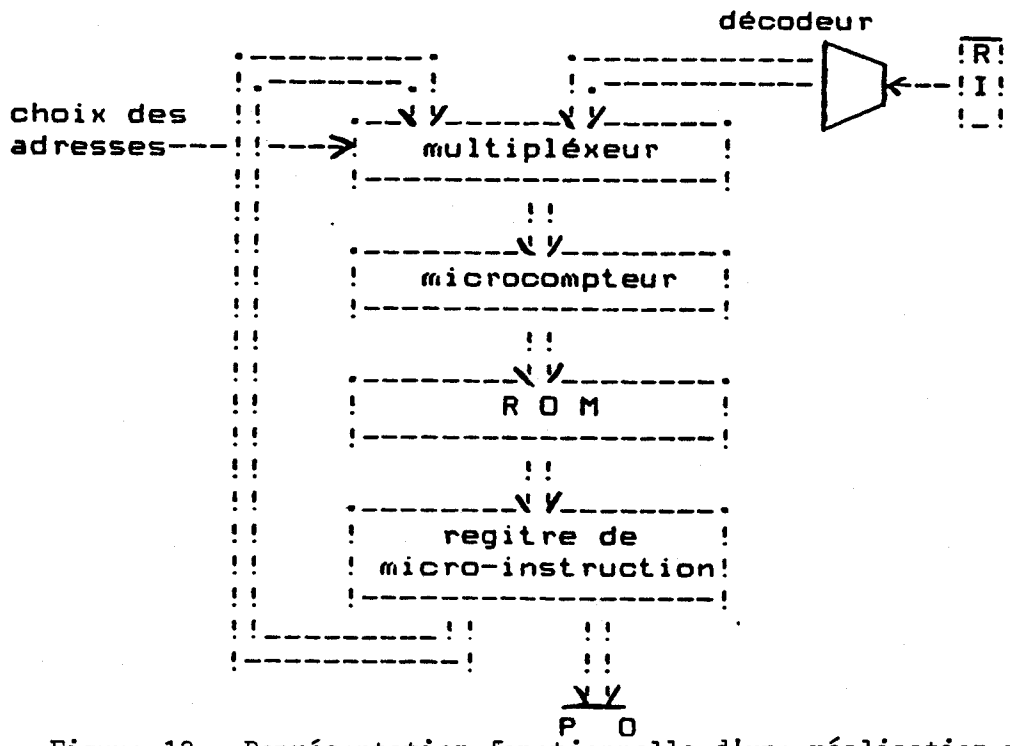


Figure 13 : Représentation fonctionnelle d'une réalisation microprogrammée.

IV.1.1.3. Remarques

Les méthodes de conception de parties contrôles visent essentiellement à atteindre les deux buts suivants :

- Régularité structurelle.
- Réduction de la surface d'implémentation.

Les différentes réalisations qu'on vient de présenter, atteignent généralement l'un ou l'autre.

La solution microprogrammée est de plus en plus abandonnée ; alors que la solution à l'aide de PLA, munie de recours manuels donc câblés (solution mixte), acquiert de plus en plus de succès ces dernières années.

IV.1.2. Méthode de conception de parties opératives

La partie opérative d'un système logique a pour fonction l'exécution proprement dite des opérations élémentaires (arithmétique, logique, de transfert), activées par la partie contrôle et l'envoi à cette dernière des indications sur le résultat de l'exécution.

La partie opérative est composée des trois types d'objets suivants :

- Opérateurs : ayant pour rôle le traitement des données.
- Eléments de mémorisation : permettant le stockage de données (latch, registre, mémoire).
- Eléments de transfert : définissant le cheminement des données d'un élément de mémorisation à un autre, à travers un opérateur ou non.

Toute méthode de conception de parties opératives se décompose globalement en deux étapes :

Etape 1 : Détermination des objets indispensables pour la réalisation.

Etape 2 : Spécification des dispositions structurelles des objets, tendant à rendre cette structure régulière.

IV.1.3. Critiques

F. MAISON a largement critiqué les systèmes d'aide à la conception actuels en ce qu'ils ne sont pas adaptés aux nouvelles possibilités de la technologie [MAI] et ne peuvent subir cette adaptation.

Il convient donc de développer une nouvelle génération de CAO qui, malheureusement risque fort d'être incompatible avec la précédente.

A notre avis de tels outils de CAO doivent tenir compte de :

- la décomposition fonctionnelle d'un système en niveaux systémiques,
- la hiérarchie structurelle,
- la définition d'états structurels de tout niveau systémique.

Ces notions seront mises en évidence par la proposition d'un nouvel outil de modélisation, à savoir, les réseaux multi-niveaux.

Parallèlement aux méthodes de conception qui sont touchées par la mutation technologique ; L'architecture des ordinateurs voit sans cesse évoluer ses concepts structurels et fonctionnels, tout en profitant des apports d'intégration des systèmes VLSI.

IV.2. Influence technologique sur l'architecture des ordinateurs

Les intérêts de l'intégration à très grande échelle sont d'abord économiques, car le coût de la logique peut baisser considérablement ; en effet, les prix de fabrication restent proportionnels à la surface d'implémentation et non au nombre de transistors.

Pour l'architecture des ordinateurs, cette intégration permet la réduction des voies de communication entre les organes d'un ordinateur implémenté sur une seule surface de silicium. L'intérêt de cette réduction est de minimiser les durées d'échange d'informations entre les organes, donc de contribuer à l'amélioration des performances de l'ordinateur.

Les architectes des ordinateurs visent à atteindre, actuellement, des performances de l'ordre 10^5 MIPS, comme le montre la figure 14.

Deux voies sont explorées dans ce domaine :

- Multiplication structurelle : qui conduit à la multiplication des organes de l'ordinateur en définissant des outils adaptés à leur communication.

- Amélioration fonctionnelle : qui consiste à utiliser des modes de fonctionnement permettant la simultanéité d'exécution.

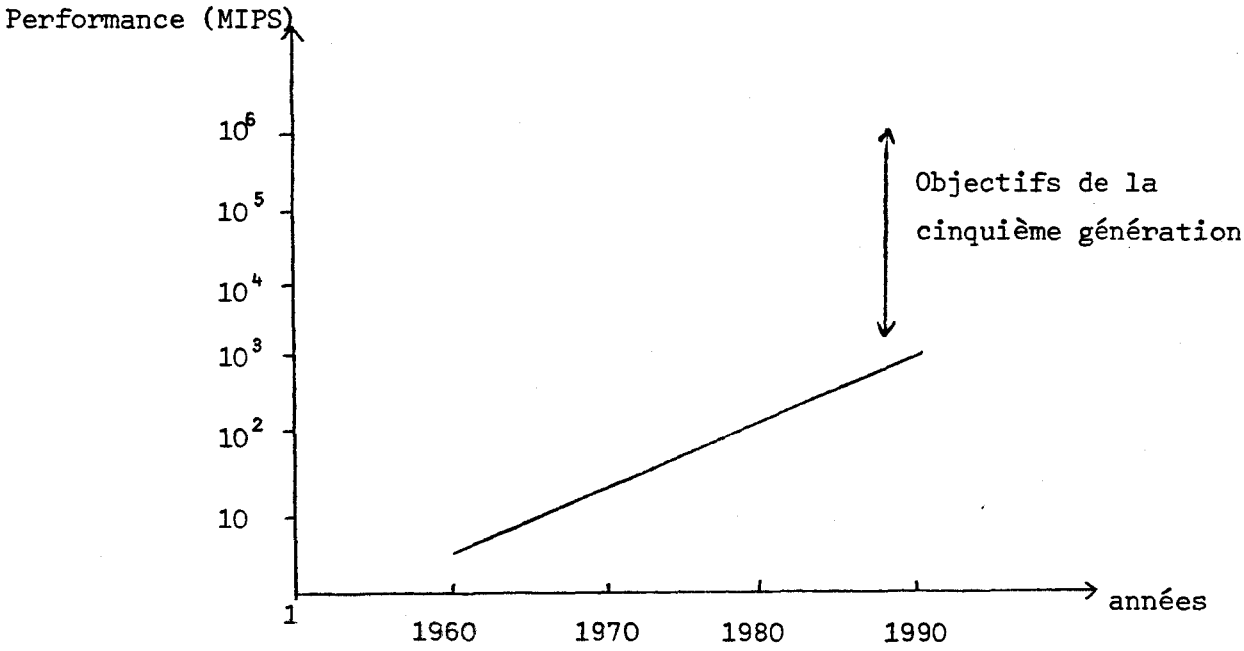


Figure 14 : Performances de la cinquième génération [MIN].

IV.2.1. Multiplicité structurelle

La structure d'un ordinateur est définie classiquement comme l'interconnexion des quatre organes suivants :

- Organe de mémorisation : contenant des instructions et des données.
- Organe de traitement : chargé d'exécuter les opérations élémentaires.
- Organe de contrôle et de commande dirigeant le séquencement des opérations élémentaires. Il est répétitivement chargé :

d'acquérir l'instruction de la mémoire,
de la décoder,
d'acquérir les opérandes,
d'envoyer des commandes à l'organe de traitement,
et de ranger le résultat.

- Un organe d'échange permettant l'établissement des communications avec l'environnement.

Les organes sont interconnectés par des voies de contrôle, de données et d'adresse.

Les architectes des ordinateurs visent à concevoir des systèmes à performances élevées, par le biais de l'exploitation du parallélisme qui a entraîné dans la majorité des travaux à des multiplications d'organes.

Nous discuterons pour chaque cas de ces multiplications structurelles, l'intérêt de l'intégration à très grande échelle.

Pour la clarté de l'exposé, nous allons définir une taxonomie semi-structurelle : bien qu'elle soit rudimentaire, elle nous permettra de faire apparaître les principales structures d'implémentation des ordinateurs.

IV.2.1.1. Taxonomie semi-structurelle

La classification, la plus répandue, est celle qui est proposée par FLYNN, elle concerne la nature de l'exécution d'un programme. Les instructions et les données sont prises en compte.

Cette classification est abstraite par rapport au niveau d'implémentation et rend difficile une définition adaptée à l'implémentation des systèmes.

La taxonomie qu'on se propose ici, est basée sur les quatre propriétés booléennes suivantes :

UC : Unicité de l'organe de Contrôle.
UT : Unicité de l'organe de Traitement.
UM : Unicité de l'organe de Mémorisation.
UP : Unicité de l'organe Périphérique.

Ces propriétés sont définies ainsi

UX = 1 si l'organe X est unique.
= 0 s'il existe plusieurs organes X dans le système.

On fera abstraction des détails matériels d'interconnexions des organes, d'où la notion de semi-structuralité.

Quelques classes de cette taxonomie sont montrées dans la figure 15.

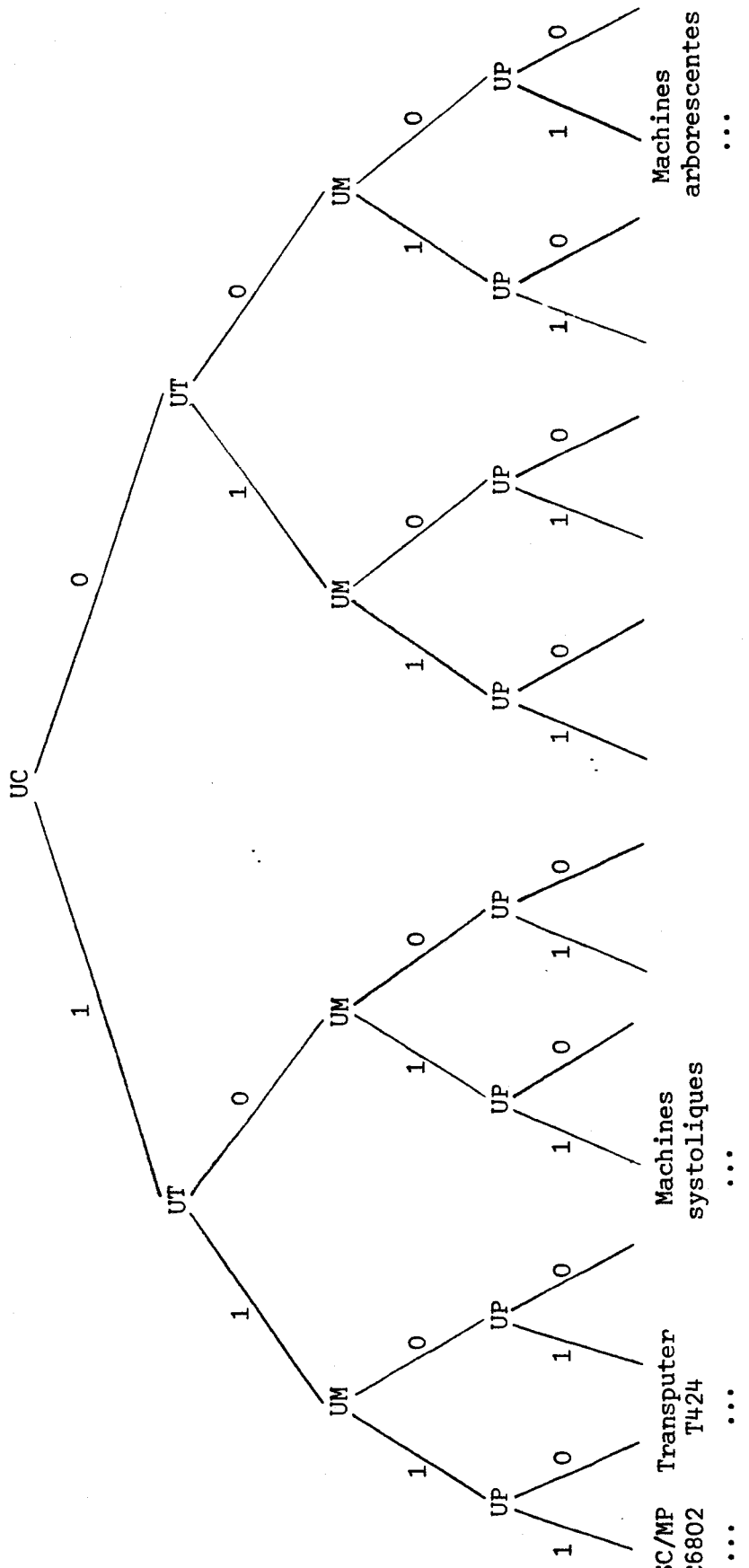


Figure 5 : Classification semi-structurelle.

(UC, UT, UM, UP) :

Cette classe applique les concepts d'un calculateur traditionnel, dit de Von Neumann. Plusieurs calculateurs de cette classe ont été réalisés, sur une seule surface de silicium, à la fin de la dernière décennie. Parmi, ceux-là, on trouve le SC/MP.

Le SC/MP est réalisé en technologie N-MOS en 1977 ; il est présenté dans un seul boîtier de 40 broches. Il exécute 30 instructions et présente quatre modes d'adressage : relatif au compteur ordinal, immédiat, indexé et auto-indexé (le registre index est modifié).

L'organe de contrôle occupe la moitié de la surface d'implémentation, il est conçu à l'aide de PLA.

La mémoire est de 64 octets.

La micro-photographie ainsi que la topologie du CS/MP sont montrées dans les planches 16 et 17 [OBR].

On trouve aussi d'autres calculateurs, de la même génération construits à une petite échelle, comme le CP 1600 (N-MOS) et le INS 8070.

(UC, $\overline{UT}$, UM, UP) :

Il est bien connu que le traitement simultané de plusieurs opérations élémentaires minimise le temps global d'exécution ; d'où l'intérêt de la multiplication des organes de traitement.

Cette classe présente des calculateurs ayant un certain nombre d'unités de traitement, généralement identiques, organisées en réseaux maillés (Fig. 18).

Ces réseaux d'organes de traitement présentent des structures régulières donc simples à implémenter et susceptibles d'effectuer des traitements en chaînes sur des matrices avec un gain optimal de vitesse.

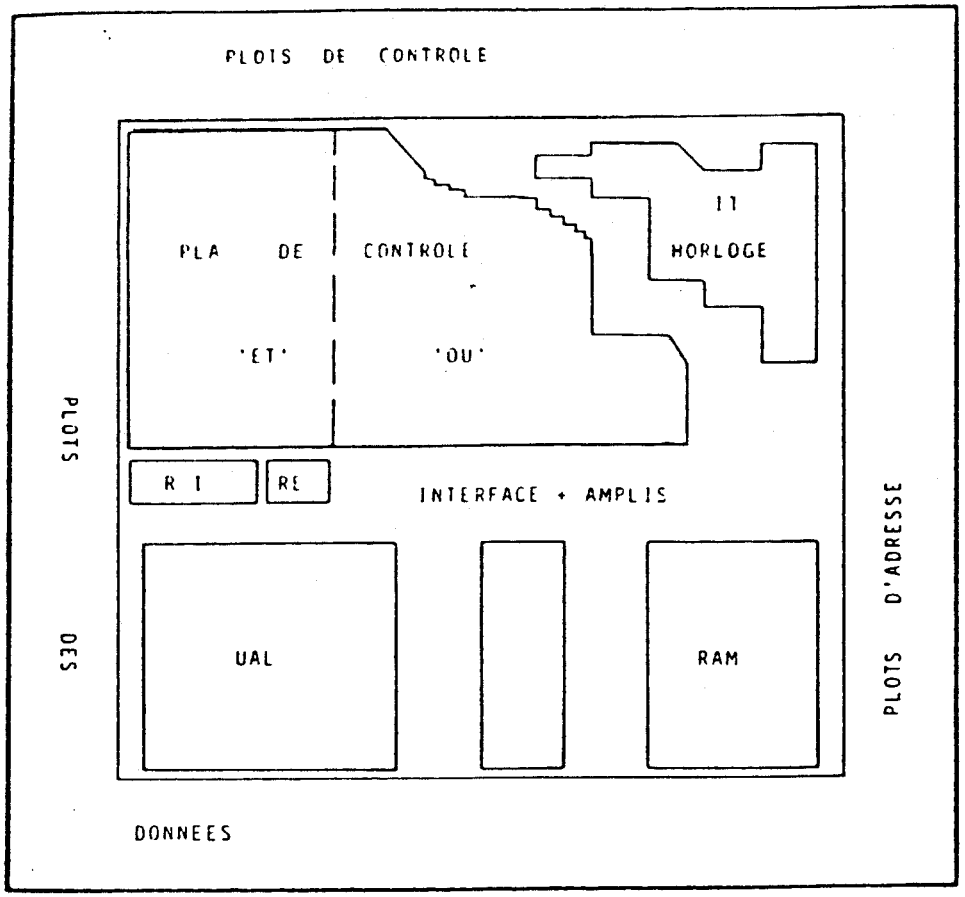


Planche 16 : Topologie du SC/MP.

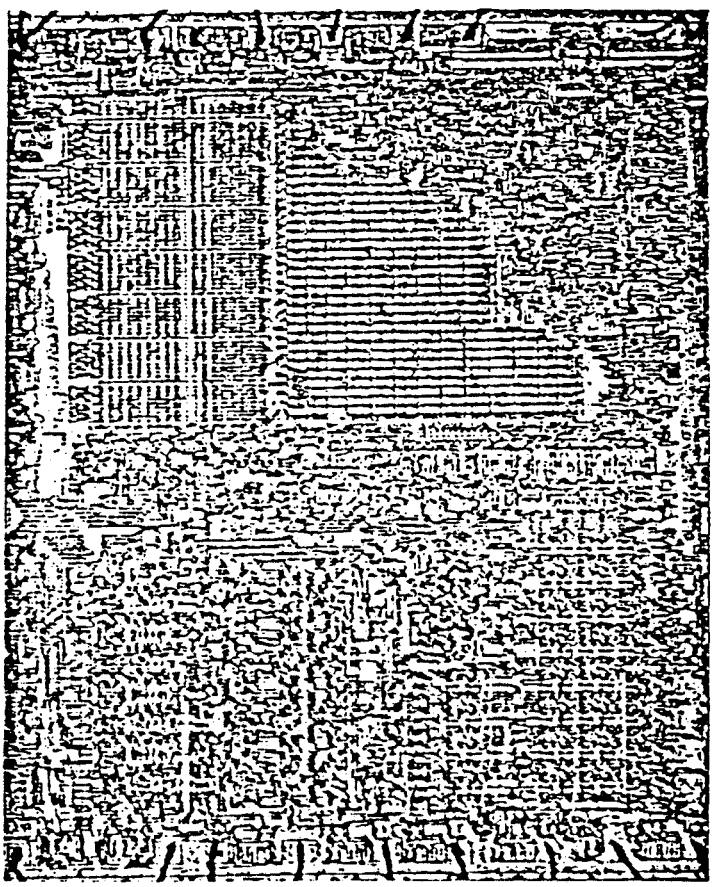


Planche 17 : Photographie du SC/MP.

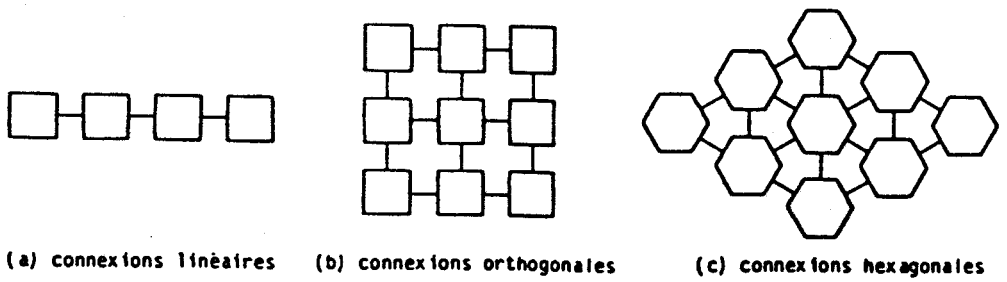


Figure 18 : Réseaux maillés d'organes de traitement [MEA].

($\overline{UC}$, $\overline{UT}$, $\overline{UM}$, UP) :

Cette classe consiste à exécuter plusieurs processus d'une manière simultanée, chaque processeur (combinaison d'un organe de contrôle et d'un organe de traitement) a pour charge l'exécution d'un processus ou la coopération entre les processus.

Comme la structuration régulière facilite l'implémentation, la structure la plus adaptée, pour cette classe, est l'organisation hiérarchique en arbre binaire (Fig. 19).

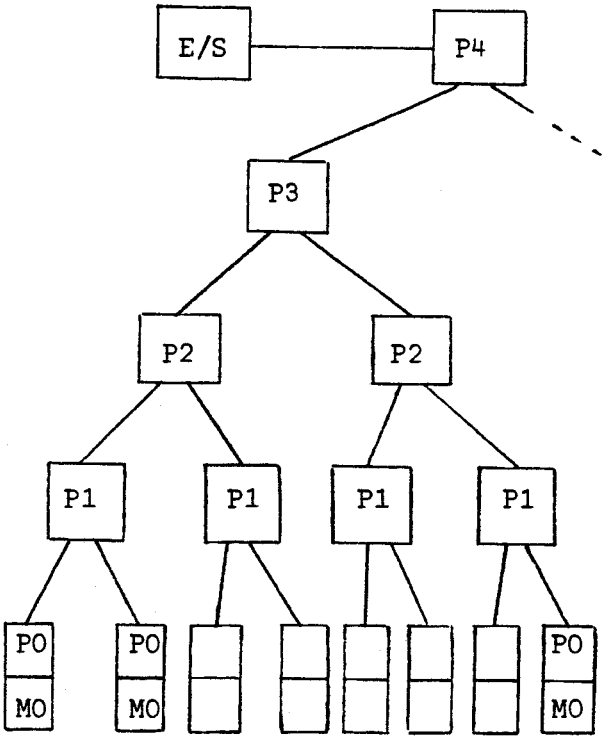


Figure 19 : Organisation arborescente.

La topologie d'implémentation, d'un tel calculateur, est régulière comme le montre la figure 20 :

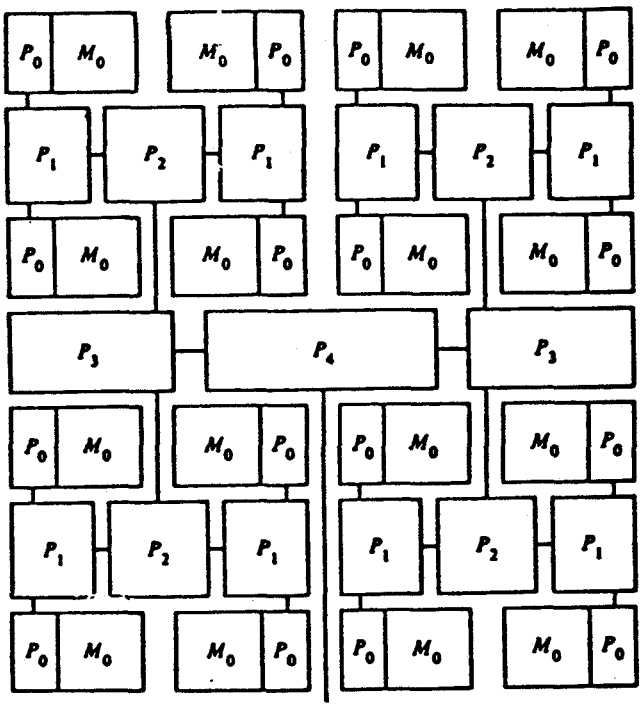


Figure 20 : Topologie d'implémentation [MEA].

(UC, UT, UM, $\overline{UP}$) :

Cette classe de calculateurs est caractérisée par la multiplication de voies de communication avec l'environnement, pour permettre la communication avec d'autres calculateurs, établissant ainsi des réseaux de processeurs adaptés au traitement parallèle.

INMOS a conçu le T424, qui est un calculateur présentant la caractéristique d'une multitude de voies de communication (4 voies).

Le T424 est réalisé en C-MOS, de capacité mémoire 256 K bits et dont le schéma fonctionnel est illustré à la figure 20.

L'intérêt d'intégration d'un tel calculateur est d'offir aux architectes des ordinateurs, la possibilité d'en combiner plusieurs, d'une manière linéaire, tabulaire...

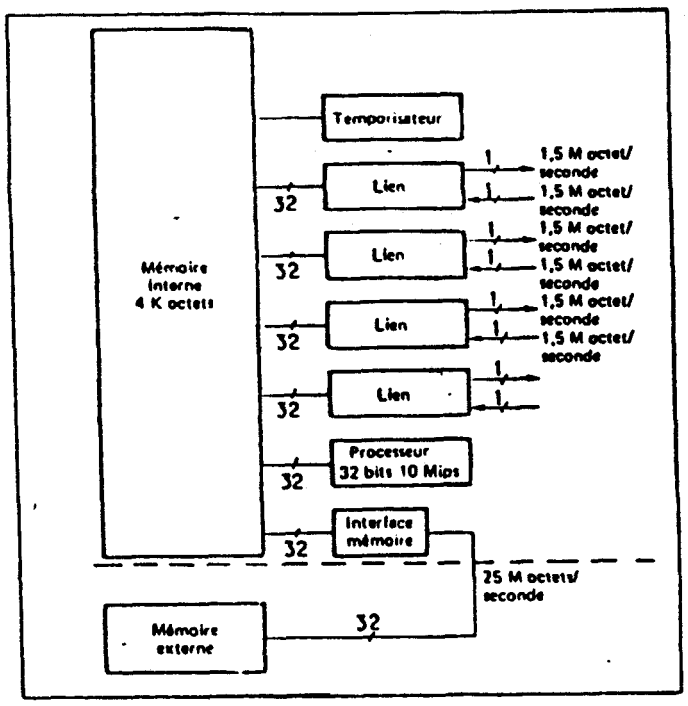


Figure 21 : Schéma fonctionnel du transputer T424 [MIN].

V. CONCLUSION

Durant ce chapitre, nous avons montré l'utilité que représentent la décomposition fonctionnelle et la hiérarchie structurelle dans le domaine de la conception des systèmes logiques à très grande échelle d'intégration.

La complexité de ces systèmes a rendu nécessaire la définition de nouveaux outils de modélisation et de nouvelles méthodes de représentation afin de rester plus proche de leurs réalités matérielles et fonctionnelles facilitant ainsi leur description et leur conception.

CHAPITRE II

OUTILS DE MODELISATION DE SYSTEMES LOGIQUES

PLAN

I. INTRODUCTION

Le but principal, d'un outil de modélisation de systèmes logiques, est de simplifier leurs représentations afin de rendre claires les différentes étapes d'analyse et/ou de conception.

Depuis la naissance des premiers systèmes de traitement de l'information leurs outils de modélisation se sont constamment développés afin de rendre claires leurs représentations structurelles et fonctionnelles.

Ces outils de modélisation sont basés sur des théories mathématiques munies d'axiomes, les relations entre ces théories et les objets matériels doivent être le plus proches que possible.

Le but de ce chapitre est de mettre en évidence l'intérêt de l'organisation dans un outil de modélisation.

Nous présentons dans ce chapitre quelques outils de modélisation, allant de la théorie de l'information introduite par SHANNON jusqu'à des outils développés en cette décennie ; ils seront exposés par ordre historique qui se justifie essentiellement pour des raisons d'évolutions technologiques.

Une transcription graphique sera utilisée pour montrer la relation qui existe entre l'outil de modélisation et la réalité matérielle.

Au cours de cet exposé, on tiendra principalement compte du caractère hiérarchique des modèles présentés.

II. THEORIE DE L'INFORMATION

SHANNON a le mérite d'introduire, en 1939, les notions de la théorie de l'information pour la représentation d'un système de traitement de l'information, afin de calculer les quantités d'information transmises dans une voie de communication entre un dispositif source et un dispositif objet.

Cette théorie a été beaucoup critiquée en ce qu'elle n'est pas un bon moyen de représentation des systèmes de traitement, mais une théorie de communication ; car elle laisse de côté plusieurs problèmes, dont les problèmes liés aux formes hiérarchiques d'organisation, pour lesquels elle ignore totalement la décomposition en niveaux d'organisation d'un système [ATL].

Ces critiques ont conduit au fait qu'un outil de modélisation doit satisfaire la condition suivante :

"Organisation hiérarchique d'un système logique".

GLUSHKOV a proposé, un modèle satisfaisant cette condition ; ce modèle est connu sous le nom de "Algorithmic State Machine" [GLU].

III. ALGORITHMIC STATE MACHINE

Le modèle, introduit par GLUSHKOV, permet d'organiser hiérarchiquement un système logique en deux niveaux fonctionnellement distincts :

- Niveau contrôle : détermine les séquences d'opérations élémentaires activées.
- Niveau opératoire : détermine l'ensemble des opérations élémentaires.

Cette représentation a été présentée au Chapitre I, elle est illustrée dans la figure B.1.

L'avantage de cette représentation est qu'elle permet la spécification d'outils de modélisation pour chaque niveau indépendamment de l'autre, puisque leurs fonctions sont différentes

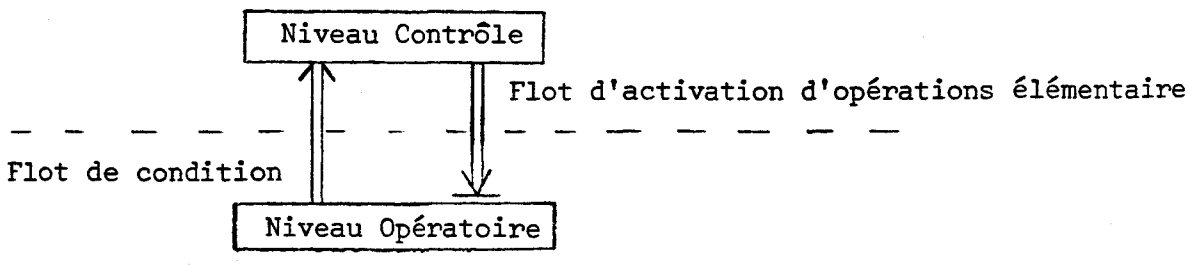


Figure B.1. : Représentation de GLUSHKOV.

Durant les vingt dernières années ; plusieurs outils de modélisation du niveau contrôle ont été développés.

Nous citerons dans les paragraphes suivants les principaux outils de modélisations utilisés en présentant leurs correspondances matérielles.

IV. AUTOMATE D'ETATS FINIS

Un niveau de contrôle peut être représenté, d'une manière abstraite, à l'aide d'un automate d'états finis A qui est défini par la donnée d'un quintuplet, comme suite :

$$A = (Q, \tau, \Sigma, q_0, F)$$

où Q : ensemble fini d'états
Σ : ensemble fini de symboles conditions
τ : fonction de transition entre les états

$$\tau : Q \times \Sigma \rightarrow Q$$

q_0 : état initial
$F \subseteq Q$: ensemble d'états finaux.

Le flot d'activation du niveau opératoire est défini par une fonction de décodage γ, dont les caractéristiques déterminent deux familles de machines séquentielles.

IV.1. Machines séquentielles de MEALY

La fonction γ définit le flot d'activation FA, à partir de l'état courant et de l'ensemble des conditions Σ.

$$FA = \gamma(Q, \Sigma).$$

L'aspect organisationnel des machines de MEALY est montré dans la figure B.2.

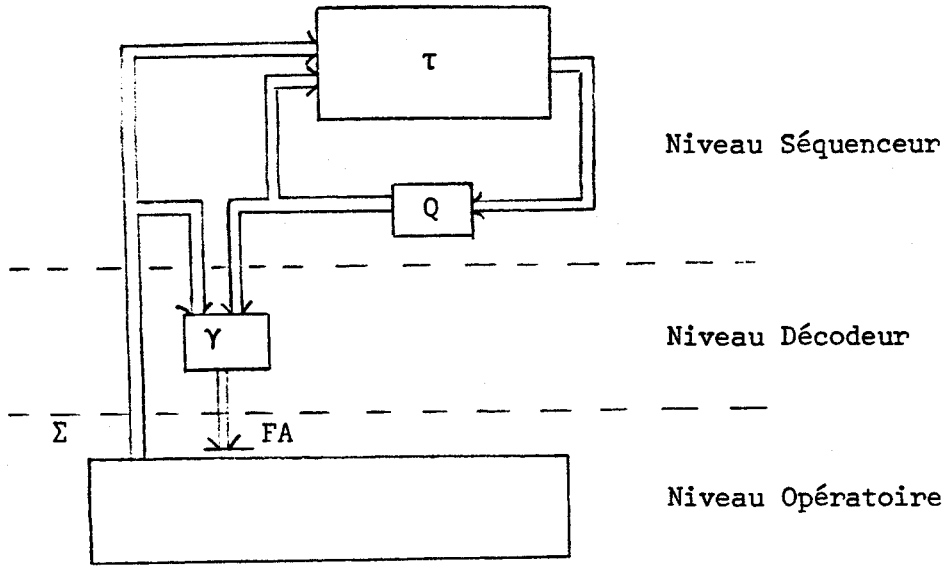


Figure B.2. : Aspect organisationnel des machines de MEALY.

IV.2. Machines séquentielles de MOORE

La fonction de décodage Υ, présentée dans les machines de MOORE, définit le flot d'activation FA, à partir de Q seulement.

L'aspect organisationnel de ces machines est montré dans la figure B.3.

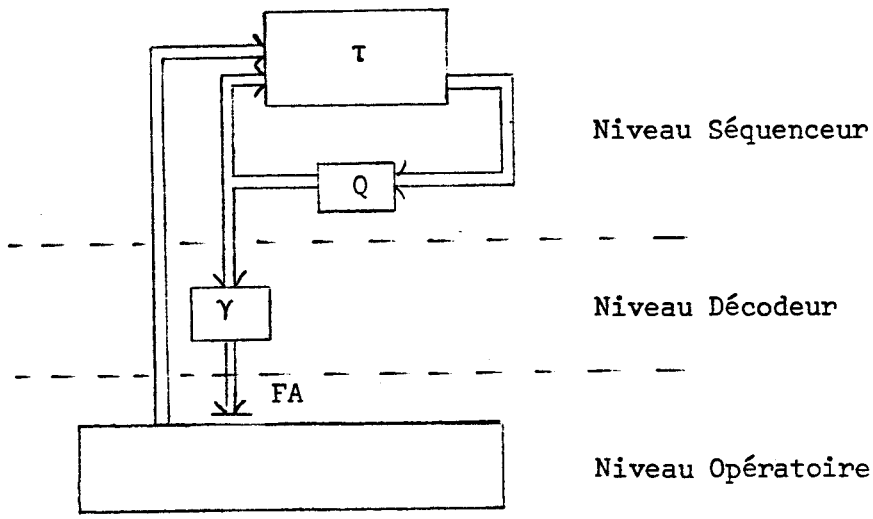
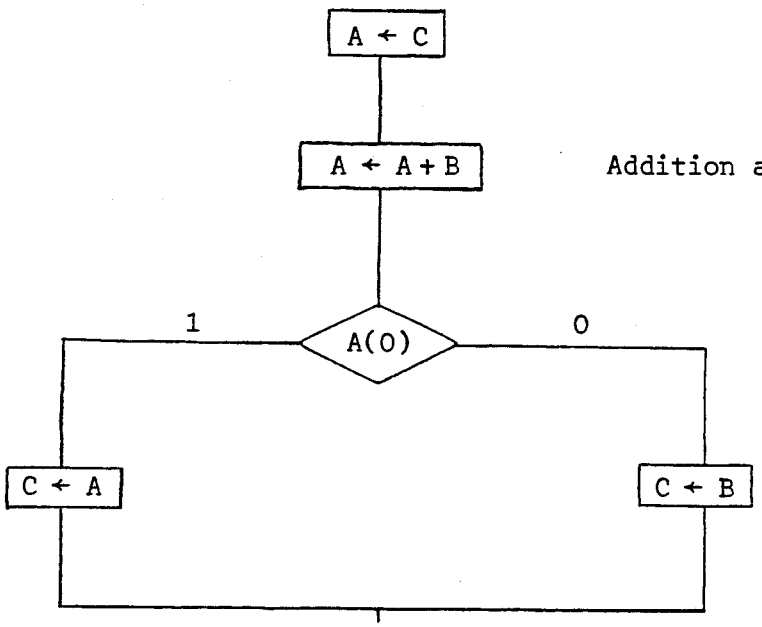


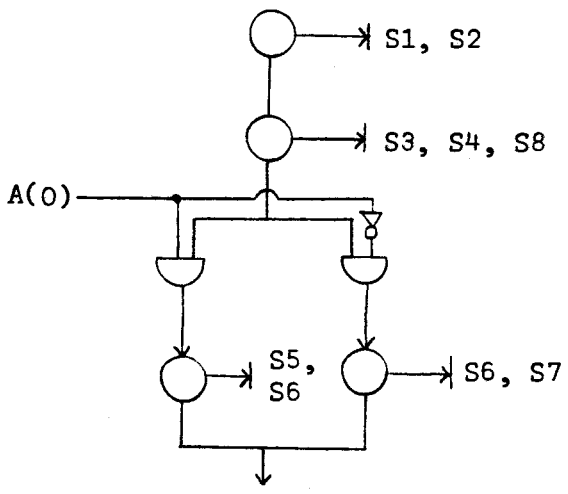
Figure B.3. : Aspect organisationnel des machines de MOORE.

IV.3. Exemple

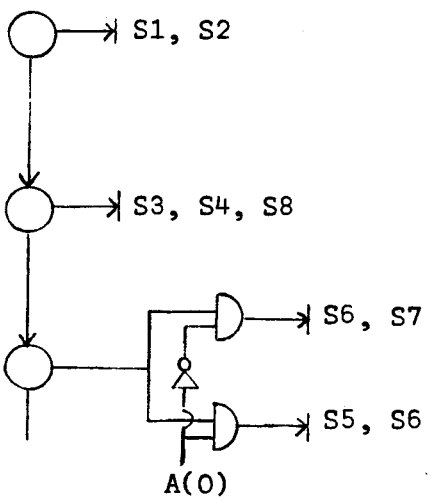
Pour illustrer ces deux familles de machines séquentielles, nous proposons un algorithme décrit sous forme d'organigramme et donnons sa représentation en forme de MEALY et MOORE ainsi que l'implémentation du niveau opératoire.



Description de l'algorithme sous forme d'organigramme.

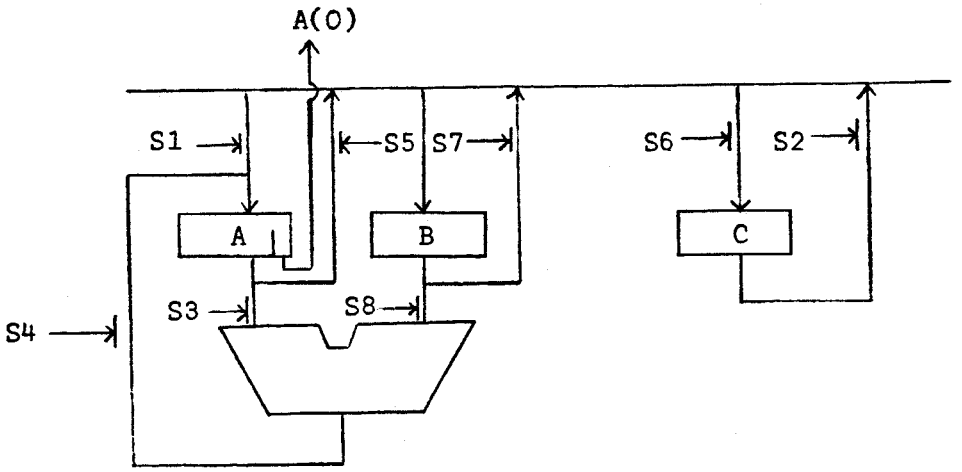


Automate de MOORE



Automate de MEALY

Le niveau opératoire sera défini ainsi :



Le niveau opératoire est composé d'éléments de mémorisation et d'une unité de traitement arithmétique et logique (UAL), cette unité peut être définie à l'aide d'équations booléennes utilisant l'algèbre de BOOLE comme moyen de définition.

Une représentation arborescente, d'un système logique, est schématisée ci-après. On donnera les différents outils utilisés pour la représentation de chaque entité.

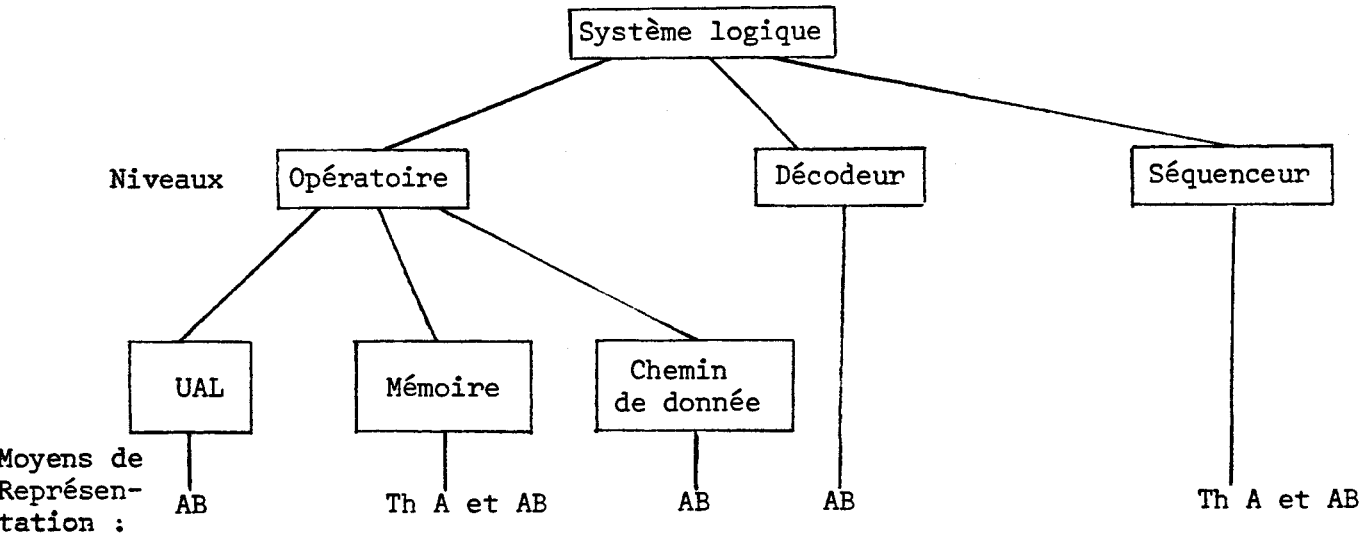


Figure B.4. : Moyens de représentation d'un système logique (AB : Algèbre de BOOLE, Th A : Théorie des Automates).

Un système logique, peut être, ainsi, défini à l'aide de l'algèbre de BOOLE et de la théorie des automates, mais ce n'est pas la façon la plus concise de les représenter et de les décrire [MERM], à cause du nombre élevé d'équations booléennes et de la représentation complexe d'un automate.

Un grand nombre de modèles ont été définis, dont la majorité est basée sur les concepts de schémas de programmes parallèles (SPP). Les SPP visent à associer à un programme donné sa structure de contrôle de base et d'éliminer les détails sur les opérations.

Il sera utile, pour la suite, de décrire les SPP, ainsi que d'autres outils de modélisation. Les descriptions seront accompagnées d'évaluations et de critiques.

V. SCHEMAS DE PROGRAMMES PARALLELES

Nous ne décrivons pas en détail ce modèle, d'autres l'ont très bien fait avant [PET], [TOU],...

Les SPP sont formés des deux niveaux : contrôle et opératoire (ou interprétatif) ; le premier est défini à l'aide de la théorie des automates, tandis que le second décrit le système logique au "niveau instruction" en définissant l'organe de mémorisation M et un ensemble d'instructions I.

Nous utilisons la même transcription schématique utilisée pour la théorie des automates pour présenter les SPP (figure B.5.).

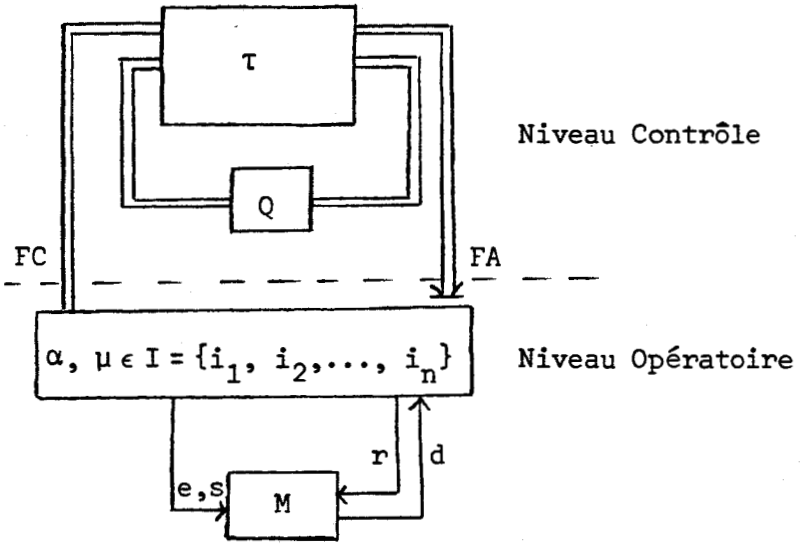


Figure B.5. : Transcription schématique de SPP
(FC : Flot de Conditions, FA : Flot d'Activation).

Le niveau contrôle est défini à l'aide d'un automate d'états finis.

Le niveau opératoire (ou interprétatif) est défini par le sexuplet $(I, \alpha, \mu, e, \sigma, M(r, d))$.

Où : - I est un ensemble fini d'instructions.
- α et μ sont deux instructions : initiale et finale.
- M(r, d) : mémoire contenant les données d et permettant de stocker les résultats r.
- e, s sont deux fonctions d'accès en mémoire pour effectuer respectivement une opération d'écriture et de lecture.

Plusieurs travaux ont exploré les concepts de SPP, pour effectuer l'analyse des fonctionnements des ordinateurs ; tels que les travaux de B. TOURSEL qui introduisent les schémas de machines [TOU].

En ce qui nous concerne, les SPP présentent des difficultés de représentation de systèmes logiques, car la fonction de transition τ reste purement combinatoire et ils font appel à la théorie des automates.

D'autres outils, faisant abstraction de τ, permettent des représentations plus concises ; parmi ces outils, on trouve les réseaux de Pétri, les réseaux de contrôle de processus parallèle...

VI. RESEAUX DE PETRI (RdP)

Les RdP sont des modèles formels des systèmes parallèles introduits par Pétri dans le cadre de la théorie générale des réseaux [PETR].

Les RdP sont utilisés comme moyen de représentation de systèmes logiques afin d'effectuer l'analyse ou la synthèse de ces derniers.

Un RdP est un graphe biparti $G = (P, T, \tau)$ muni d'une fonction M
où $P = \{p_0, p_1, \ldots, p_n\}$: ensemble fini de places.
$T = \{t_1, t_2, \ldots, t_m\}$: ensemble fini de transitions.
τ : ensemble d'arcs liant une place à une transition tel que :

$$\tau \subseteq (P \times T) \cup (T \times P)$$

M : fonction de marquage associant à chaque place son état de marquage dans $\mathbb{N}$, tel que $M : P \to \mathbb{N}^n$.

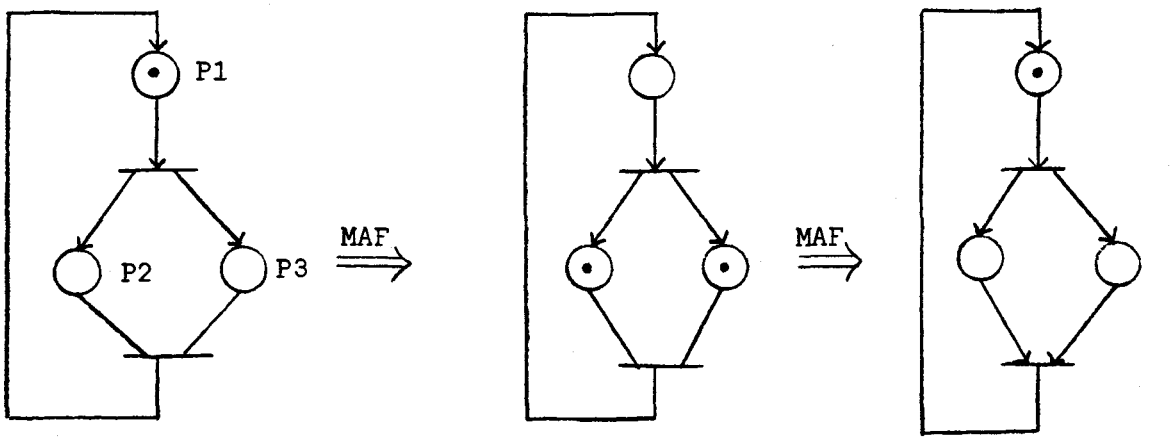
IV.1. Evolution des réseaux de Pétri

L'évolution d'un RdP au cours du temps est définie par la mise à feu des transitions t_j en modifiant l'état de marquage des places d'entrée des t_j de la façon suivante [GIR] :

Si toutes les places d'entrée d'une transition t_i contiennent des marques, alors t_i est dite déclenchable et on retranche une marque à toutes les places d'entrée en ajoutant une à chaque place de sortie de t_i.

Les places sont représentées graphiquement par des cercles (), les transitions par des tirés ().

Exemple d'évolution d'un réseau de Pétri :



$ec_1 = (1, 0, 0)$ $ec_2 = (0, 1, 1)$ $ec_1 = (1, 0, 0)$

L'état de contenu (ou de marquage) des places, (noté ec_i) change après chaque mise à feu (MAF). La structure du RdP est statique, elle implique les transitions de contenu d'un ec_i à un ec_{i+1} après chaque MAF.

Les RdP ont fait l'objet de plusieurs études, au niveau architecture système, telle que la représentation de processus parallèles pour étudier les problèmes de synchronisation... Leur usage, ici, est la modélisation du niveau contrôle d'un système logique.

Plusieurs extensions des RdP ont été définies [MOA], parmi lesquelles on trouve les réseaux de Pétri interprétés (RdPI) qui s'adaptent mieux à la modélisation de systèmes logiques.

VI.2. Réseaux de Pétri Interprétés (RdPI)

Les réseaux de Pétri Interprétés tiennent compte de deux notions :

- La notion temporelle : A chaque place est associée une durée δ_i de disponibilité d'une marque.

- La notion d'interprétation : - A chaque place est associée une opération, si cette place contient une marque alors l'opération qui lui est associée est activée.

  - A chaque transition est associée une condition, si cette condition est satisfaite et si les places d'entrée de la transition sont pleines, alors cette transition est déclenchable.

La transcription schématique d'un système logique, à l'aide des notions de RdPI, est montrée dans la figure B.6.

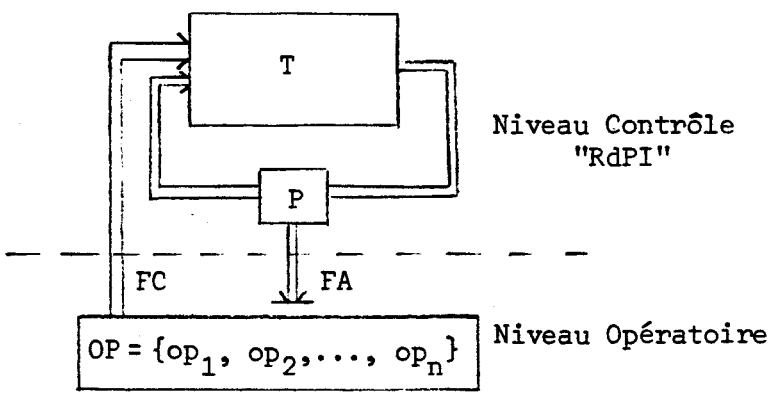


Figure B.6. : Transcription schématique d'un système logique à l'aide de RdPI.

Les RdPI permettent de représenter, de manière graphique, l'exécution parallèle de deux opérations ou plus ; car à chaque instant, deux places ou plus peuvent contenir des marques et permettent ainsi l'activation de plusieurs opérations, indépendamment l'une de l'autre.

Ils permettent aussi la détection des conflits entre les opérations qui se partagent les mêmes ressources, telles que : registres, chemins de données...

Mais ce formalisme peut devenir complexe par la traduction de certains mécanismes de synchronisation [ZAC] ; de même, il ne présente pas de caractère hiérarchique du niveau de contrôle. Ce caractère a été largement discuté au premier chapitre.

C'est ainsi que les Réseaux de Contrôle des Processus Parallèles (RCPP) ont été développés pour répondre à ces besoins.

VII. RESEAUX DE CONTROLE DES PROCESSUS PARALLELES [ZAC]

Un Réseau de Contrôle des Processus Parallèles (RCPP) est défini par la donnée d'un quintuplet (X, P, Q, f, P_0) avec :

$X = \{x_1, x_2, \ldots, x_n\}$ un ensemble fini de variables booléennes appelées variables d'entrée.

$P = \{p_1, \ldots, p_m\}$ un ensemble fini d'objets appelés phases.

$Q = \{q_1, \ldots, q_m\}$ un ensemble fini de variables booléennes en bijection avec les phases.

f : application de $P \times P \rightarrow F(Q, X)$; où $F(Q, X)$ est l'ensemble des fonctions booléennes simples de Q et de X telles que $f(p, p) = 0$.

P_0 : un ensemble de phases initiales contenu dans P ($P_0 \subseteq P$).

A un RCPP peut être associé, de façon biunivoque, un réseau dont les noeuds s'identifient avec les phases et où l'arc (p_i, p_j) porte la fonction $f(p_i, p_j)$, qui est désignée par f_{ij}.

L'évolution d'un RCPP consiste à n'effectuer la transition d'une marque, d'une phase p_i à une phase p_j, que si la fonction booléenne $f_{ij} = 1$, comme le montre la figure B.7.

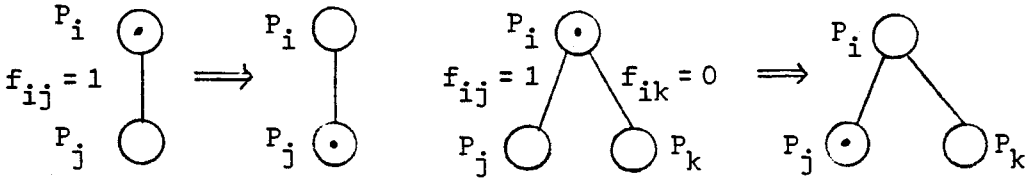


Figure B.7. : Evolution de RCPP.

La transcription schématique d'un système logique, à l'aide des notions de RCPP, est montrée dans la figure B.8.

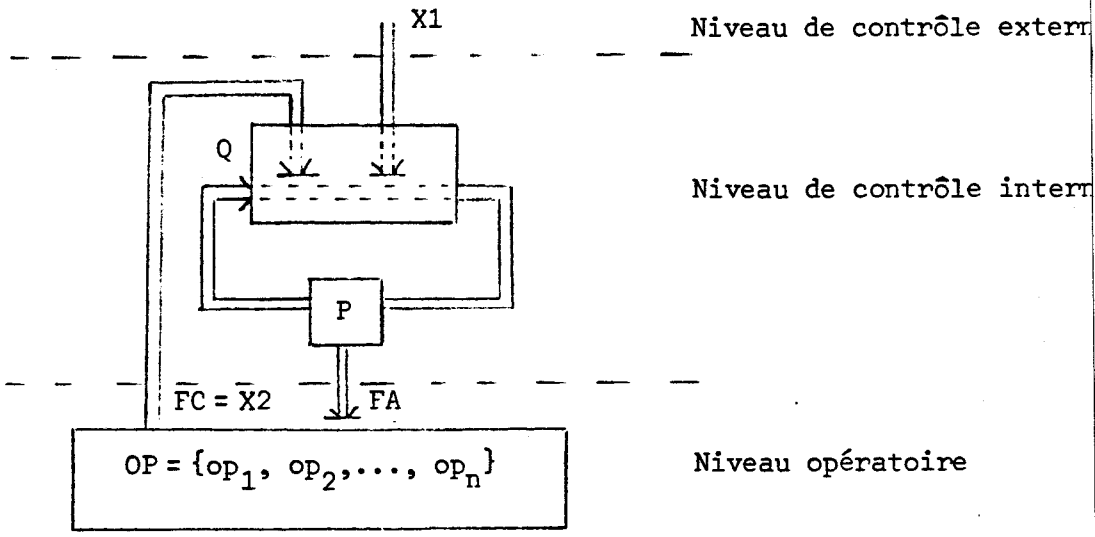
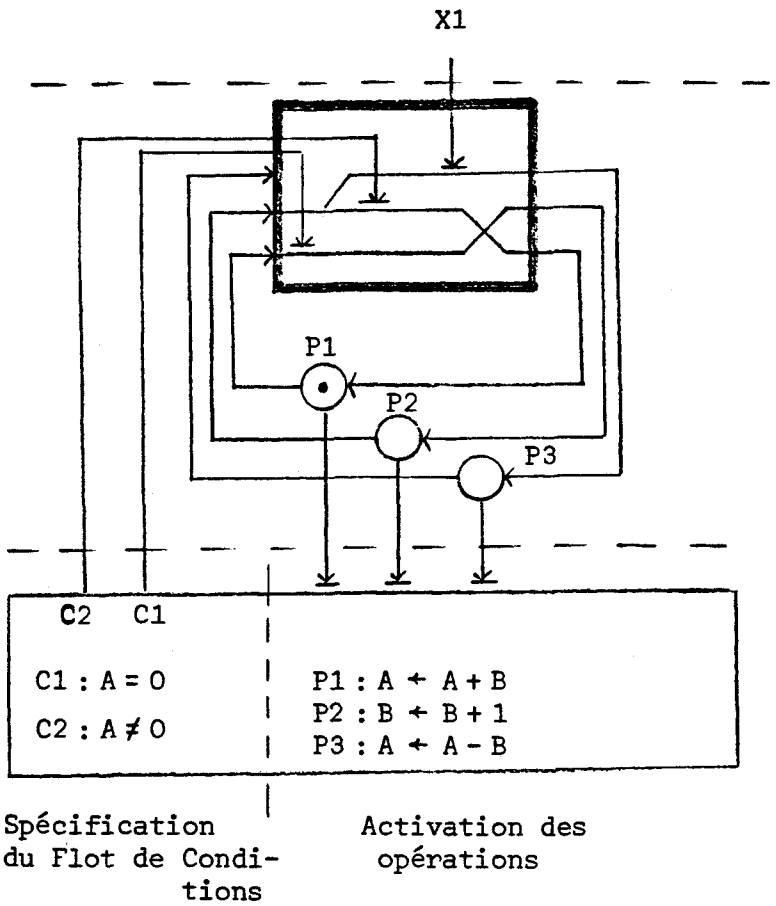
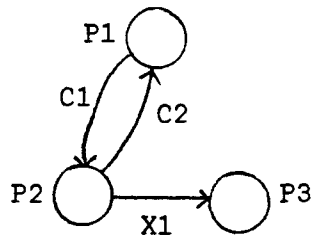


Figure B.8. : Transcription schématique d'un système logique à l'aide de RCPP (avec $X = X1 \cup X2$).

Un exemple d'illustration est exposé, dans la figure B.9, montrant clairement :

- la séparation des niveaux contrôle et opératoire,
- l'interaction des niveaux,
- les hiérarchies de contrôle d'un système logique.





Automate de contrôle correspondant.

Figure B.9. : Représentation d'un système logique à l'aide de RCPP.

L'organisation hiérarchique, ainsi définie, n'est pas toujours facile à établir. D'autres modèles ont été développés représentant d'une manière homogène les données et les commandes, en tenant compte de l'aspect modulaire des systèmes logiques.

Nous présentons, ici, un modèle fonctionnel de description de systèmes logiques et de flots d'informations, nommé MODFLO.

VIII. MODFLO : Modèle fonctionnel [ALA]

MODFLO est un modèle fonctionnel de description des systèmes logiques, développé à l'Université du Languedoc ; il est utilisé comme outil de simulation et de conception des systèmes logiques.

Contrairement aux autres modèles, MODFLO permet une représentation homogène des données et des commandes par des flots et une définition structurelle par des entités primitives fonctionnelles.

Un attribut de subjectivité associé aux flots est défini, afin d'interpréter le comportement dynamique d'un système en le partitionnant en parties actives et inactives (intéressantes et non-intéressantes).

VIII.1. Flot d'information

Un flot F est défini par un ensemble de paramètres et un ensemble d'attributs

$F = \{b, \theta, c, d\}$, Is1, Is2, $(a_1, \ldots, a_n)$, où :

b : le contenu du flot.
θ : l'instant de création du flot.
c : la capacité du flot (exprimée en nombre de bits).
d : la durée d'existence du flot.
Is1 : le premier attribut de subjectivité du flot, caractérise la nature du flot, Is1 = I si le flot est intéressant.
= NI si le flot est non-intéressant.
= INI si le flot est formé de deux flots élémentaires I et NI.
Is2 : le deuxième attribut de subjectivité du flot est défini si et seulement si Is1 = I. Is2 ∈ {In, Iinh, Ic, Iv, Inv, Itv, Itnv}, où :
In : est l'attribut d'un flot intéressant de *données* normales.
Iinh : est l'attribut d'un flot intéressant de *données* qui est la sortie d'une entité primitive fonctionnelle inhibée.

Ic : est l'attribut d'un flot intéressant de *données* ayant un contenu b constant.

Iv(Inv) : est l'attribut d'un flot intéressant de *commande* qui est validé (non validé). Ce flot est la sortie d'une entité de décision (1-N).

Itv(Itnv) : est l'attribut d'un flot intéressant de *commande* qui est validé (non validé). Ce flot est la sortie d'une entité de décision (K-N).

VIII.2. Entités primitives fonctionnelles (EPF)

Les EPF ne sont pas des entités matérielles ; mais peuvent correspondre d'une manière bijective, ou non, à des éléments matériels réels.

Deux classes d'EPF ont été distinguées.

VIII.2.1. Les EPF non opératives

Les EPF non opératives agissent uniquement sur la capacité (c) des flots Les différentes entités de cette classe sont les suivantes :

- EPF de passage "PASS".
- EPF d'éclatement "ECL".
- EPF de divergence "DIV".
- EPF de fusionnement "FUS".
- EPF de flot source "SRC".

Ces EPF non opératives sont représentées graphiquement par la figure B.9.

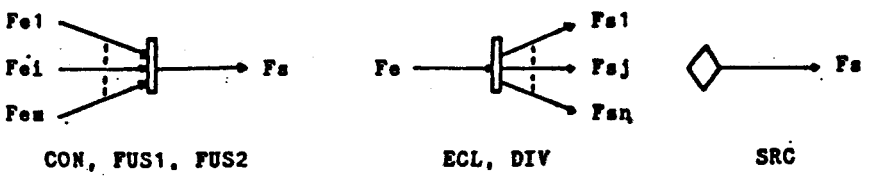


Figure B.9. : EPF non opératives.

VIII.2.2. Les EPF opératives

Cette classe comprend des EPF ayant un rôle opératif (transformation, transfert, prise de décision, etc...). Ces entités peuvent être commandées, ou non. Les différentes entités de cette classe sont les suivantes :

- EPF de passage "PASS".
- EPF de transformation commandée "TRNS COM".
- EPF de transformation non commandée "TRNS NON COM".
- EPF de décision à une entrée et n sorties mutuellement exclusives "DEC1N".
- EPF de décision à m entrées et n sorties "DECMN".
- EPF d'écriture-mémoire "ECR".
- EPF d'écriture-mémoire "LEC".
- EPF de mémorisation "MEM".
- EPF d'horloge "HOR".

Ces EPF opératives sont représentées graphiquement par la figure B.10.

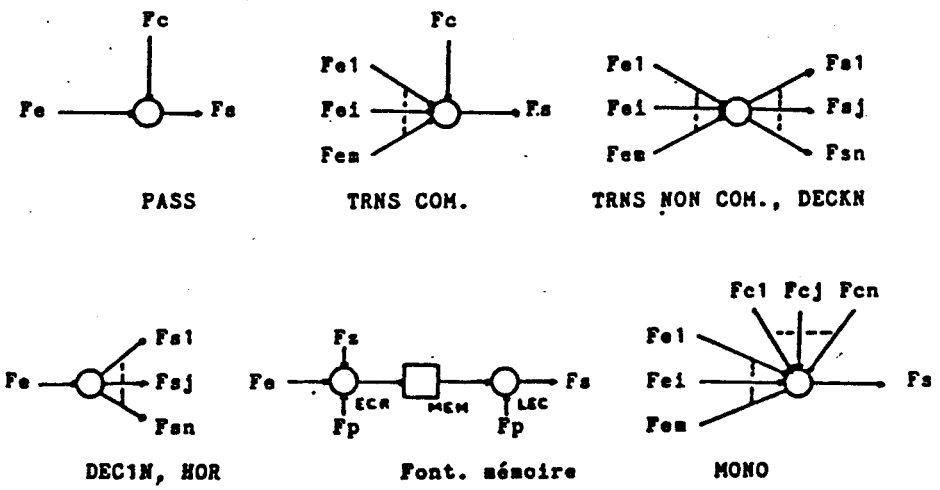


Figure B.10. : EPF opératives

Aux EPF opératives est associé un ensemble d'attributs temporels, qui sera pris en compte pour la génération de flots de sorties.

VIII.3. Exemples de modélisation à l'aide de MODFLO

Pour illustrer ce modèle, deux exemples sont présentés :

- Multiplexeur à quatre entrées. (figure B.11.a.).

- Registre à décalage, permettant d'effectuer le chargement parallèle et le décalage à droite. (figure B.11.b.).

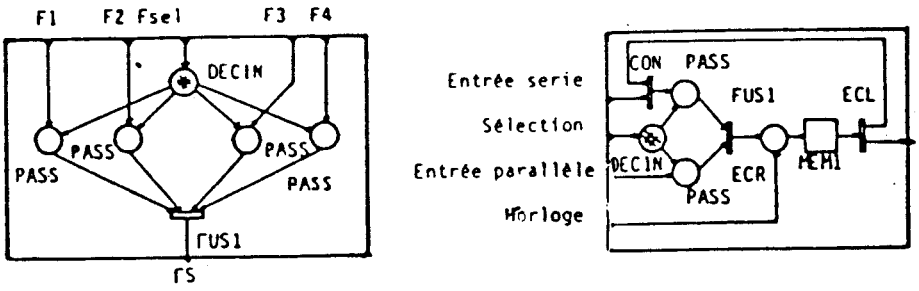


Figure B.11. : Modélisation d'un multiplexeur (a) et d'un registre à décalage (b).

Il est clair que la description d'un système logique, à l'aide de MODFLO, n'est pas structurée ; ce qui pourrait entraîner des difficultés chez l'utilisateur.

VIII.4. Propositions

Pour faciliter la description d'un système logique, à l'aide de MODFLO, il serait utile de décomposer le système en plusieurs niveaux fonctionnellement distincts.

Nous proposons une décomposition méthodique, sans mettre en cause les principes du modèle MODFLO. Les niveaux fonctionnels de cette décomposition sont les suivants :

- Niveau 4 : niveau de génération des flots de l'horloge.
- Niveau 3 : niveau de génération des flots de sélection.
- Niveau 2 : niveau de génération des flots de sélection effective ou "décodée".
- Niveau 1 : niveau de traitement des flots sélectionnés.
- Niveau 0 : niveau de mémorisation.

Un système logique peut ne pas comporter les cinq niveaux cités, comme le montre l'exemple du multiplexeur, où le niveau 4 n'est pas présenté.

Nous donnons, ci-après, la décomposition fonctionnelle, du multiplexeur et du registre à décalage, où la représentation est plus claire.

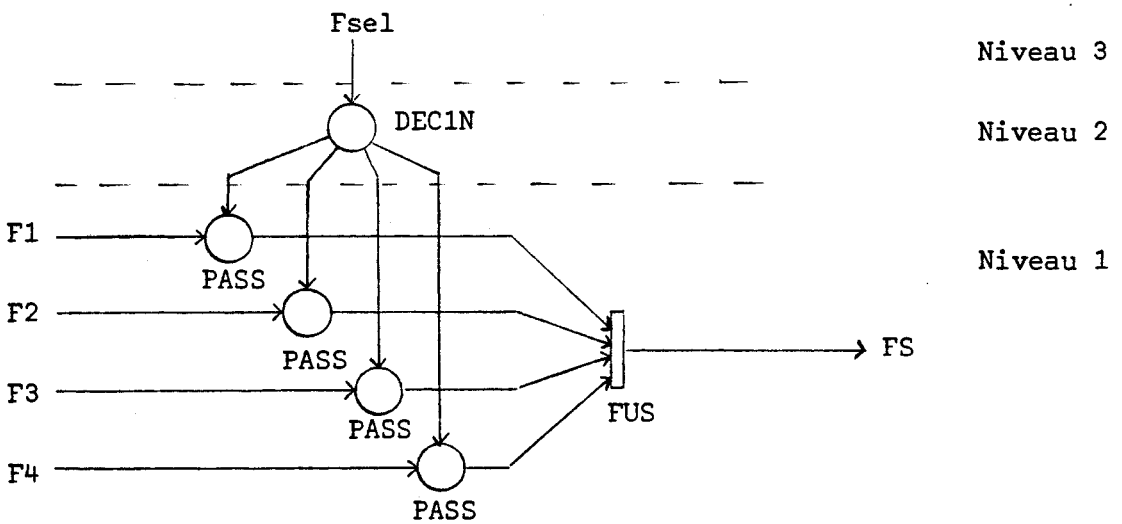


Figure B.12. : Décomposition fonctionnelle du multiplexeur.

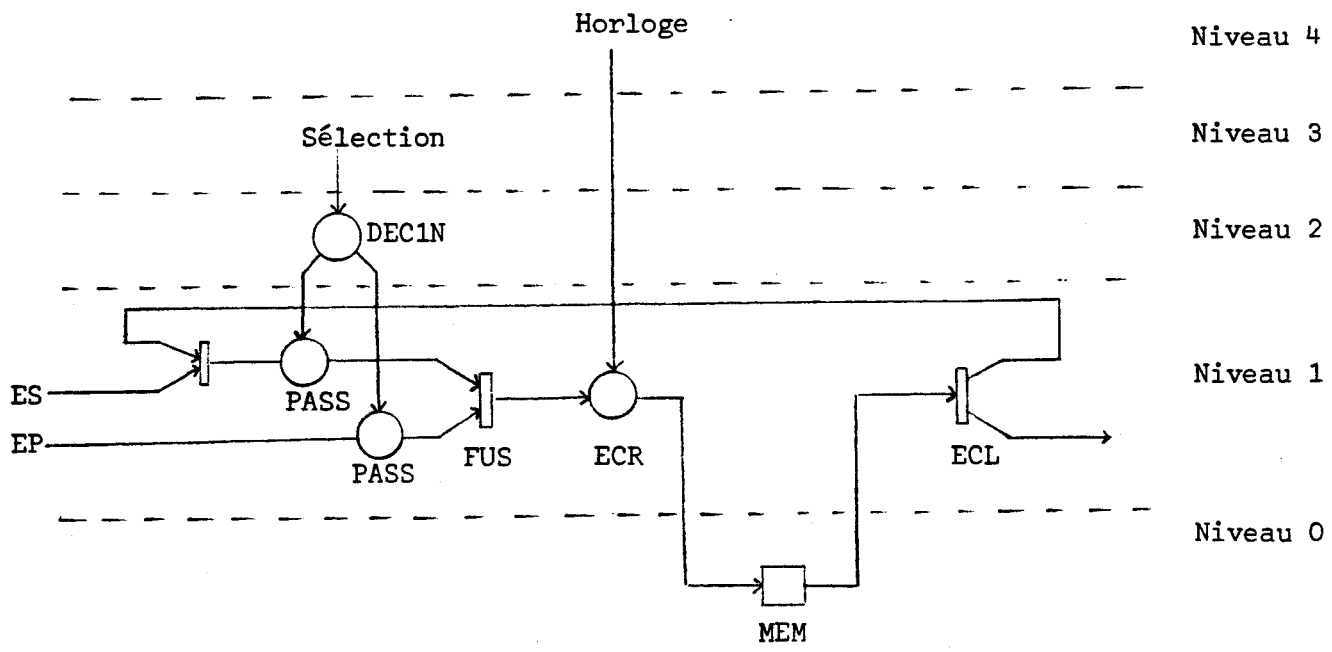


Figure B.13. : Décomposition fonctionnelle du registre à décalage.

Les flots de sélection, pour ces deux exemples, sont générés de l'extérieur du système ; il peuvent être générés de l'intérieur dans le cas des systèmes présentant un mécanisme de séquencement interne.

La décomposition fonctionnelle présentée, ici, sera plus détaillée dans les chapitres qui suivent.

Mis à part MODFLO, qui utilise des primitives d'une manière homogène dans le niveau commande et dans le niveau donnée, les autres modèles présentés permettent la description et l'analyse du flot de contrôle.

IX. CONCLUSION

Nous avons présenté, dans ce chapitre, les concepts et les éléments constitutifs de quelques outils de modélisation des systèmes logiques.

Il a été rappelé pour chacun de ces outils l'intérêt de la hiérarchisation en niveaux afin de rendre à la fois, simple et claire, la représentation des systèmes logiques.

Il a été constaté qu'à chacun de ces niveaux est associée une fonction de contrôle ou de données, ces niveaux doivent tenir compte des aspects structurels et comportementaux des objets matériels.

Cela nous a servi de notion de base pour définir un nouvel outil de modélisation, appelé réseau multi-niveaux qui tiendra compte des deux aspects suivants :

- Décomposition fonctionnelle du système logique en plusieurs niveaux.

- Hiérarchie structurelle des intéractions des niveaux.

Dans ce chapitre nous n'avons pas tenu compte des niveaux d'abstraction, ou de description, des systèmes logiques car ce qui a été visé, ici, était l'évaluation du caractère hiérarchique dans les outils de modélisation.

Nous introduisons dans le chapitre suivant une manière textuelle de décrire les systèmes logiques à l'aide de langages de description.

CHAPITRE III

LANGAGES DE DESCRIPTION DE SYSTEMES LOGIQUES

PLAN

I. INTRODUCTION

Le texte et l'image sont deux moyens, de communication "Homme-Homme" et "Homme-Machine", clairs et naturels.

Les descriptions textuelles et graphiques sont devenues utiles, pour clarifier les représentations de systèmes logiques, à l'aide de langages textuels et graphiques, dont la syntaxe et la sémantique doivent être plus proches de la réalité matérielle.

Y. CHU, introduisant l'un des premiers langages textuels, a donné une analogie entre les langages de description de systèmes logiques et le symbolisme mathématique. Cette analogie est illustrée dans la figure C.1.

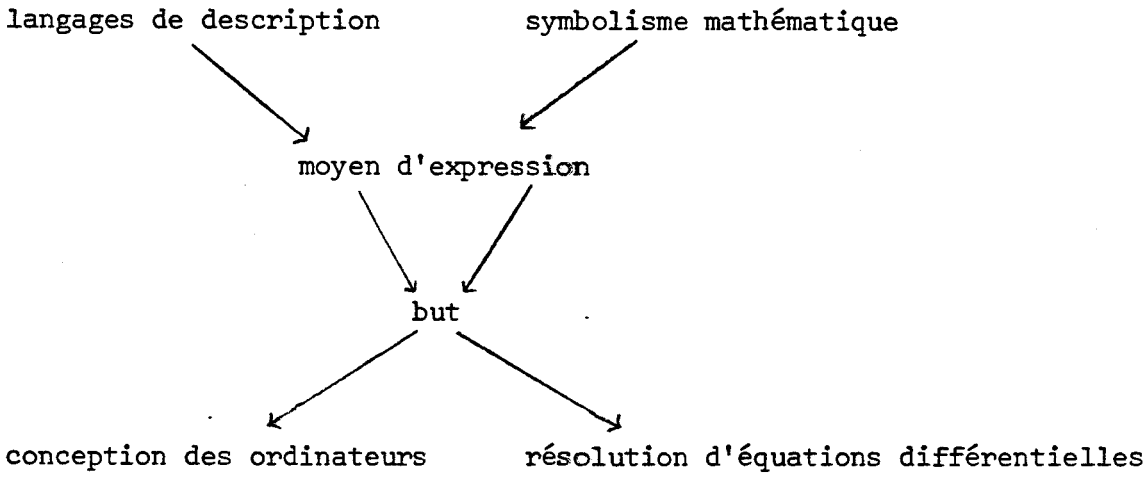


Figure C.1. : Analogie [CHU].

Le premier paragraphe de ce chapitre est consacré à la présentation des domaines d'application de ces langages, deux classes d'application seront distinguées, la première correspond aux communications homme-homme et la seconde aux communications homme-machine ; pour chacune de ces deux classes nous montrerons quelques domaines précis d'application.

Le second paragraphe introduira les différents niveaux de description, appelés aussi niveaux d'observation, allant du niveau le plus abstrait qu'est le niveau architecture système jusqu'au plus détaillé qu'est le niveau électrique.

Au troisième paragraphe, nous présenterons trois modes de description, parmi lesquels on trouve le mode semi-structurel introduit pour faciliter le passage entre les modes de description. Pour chacun des modes ; nous montrerons, la manière utilisée pour décrire un système logique.

Le quatrième paragraphe abordera le rôle d'un langage de description de systèmes logiques dans un système intégré d'aide à la conception de systèmes VLSI.

Ensuite, le métalangage CONLAN sera présenté ; celui-ci comporte une famille de langages destinés à la description de systèmes logiques à différents niveaux de description.

Enfin, nous présenterons l'application de quelques outils de modélisation, cités dans le chapitre précédent, aux niveaux de description.

On ne s'intéressera, ici, qu'aux langages textuels.

II. DOMAINES D'APPLICATION

Le rôle principal d'un langage de description de systèmes logiques est de présenter un moyen de communication facilitant le transfert de connaissance et d'échange d'informations concernant un système logique donné.

Deux types d'application sont distingués, l'un visant les communications homme-homme et l'autre les communications homme-machine.

II.1. Communication homme-homme

Les langages de description sont utilisés pour un but d'interfaçage entre deux individus ou plusieurs, comme le montrent les cas suivants :

- Concepteur/utilisateur : Le langage est utilisé comme moyen de rédaction d'un rapport de documentation du système conçu.

- Concepteur/concepteur : Le langage est utilisé comme moyen de communication entre plusieurs concepteurs travaillant sur un projet global.

- Enseignant/étudiant : Le langage est utilisé comme support d'un cours de structure et fonctionnement des ordinateurs.

II.2. Communication homme-machine

La machine est destinée à effectuer les opérations suivantes :

- La simulation du système logique décrit à l'aide du langage.
- Son test.
- Sa conception.

Le langage de description a un rôle d'interface entre l'utilisateur (concepteur, étudiant,...) et la machine.

III. NIVEAUX DE DESCRIPTION

Les langages de description de systèmes logiques sont destinés principalement à des fins de simulation et de conception de systèmes logiques.

Pour chacun de ces deux domaines d'utilisation on distinguera plusieurs niveaux de description.

En effet, pour les langages destinés à la conception quatre niveaux de description sont distingués ainsi que six pour ceux destinés à la simulation.

III.1. Niveaux destinés à la conception

Pour les langages destinés à la conception, quatre niveaux de description sont distingués [AMB] :

1. Niveau architectural : le système manipule des informations, du type : nombres, instructions d'un processeur, adresse,...

2. Niveau logique : Le système est décrit à l'aide d'équations booléennes et de la théorie des automates. On s'intéresse ici au codage des informations utilisées au niveau architectural.

3. Niveau électrique : Le système traite des informations de nature purement électrique, telles que les tensions et les intensités.

4. Niveau implanté : La description en ce niveau est purement géométrique, elle se fait au moyen de schémas normalisés dit schémas des masques.

III.2. Niveaux destinés à la simulation

Pour les langages destinés à la simulation, six niveaux de description sont distingués [BOR], allant du niveau de description le plus global qu'est le niveau architecture système, jusqu'au plus détaillé qu'est le niveau électrique.

1. Le niveau architecture système, décompose le système logique en plusieurs processus communiquant ou non entre-eux ; les résultats de simulation à ce niveau, doivent permettre la détection des problèmes liés au système logique à ce niveau, tels que : les problèmes de synchronisation, d'interaction entre les processus, ainsi que les cas de famine, d'interblocage.

2. Le Niveau architecture matérielle, appelé aussi PMS (Processor Memory Switch) où le système est décrit en terme de ressources matérielles, telles que, les processeurs, les unités mémoires et le réseaux de connexions entre ces ressources.

3. Niveau instruction où toute instruction, du système, et ses règles d'interprétation sont spécifiées.

4. Niveau transfert de registre : Le système est décrit en terme d'éléments de mémorisation (latch, registre, mémoire), l'exécution des opérations élémentaires entre ces éléments est spécifiée par des règles de transfert.

5. Niveau logique : La structure du système est définie par des connexions entre les portes logiques que comporte le système.

6. Niveau électrique : Le système est décrit en terme d'éléments électriques, à savoir les transistors, les diodes, les résistances,... ainsi que leurs liaisons.

Le tableau C.2. donne un aperçu sur les éléments de composition de système logique, les résultats de simulation et des exemples de langages pour chacun des six niveaux de description.

Niveau de description	Exemples de langages	primitives	Résultat de simulation
Architecture système	LOGOS ADLIB SARA...	processus	Détection des problèmes d'interaction et de synchronisation entre les processus, interblocage, famine,...
Architecture matérielle	PMS ADL SLIDE...	mémoires, processeurs, périphériques,	Evaluation de performances.
Instruction	ISP LCD CAP...	modules matériels protocole de communication	- Détection de conflits d'accès aux ressources. - Evaluation de performances fines.
Transfert de registre	CDL DSM AHPL NEGLAN...	éléments de mémorisation	Analyse comportementale.
Circuit combinatoire	SPLICE LOGSIM TEGAS...	bascules portes logiques	Détection de pannes.
Circuit électrique	SPICE	transistors résistances...	- Simulation de phénomènes continus (I, V). - Validation des paramètres électriques.

Tableau C.2. : Niveaux de description.

IV. MODES DE DESCRIPTION

A chaque niveau, le système est décrit en terme de primitives associées à ce niveau. Cette description est caractérisée par un mode de description.

Deux modes de description sont principalement distingués :

- Description comportementale.
- Description structurelle.

Ces deux modes peuvent être utilisés pour une seule description appelée "description mixte" [JAH].

Nous allons définir un autre mode de description, la description semi-structurelle qui sera justifiée dans les chapitre IV et V.

IV.1. Description structurelle

Le système est décrit en terme de primitives structurelles (éléments matériels ou sous-système) munies de connexions entre-elles.

Ce mode est utilisé, par exemple au niveau électrique, pour spécifier les connexions entre les différentes primitives électriques que composent le système.

IV.2. Description semi-structurelle

Le système est décrit en terme de primitives structurelles, sans spécifier les connexions structurelles entre-elles.

Ce mode est utilisé, par exemple au niveau transfert de registre, pour déclarer les différents registres utilisés dans la description comportementale.

IV.3. Description comportementale (ou algorithmique) [ETI]

Tout système logique a pour rôle l'exécution d'un certain processus, il peut être comportementalement assimilé à ce processus.

Pour ce mode de description, le système est décrit à l'aide de règles d'évolution au cours du temps, sans tenir compte de sa réalisation matérielle.

La description comportementale d'un système, implique sa structure complète, c'est-à-dire les différentes connexions entre les primitives.

Les deux derniers modes sont illustrés dans l'encadré suivant, utilisant "CDL" comme langage de description [CHU].

```
DECLARE
MEMORY [0 : 1023] <0 : 15>
ACCUMULATOR <0 : 16>
PC <0 : 11>
IR <0 : 15>
```

Description semi-structurelle.

```
IR := MEMORY [PC]
IF IR <8 : 11> = 10 THEN... etc
```

Description comportementale.

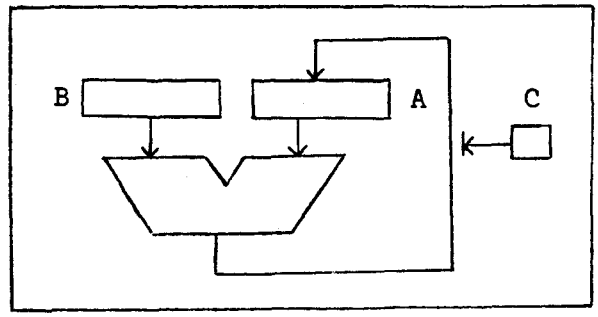
Encadré C.3.

IV.4. Transformation de modes de description

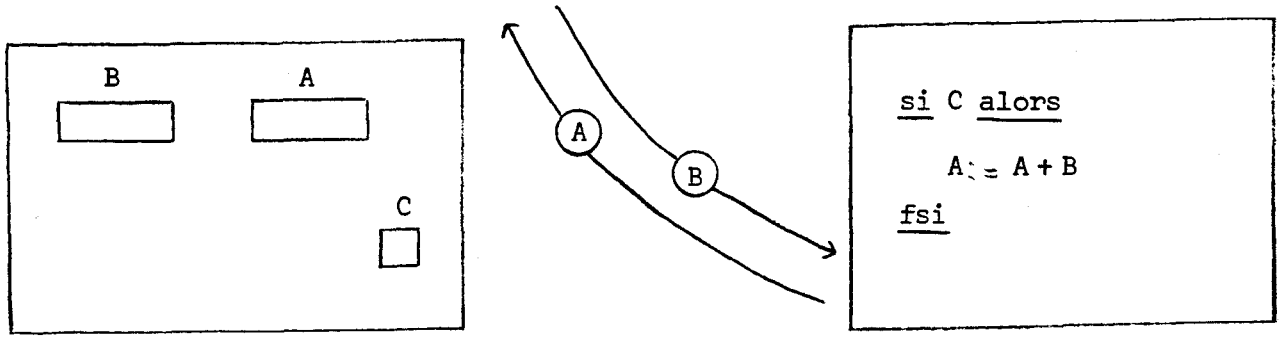
Il s'agit de passer d'une description d'un système logique en un mode de description à une autre description en un autre mode.

Le passage d'un mode à un autre vise, soit l'analyse du fonctionnement du système, soit sa conception.

Deux sens de passage entre les modes de description sont alors distingués et montrés dans les encadrés C.3. bis.



Description structurelle



Description semi-structurelle

Description comportementale

(A) : Transformation pour la conception.

(B) : Transformation pour l'analyse fonctionnelle.

Encadrés C.3.bis. : Transformation de modes de description.

Nous exposerons dans ce qui va suivre un système intégré d'aide à la conception de système logique.

V. SYSTEME INTEGRE D'AIDE A LA CONCEPTION DE SYSTEMES VLSI

Les processus de simulation et de conception d'un système logique, décrit à l'aide d'un langage de description à un niveau de description donné, nécessitent la disposition d'outils logiciels permettant :

- La simulation du système logique à différents niveaux de description.
- La génération de test.
- Le dessin de masques.

Un système intégré, d'aide à la conception de systèmes logiques à très grande échelle d'intégration (VLSI), regroupe l'ensemble de ces outils logiciels comme le montre la figure C.4.

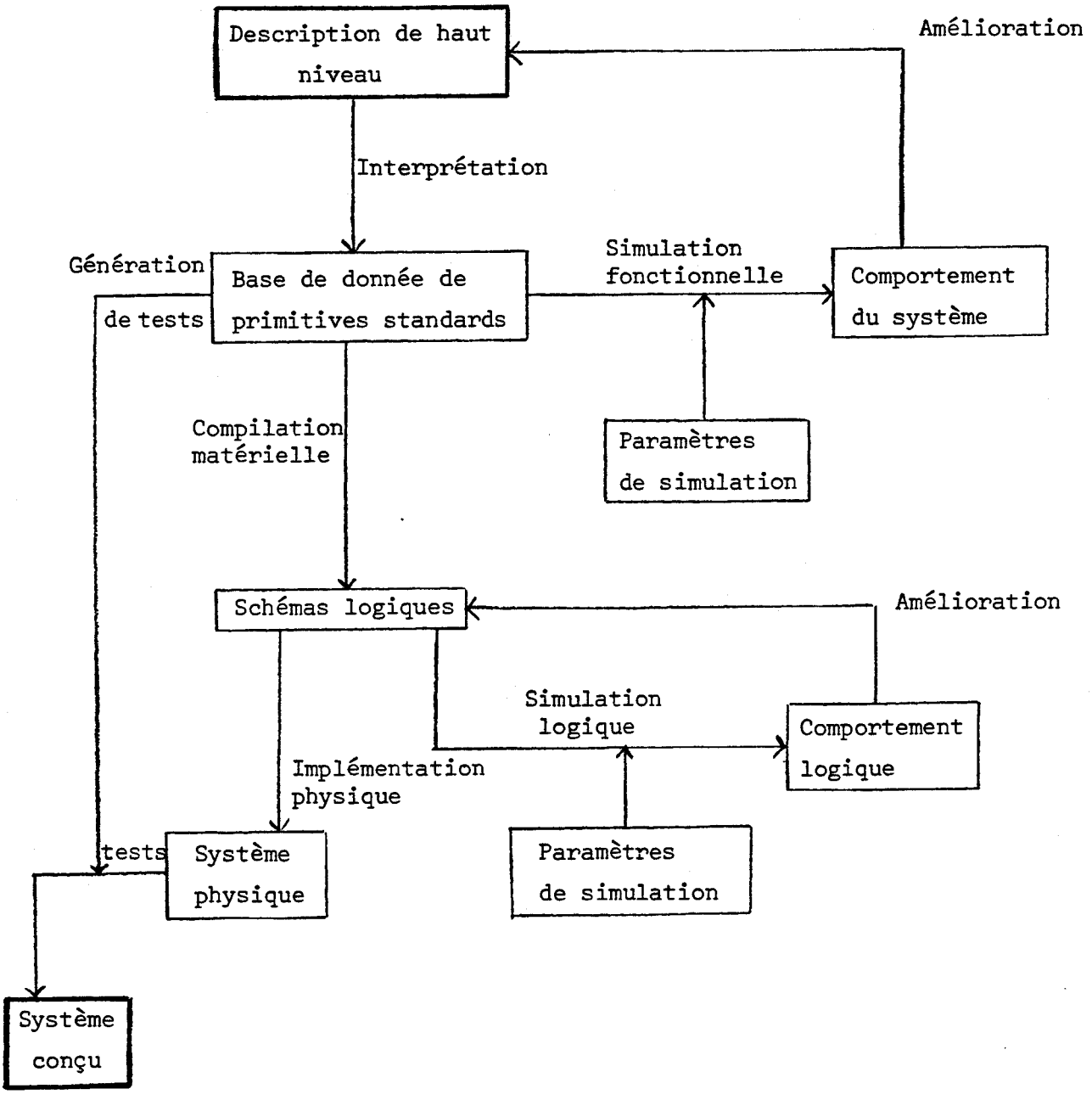


Figure C.4. : Structure globale d'un système intégré d'aide à la conception de systèmes VLSI [SHI].

Durant les quatre dernières années plusieurs systèmes intégrés, pour la CAO de VLSI, ont été développés : EPISODE [CHI], OASIS [MIL], CASSIOPE [LEC], etc...

Chacun de ces systèmes utilise des notations pour la description de haut niveau, ces notations correspondent à celles d'un langage de description. Des réunions internationales sur ce thème ont montré la nécessité :

- d'unification des notations,
- de définition d'un système de description à différents niveaux de description.

VI. CONLAN : LANGAGE MULTI-NIVEAUX

Depuis le début des années soixante dix, le nombre des langages de description de systèmes logiques n'a pas cessé d'augmenter. Chacun défini son propre langage, utilise son propre symbolisme pour simuler sa propre machine.

Cette multitude de langages de description (Tour de Babel) [LIP] a mené à une définition unifiée de notations et de concepts de ces langages. C'est ainsi que le projet international CONLAN (CONsensus LANgage) prit naissance.

VI.1. But de CONLAN

Le but visé initialement par LIPOVSKI, l'initiateur du projet, était de définir un langage multi-niveaux (de description). Devant la nécessité de procéder au préalable à une réflexion théorique, ce but a été dévié. CONLAN est devenu une base syntaxique et sémantique accompagnée de définition formelle et d'un modèle d'interprétation [BOR].

CONLAN regroupe une famille de langages de description, chacun est destiné à un niveau de description donné, réalisant une intégration verticale (paragraphe suivant) :

1. Utilisable à tous les niveaux de description.
2. Offrant une méthode uniforme de description.
3. Permettant des traitements sur une description multi-niveaux.

La figure C.5. montre les différents langages utilisés pour la description de systèmes logiques à différents niveaux de description, ainsi que les outils de simulation, de test, de dessin de circuits imprimés, de dessin de masques,... etc..., dans le système CONLAN [MER].

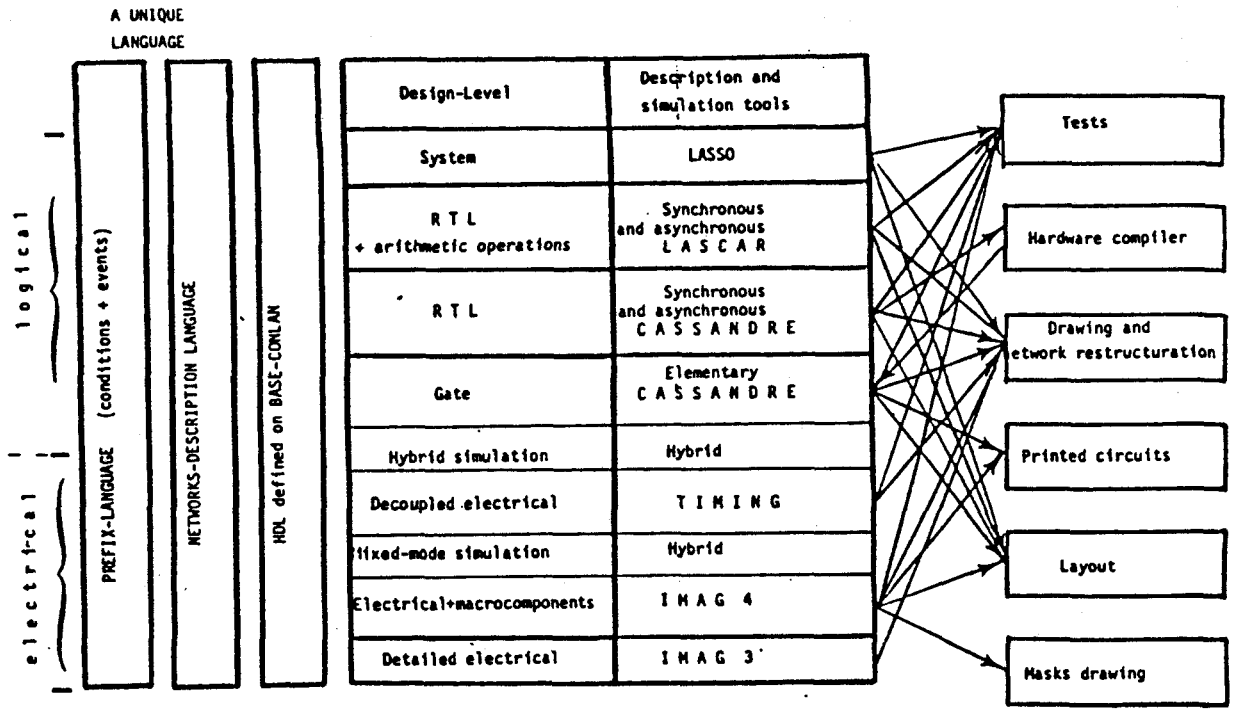


Figure C.5. : Système CONLAN.

VI.2. Migration verticale

La migration verticale consiste à passer d'une description à un niveau donné à son équivalente à un autre niveau (subordonné pour permettre la conception).

Ce passage de représentation correspond à l'interprétation des primitives associées à chaque niveau ainsi que leurs interconnexions.

A tous les niveaux doivent être associés des outils de vérification, de simulation des descriptions, pour permettre des corrections et des modifications afin de passer à un niveau plus détaillé, jusqu'au niveau électrique qui correspondra à la génération de masques.

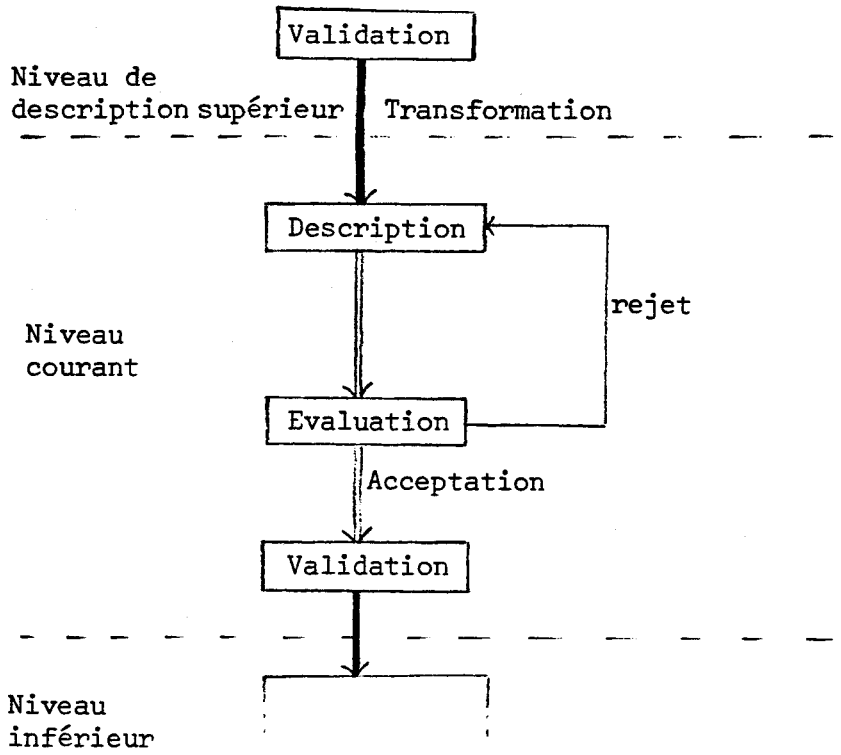


Figure C.6. : Migration verticale [ETI].

VII. FORMALISMES ET NIVEAUX DE DESCRIPTION

Le chapitre B a été consacré à la définition de quelques outils de modélisation comportementale de systèmes logiques.

Chacun de ces outils ainsi que d'autres formalismes sont classiquement utilisés pour décrire un système logique à un niveau de description donné.

Parmi ces outils, il existe ceux qui permettent des descriptions à deux ou trois niveaux. Nous donnons l'exemple des RdP qui sont utilisés aux trois niveaux de description suivants :

- Niveau architecture système [GIR].
- Niveau architecture matérielle [KRY].
- Niveau transfert de registre [PAW].

Nous montrons, dans la figure C.7, quelques formalismes utilisés pour chaque niveau de description.

Niveau de description	Formalismes.
Architecture système	RdP [GIR].
Architecture matérielle	RdP [KRY].
Instruction	Schéma de programmes parallèles.
RTL*	Théorie des automates, RdP.
Logique	Algèbre de Boole. Théorie de commutation.
Electrique	Théorie générale des fonctions.

Figure C.7. : Niveaux de description et formalismes.

VIII. CONCLUSION

La définition d'un formalisme ou d'un outil de modélisation, pour chaque niveau de description, rend complexes les méthodes de passage entre les niveaux de description ; ce qui augmentera, par conséquent, le coût des outils logiciels de ces passages.

Il sera, alors, utile de définir un seul outil de modélisation multi-niveaux, permettant :

- La modélisation d'un système logique à différents niveaux de description.

- La simplification de passage d'une description d'un niveau donné à un autre niveau de description.

Ces deux points seront parmi les objectifs visés pour définir les REseaux MUlti-Niveaux (REMUN), dans le chapitre suivant.

(*) RTL : Register Transfert Level (Niveau transfert de registre).

CHAPITRE IV

REMUN : RESEAUX MULTI-NIVEAUX

PLAN

I. INTRODUCTION

Il est bien connu que la modélisation de systèmes logiques à l'aide des RESEAUX, offre un bon moyen de représenter ces systèmes d'une manière graphique.

D'autre part, il est couramment admis que la décomposition fonctionnelle des systèmes logiques en PLUSIEURS NIVEAUX, ou bien en parties, est utilisée comme notion de base des "bons" outils d'aide à la conception de ces systèmes.

Ces deux remarques nous ont permis de définir les RESEAUX MULTI-NIVEAUX, qui permettent la modélisation structurelle et fonctionnelle des systèmes logiques.

D. BORRIONNE [BOR] propose les différentes caractéristiques d'un outil d'aide à la conception des systèmes logiques ; parmi ces caractéristiques, l'outil doit permettre la modélisation à différents niveaux de description, pour rendre la simulation, du système décrit, efficace et moins coûteuse.

A chaque niveau de description, l'outil de modélisation doit tenir compte des propriétés structurelles et comportementales du système logique afin d'effectuer sa simulation et sa vérification fonctionnelle.

Au niveau instruction, par exemple, B. TOURSEL [TOU] a insisté sur le fait qu'un outil de modélisation doit tenir compte des différents modes de fonctionnement des calculateurs et le fait que les outils de description statiques n'en permettent pas la représentation.

Les REMUN que l'on va définir dans ce chapitre permettent la modélisation d'un système logique à différents niveaux de description et tiennent compte des propriétés appliquées à ces niveaux, en disposant de la notion de structure dynamique.

Les objectifs que nous avons visés, pour définir les REMUN, sont énoncés dans le paragraphe suivant.

II. OBJECTIFS VISES

L'implémentation d'un langage de description de systèmes logiques doit être basée sur un modèle conceptuel à l'aide duquel on vérifie les propriétés de simulation, de tests et de vérification en général.

La phase de modélisation est alors indispensable et montrée dans la figure D.1.

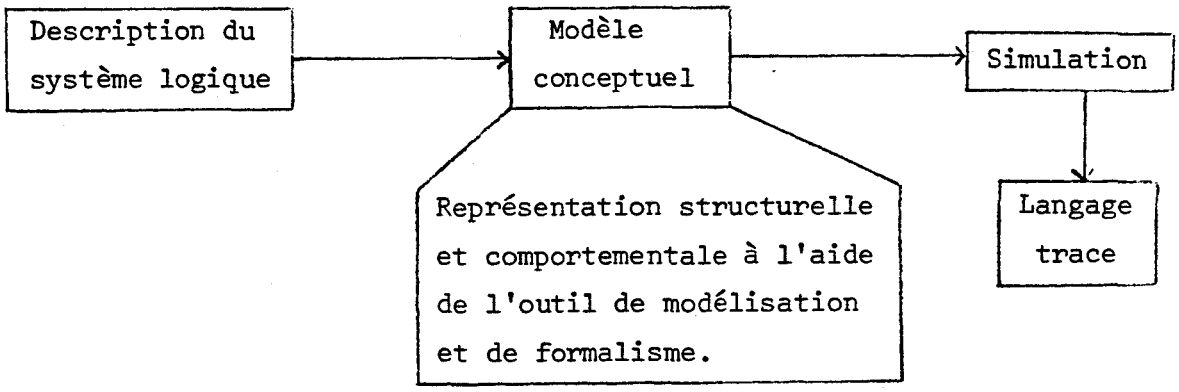


Figure D.1. : Description, modélisation et simulation.

Pour définir l'outil de modélisation REMUN, nous avons fixé les objectifs suivants :

1. Modélisation d'un système logique à tous les niveaux de description.

2. Décomposition fonctionnelle d'un système logique à un niveau donné.

3. Hiérarchisation structurelle.

4. Variation structurelle au cours du temps, ce qui entraînera une structure dynamique du modèle.

5. Définition d'applications (bijectives si possible), associées à tous les niveaux de description, des éléments de l'outil vers les éléments de la structure réelle du système.

6. Facilité de passage d'un niveau de description i à un autre niveau j.

Les deux derniers objectifs sont illustrés dans la figure D.2.

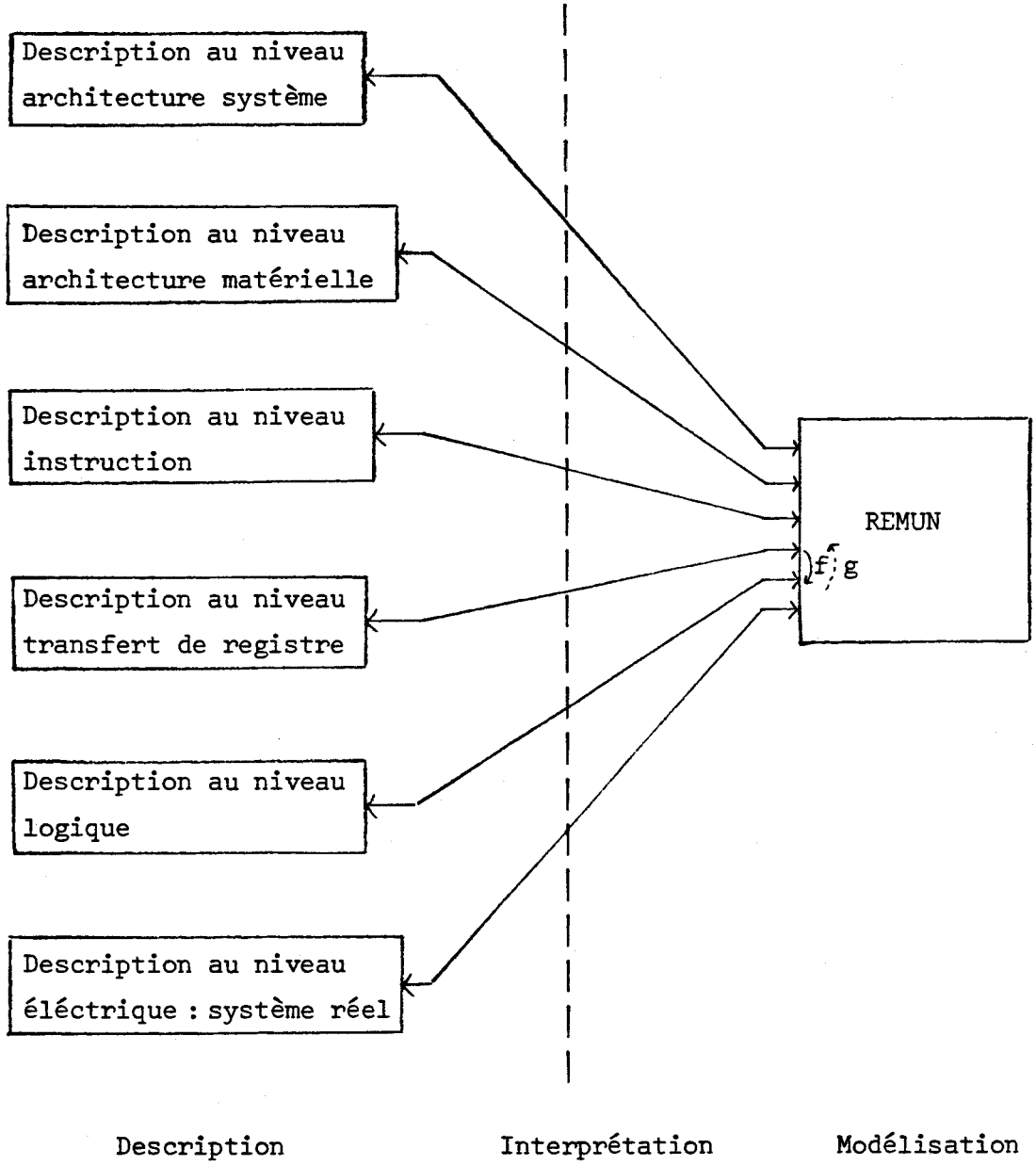


Figure D.2. : REMUN et niveau de description.
(f : application de concrétisation,
g : application d'abstraction).

Les applications de transformation f et g sont détaillées ci-après.

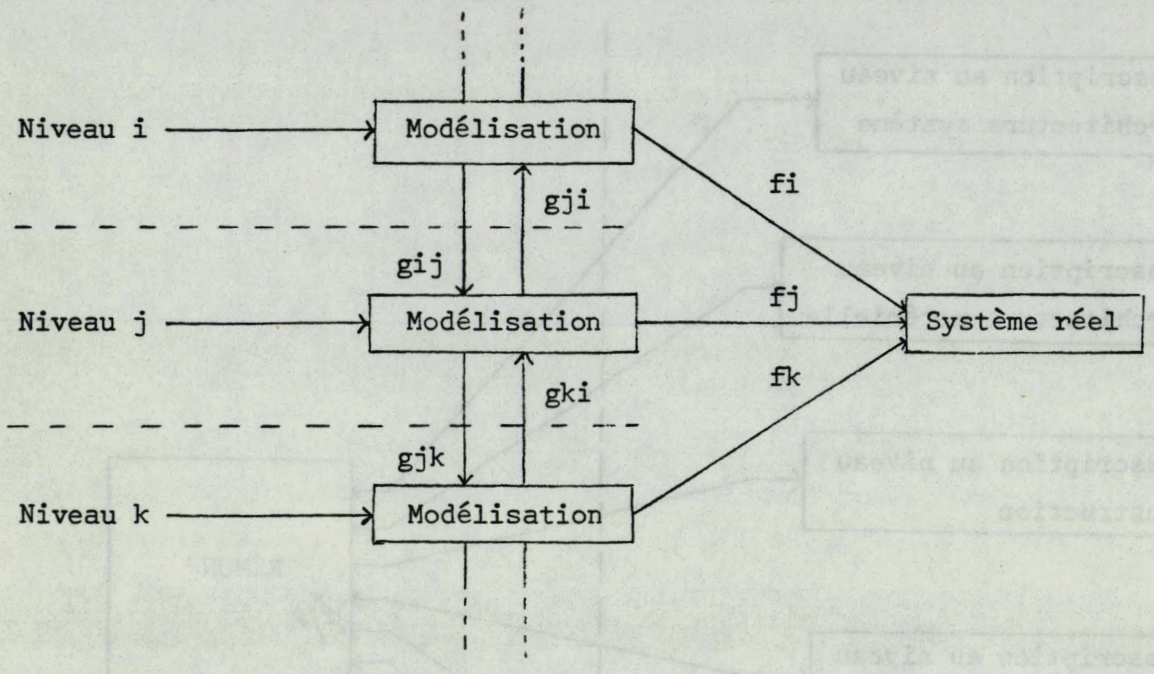


Figure D.2.bis. : Applications de transformation.

III. ELEMENTS DE LA THEORIE DES RESEAUX [KUN]

Nous allons exposer la terminologie en vigueur en théorie des réseaux, ainsi que les principales définitions, afin de faciliter l'étude antérieure sur les réseaux multi-niveaux.

III.1. Articulation

Définition 1 : Une articulation est un organe dont on ne détaille pas la structure interne et qui communique avec l'extérieur en un nombre fini de points appelés connecteurs.

Les connecteurs d'une articulation peuvent être porteurs d'une orientation. On distinguera trois types de connecteurs :

- entrées,
- sorties,
- connecteurs indifférents.

Nous nous intéressons qu'aux articulations complétement orientées où les connecteurs de la troisième catégorie ne seront pas présentés.

Représentation d'une articulation

Dans la théorie des réseaux la représentation d'une articulation est donnée sous forme d'un cercle.

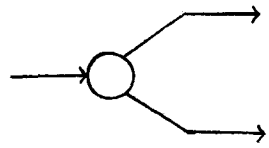


Figure D.3. : Représentation d'une articulation.

III.2. Conducteur

Définition 2 : Un conducteur est une liaison entre deux connecteurs, permettant le transfert d'une quantité d'information d'un connecteur à un autre.

La représentation d'un conducteur se fera à l'aide d'une flèche.

Un conducteur peut lier deux connecteurs de deux articulations différentes (figure D.4.a.), comme il peut lier deux connecteurs de la même articulation (figure D.4.b.).

Figure D.4.a.

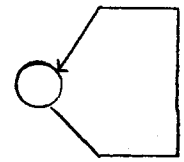


Figure D.4.b.

Définition 3 : Un réseau d'articulations R est formé d'un nombre fini d'articulations A et d'un ensemble C de conducteurs, tels que $C \subseteq A \times A$.

Exemple :

Soit R un réseau d'articulations tels que :

$A = \{a_1, a_2\}$
$C = \{(a_1, a_2), (a_2, a_1), (a_2, a_2)\}$.

R est représenté graphiquement dans la figure D.5.

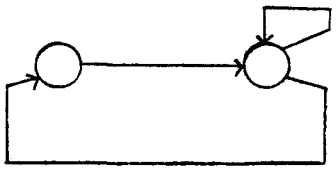


Figure D.5.

III.3. Hiérarchisation de réseaux

La hiérarchisation d'un réseau consiste à établir une répartition des articulations en niveaux à partir de la racine, suivant les deux propriétés suivantes :

- Il n'y a pas de liaisons entre les articulations d'un niveau.
- Chaque élément du $i^{ème}$ niveau provient d'exactement du $(i-1)^{ème}$ niveau.

Exemple :

Soit le réseau d'articulations représenté graphiquement dans la figure ci-dessous.

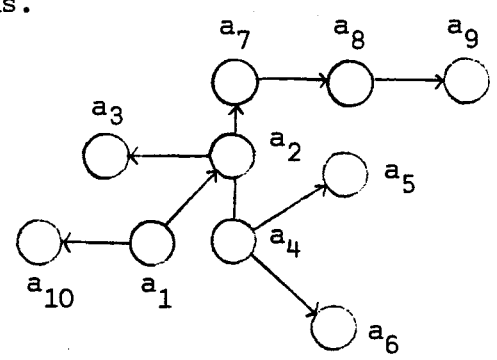


Figure D.6. : Réseau non hiérarchisé.

On remarque que l'articulation a_1 ne présente pas de connecteurs d'entrée, nous considérons alors qu'elle est la racine du réseau hiérarchisé présenté dans la figure D.7.

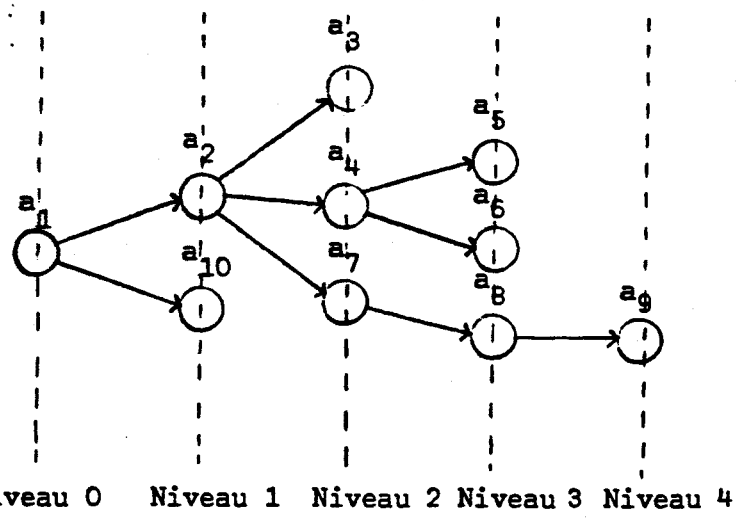


Figure D.7. : Réseau hiérarchisé.

Remarque : La notion de hiérarchie ainsi définie est purement structurelle et ne fait introduire aucun aspect fonctionnel. La hiérarchie dans les REMUN tiendra compte et de l'aspect structurel et de l'aspect fonctionnel.

III.4. Réseaux de contacts

Un réseau de contact est un réseau d'articulation dont les conducteurs sont associés à des quantités booléennes.

Exemple :

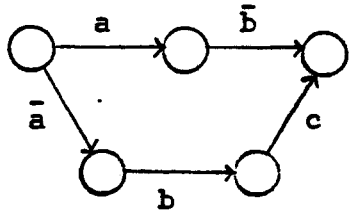


Figure D.8. : Réseau de contacts.

III.5. Réseau marqué

Soient un réseau d'articulations R et un ensemble de marques M.

Le réseau R sera dit marqué sur les articulations, si à chaque articulation correspond un sous-ensemble de M, qui sera appelé marquage de l'articulation.

Nous citerons l'exemple des réseaux de Pétri qui sont décrits au Chapitre II.

IV. NOTIONS DE BASE DES RESEAUX MULTI-NIVEAUX

Toute description et modélisation de systèmes logiques utilise l'un des deux attributs : structure ou fonction, ou même les deux pour les modélisations et descriptions mixtes.

IV.1. Notion de structure : est un attribut définissant les primitives que constitue le système logique ainsi que les connexions entre ces primitives.

IV.2. Notion de fonction

A toute primitive structurelle est associée une fonction qui détermine une partie comportementale du système au cours du temps.

P. BAKOWSKI a introduit un modèle conceptuel, en se basant sur ces deux notions. Ce modèle permet une hiérarchisation de systèmes logiques en plusieurs niveaux, chaque niveau est défini par sa propre structure et sa propre fonction [BAK].

V. DEFINITION DES ELEMENTS DES REMUN

La définition des REMUN sera basée sur les notions de structure et de contenu (ou marquage), qui correspondent d'une manière implicite aux notions de structure et de fonction. Cette correspondance s'éclaircira, par la suite, dans l'exposé.

D'autre part, les REMUN seront basés sur les relations de transition entre les notions de structure et de contenu impliquant la modélisation dynamique d'un système logique.

Nous définissons, alors, ci-après les éléments de base des REMUN :

- état structurel,
- état de contenu,
- les transitions entre ces états :
  - . transition contenu-contenu,
  - . transition structure-contenu,
  - . transition contenu-structure,
  - . transition structure-structure.

V.1. Etat structurel

Définition 1.1. : Un état structurel es est une partie instantannée d'une structure globale, défini par le couplet (P, C) où :

P : ensemble de places (ou articulations).
C : ensemble de conducteurs, avec $C \subseteq P \times P$.

Exemple 1.1. :

Soit es un état structurel défini par :

- $P = \{p_1, p_2, p_3, p_4\}$

- et $C = \{(p_1, p_2), (p_1, p_4), (p_3, p_4)\}$.

L'état structurel (P, C) est représenté graphiquement ainsi :

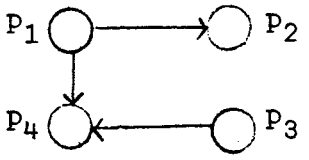


Figure D.9. : Représentation graphique d'un es.

Nous avons défini un état structurel d'une manière analogue à celle d'un réseau dirigé d'articulations, pour que les propriétés de la théorie des réseaux restent valables pour notre étude.

Définition 1.2. : Un ensemble d'états structurels ES est défini par un ensemble de places P et un ensemble de conducteurs C, tels que :

$\forall es_i \in ES$ et $es_i = (P_i, C_i)$ on a $P_i \subseteq P$ et $C_i \subseteq C$.

Exemple 1.2. :

Soit ES un ensemble d'états structurels es_1 et es_2, défini par :

$P = \{p_1, p_2, p_3, p_4\}$ et $C = \{(p_1, p_2), (p_1, p_4), (p_3, p_4), (p_2, p_3)\}$.

$es_1 = (P1, C1)$ avec $P1 = P$ et $C1 = \{(p_1, p_2), (p_1, p_4), (p_3, p_4)\}$

$es_2 = (P2, C2)$ avec $P2 = \{p_1, p_2, p_4\}$ et $C2 = \{(p_1, p_2), (p_1, p_4)\}$.

ES est représenté graphiquement ainsi :

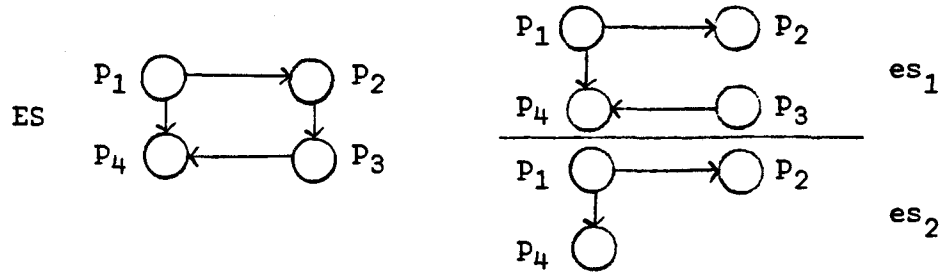


Figure D.10. : Représentation graphique de ES, es_1 et es_2.

Remarque : Un état structurel es_i est considéré comme un réseau partiel de ES.

V.2. Etat de contenu

Définition 2.1. : Un état de contenu, associé à un état structurel es = (P, C), est une application de P dans $\mathbb{N}^n$, avec $n = |P|$.

Si n est le nombre de places de es, ($|P| = n$), l'état de contenu est défini par la donnée d'un vecteur de $\mathbb{N}^n$. On note

$\overline{ec} = (m_1, m_2, \ldots, m_n)$ et $m_i = \overline{ec}(p_i)$.

L'état de contenu est ainsi défini comme un marquage dans les réseaux marqués, tels que les RdP.

Nous ne nous intéressons, ici, qu'à des états de contenu binaires. On considèrera alors qu'à une place p_i donnée, on ne peut associer que les deux états de contenu suivants :

$\overline{ec}(p_i) = 1$ si la place p_i est "pleine", c'est-à-dire qu'elle contient une marque.

$\overline{ec}(p_i) = 0$ si la place p_i est "vide", c'est-à-dire qu'elle ne contient pas de marque.

Exemple 2.1. :

Soit es un état structurel ainsi défini

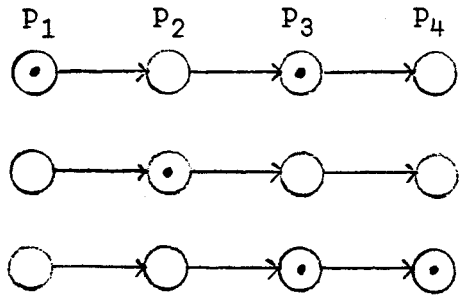
es = (P, C) tels que $P = \{p_1, p_2, p_3, p_4\}$
$C = \{(p_1, p_2), (p_2, p_3), (p_3, p_4)\}$.

Nous proposons ci-après quelques états de contenu $\overline{ec}_i$, accompagnés de leurs représentations graphiques.

$\overline{ec}_1 = (1\ 0\ 1\ 0)$ correspond à

$\overline{ec}_2 = (0\ 1\ 0\ 0)$ correspond à

$\overline{ec}_3 = (0\ 0\ 1\ 1)$ correspond à



Les notions d'état structurel et de contenu, que nous avons présentées précédemment, opposent toutes transitions ou changement d'états au cours du temps ; ce ne sont alors que des notions statiques.

Nous allons nous intéresser, dans ce qui va suivre, aux notions dynamiques, en énonçant les règles d'évolution structurelle et de contenu.

V.3. Transitions

Une séquence d'états structurels (respectivement, de contenu) est une suite d'états structurels (respectivement de contenu). Le passage d'un état à un autre obéit à des règles de transition, dont l'ensemble figure dans le tableau ci-dessous.

Transition	ec	es
ec	tc	tcs
es	tsc	ts

tc : transition de contenu.
ts : transition structurelle.
tcs : transition contenu-structure.
tsc : transition structure-contenu.

Tableau D.11. : Ensemble des transitions.

Nous allons définir d'abord les transitions qui permettent l'établissement d'un seul niveau : niveau ouvert, puis les transitions entre les différents niveaux qui permettent l'établissement du réseau multi-niveaux (REMUN).

V.3.1. Transition de contenu

Définition 3.1. : Etant donné es un état structurel, une transition de contenu - à travers es - est le passage instantanné d'un état de contenu à un autre état de contenu suivant une règle d'évolution.

Notation 3.1. :

Soient es_i un état structurel et tc_i la transition de contenu tc_i qui est une application d'un ensemble d'états de contenu EC_i, associé à es_i, dans lui-même.

$$tc_i : EC_i \rightarrow EC_i.$$

Règle d'évolution

$$\text{Soit } es_i = (P_i, C_i) \text{ tels que : } P_i = \{p_1, p_2\}$$
$$\text{et } C_i = \{(p_1, p_2)\}.$$

$$\text{Soit } EC_i = \{\overline{ec}_0, \overline{ec}_1, \overline{ec}_2, \overline{ec}_3\}$$
$$\text{tels que : } \overline{ec}_0 = (0, 0),$$
$$\overline{ec}_1 = (0, 1),$$
$$\overline{ec}_2 = (1, 0)$$
$$\text{et } \overline{ec}_3 = (1, 1).$$

On distinguera deux types d'évolution :

i - Evolution avec conservation : où l'état de contenu de la place, de l'origine du conducteur, p_1 s'affecte à l'état de contenu de la place, du but du conducteur, p_2 ; en conservant l'état de contenu de p_1.

Algorithmiquement, la règle d'évolution avec conservation est ainsi définie :

$$ec(p_2) := ec(p_1).$$

On représente d'abord cette évolution de contenu d'une manière graphique (Tableau D.12.) pour mieux l'illustrer.

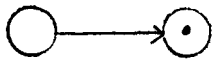
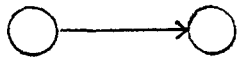
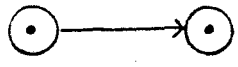
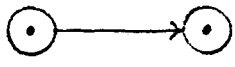
$\overline{ec}_i$ avant la transition	$\overline{ec}_j$ après la transition
	
	
	
	

Tableau D.12. : Evolution avec conservation.

La transition de contenu avec conservation est notée tc1.

D'après le tableau D.12, on définit tc1 ainsi :

$$tc1 : EC \rightarrow EC$$
$$\overline{ec}_0 \rightarrow \overline{ec}_0$$
$$\overline{ec}_1 \rightarrow \overline{ec}_0$$
$$\overline{ec}_2 \rightarrow \overline{ec}_3$$
$$\overline{ec}_3 \rightarrow \overline{ec}_3$$

ii - Evolution sans conservation : où l'état de contenu de la place, de l'origine du conducteur, p_1 s'affecte à l'état de contenu de la place, du but du conducteur, p_2 ; en remettant l'état de contenu de p_1 à zéro.

Algorithmiquement, la règle d'évolution sans conservation est ainsi définie :

$$ec(p_2) := ec(p_1),$$
$$ec(p_1) := 0.$$

On représente d'abord cette évolution de contenu d'une manière graphique (Tableau D.12.bis.) pour mieux l'illustrer.

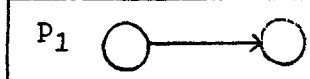
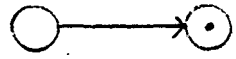
$\overline{ec}_i$ avant la transition	$\overline{ec}_j$ après la transition
	
	
	
	

Tableau D.12.bis. : Evolution sans conservation.

La transition de contenu sans conservation est notée tc0.

D'après le tableau D.12.bis, on définit tc0 ainsi :

$$tc0 : EC \rightarrow EC$$
$$\overline{ec}_0 \rightarrow \overline{ec}_0$$
$$\overline{ec}_1 \rightarrow \overline{ec}_0$$
$$\overline{ec}_2 \rightarrow \overline{ec}_1$$
$$\overline{ec}_3 \rightarrow \overline{ec}_1$$

Une transition de contenu, avec ou sans conservation, ne s'effectuera que si le conducteur, à travers lequel, elle se réalise est actif. Donc elle dépendra de son état structurel ; d'où la transition structure-contenu.

V.3.2. Transition structure-contenu

Définition 4 : Une transition structure-contenu (notée tsc) est une implication structurelle définie par une application de es dans tc.

$$tsc : es \mapsto tc$$

où "$\mapsto$" est le symbole de l'implication structurelle.

De même, un ensemble de transitions structure-contenu est ainsi défini :

$$TSC : ES \Mapsto TC.$$

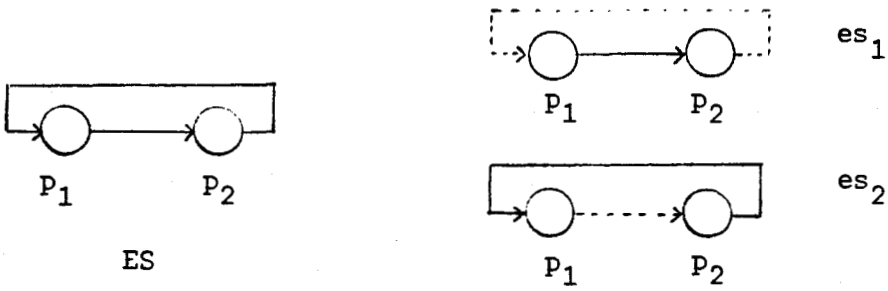
Exemple 4 :

Soit ES = (P, C) tel que $P = \{p_1, p_2\}$ et $C = \{(p_1, p_2), (p_2, p_1)\}$, étant donnés es_1, $es_2 \in ES$ tels que

es_1 = (P1, C1) avec P1 = P et $C1 = \{(p_1, p_2)\}$

et es_2 = (P2, C2) avec P2 = P et $C2 = \{(p_2, p_1)\}$.

ES, es_1 et es_2 sont graphiquement représentés ci-dessous.



L'ensemble des transitions structure-contenu sont les suivantes :

$$TSC : ES \Mapsto TC,$$
$$es_1 \mapsto tc_1,$$
$$es_2 \mapsto tc_2.$$

Les transitions de contenu tc_i, impliquées par les états structurels es_i, sont déterminées en tenant-compte des règles d'évolution déjà définies.

Nous avons introduit tous les éléments de base que comporte un niveau ouvert, que nous définissons ci-après.

V.4. Niveau ouvert

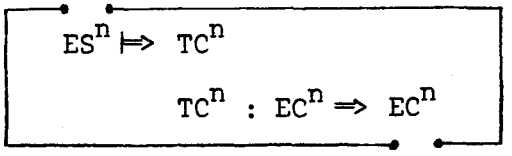
Nous allons nous intéresser, ici, à un seul niveau qui pourra être lié par la suite à d'autres niveaux formant ainsi un REMUN.

Un niveau est qualifié d'ouvert, si sa structure est modifiable par son environnement.

Définition 5 : Un niveau ouvert n est la donnée d'un doublet (ES^n, EC^n) où ES^n est l'ensemble d'états structurels du niveau n, et EC^n est l'ensemble d'états de contenu du niveau n.

L'ensemble d'états structurels ES^n implique un ensemble de transitions de contenu TC^n (Définition 4) tels que $TC^n : EC^n \Rightarrow EC^n$.

Nous représentons alors un niveau ouvert de la façon suivante :



Pour permettre une transition de contenu, à travers un état structurel donné, il faut effectuer une sélection (ou activation) de cet état.

La transition contenu-structure, que nous allons introduire, permettra la sélection (ou l'activation) citée.

V.5. Transition contenu-structure

La transition contenu-structure, comme son nom l'indique, relate un état de contenu d'un niveau n, avec un état de structure d'un niveau k.

Définition 6 : Une transition contenu-structure tcs_i^n, d'un niveau n, associe à un état de contenu $\overline{ec}_i^n$ (du niveau n) un état structurel es_i^k (du niveau k).

Notation :

Transition contenu-structure du niveau n au niveau k :

$$tcs_i^n : \overline{ec}_i^n \rightarrow\!| \; es_i^k$$

Ensemble de transitions contenu-structure :

$$TCS^n : EC^n \Rightarrow\!| \; ES^k.$$

La transition tcs_i^n associe l'état de contenu d'une place p_1^n du niveau n à un conducteur (p_1^k, p_2^k) du niveau k.

Nous représenterons graphiquement cette transition contenu-structure de la façon suivante :

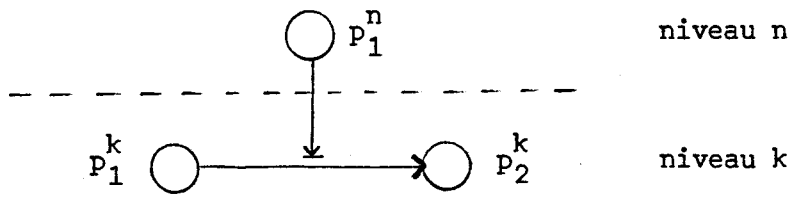


Figure D.13. : Représentation graphique d'une tcs_i^n.

La transition contenu-structure est appliquée à différents niveaux de description. Nous citerons quelques applications de tcs aux niveaux suivants :

- niveau électrique,
- niveau logique,
- niveau transfert de registre,
- niveau instruction.

L'application des REMUN au niveau transfert de registre sera détaillée au Chapitre V où un langage de description de système logique sera défini.

Un exemple de simulation d'une description au niveau architecture système sera exposé après la définition de LREMUN : langage de description des REMUN.

V.5.1. Niveau électrique

On s'intéressera à la technologie N-MOS, présentée au Chapitre I.

Un transistor N-MOS présente la caractéristique fonctionnelle d'un interrupteur, définissant les deux états structurels suivants :

- état structurel "ouvert",
- état structurel "fermé".

L'état structurel est déterminé par la valeur (ou l'état de contenu) de la grille et implique une transition de contenu de la source au drain.

L'application de la tcs sera ainsi schématisée dans la figure D.14.

Figure D.14. : Modélisation d'un transistor N-MOS par une tcs.
(PG, PS, PD : Places Grille, Source, Drain).

En tenant compte des états de contenu de PG, PS et PD ainsi qu'aux "valeurs logiques" de G, S et D, nous présentons dans le tableau ci-dessous les fonctionnements d'un transistor N-MOS et d'une transition contenu-structure.

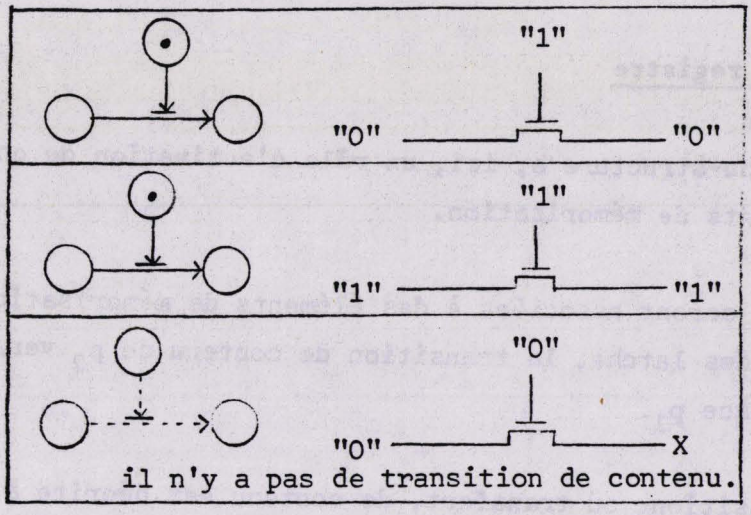


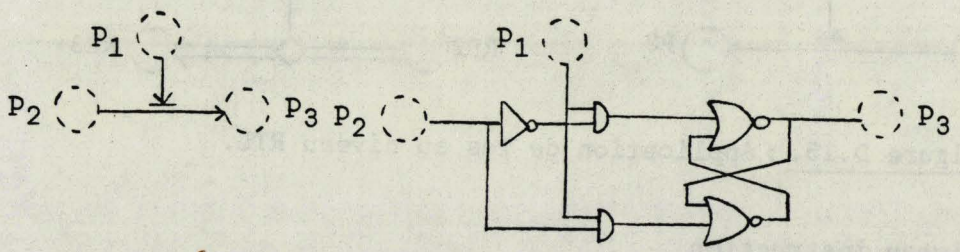
Tableau D.15. : Correspondance des fonctionnements.

Quand $\overline{ec}$ (PG) = 0, le conducteur (PS, PD) n'est pas actif ce qui n'implique pas la transition de contenu de PS à PD ; Ce qui correspond au régime bloqué (absence de canal : aucun courant ne peut circuler entre S et D) chez le transistor.

V.5.2. Niveau combinatoire

En ce niveau, un noeud de transition contenu correspondra à un point de mémorisation, obtenu par rétrocouplage de deux porte NON-OU (ou NON-ET).

L'application d'une tcs sera ainsi définie de la façon suivante :



Pour la représentation d'une transition contenu-structure, à l'aide de portes logiques, ci-dessus, la place p_1 est considérée comme une place de commande de la transition du contenu de p_2 dans p_3.

V.5.3. Niveau transfert de registre

La transition contenu-structure a, ici, un rôle d'activation de chemin de données entre les éléments de mémorisation.

Les places p_2 et p_3 seront associées à des éléments de mémorisation, tels que des registres ou des latchs, la transition de contenu de p_2 vers p_3 est commandée par la place p_1.

L'opération de transition, ou transfert, de contenu est décrite à l'aide du langage de description CDL de la façon suivante :

```
REGISTER,
     RP2(0 - 7),
     RP3(0 - 7),
SWITCH,
     SP1(ON),
/SP1(ON)/RP3 ← RP2.
```

La représentation graphique, de cette transition de contenu, est actuellement connue sous la forme suivante :

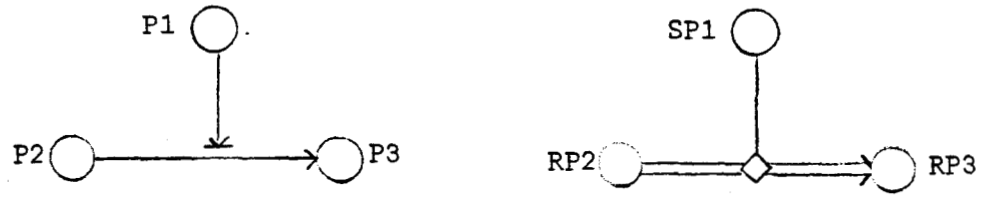


Figure D.15. : Application de tcs au niveau RTL.

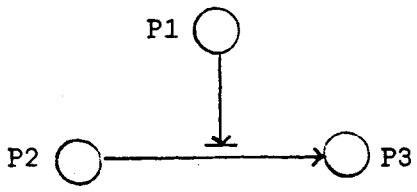
V.5.4. Niveau instruction

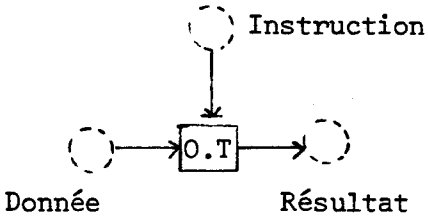
La transition contenu-structure a un rôle d'activation d'une partie de l'organe de traitement qui correspond à l'instruction à exécuter.

La place P2 sera associée à une donnée d'entrée à l'organe.

La place P3 sera associée à la donnée traitée et ses indications, donc au résultat de l'exécution de l'instruction.

Et la place P1 sera associée à l'instruction.





L'organe de traitement (O.T) de données sera activé s'il se trouve à l'interconnexion du flot d'instruction et du flot de donnée.

Cela nous amènera à la classification des systèmes, au niveau instruction, présentée par M. FLYNN [FLY].

Remarque : Les transitions de contenu utilisées correspondent à des évolutions avec conservation, car leurs structures sont utilisées pour "l'écoulement" du flot de données. Tandis que pour le flot de contrôle, seront utilisées les transitions de contenu à évolution sans conservation.

VI. RESEAUX MULTI-NIVEAUX (REMUN)

Un réseau multi-niveaux est composé d'un nombre fini n de niveaux ouverts (Définition 5) ; Chaque niveau est défini par sa propre structure.

Les interactions entre ces niveaux sont déterminées à l'aide de transitions contenu-structure.

Définition 7 : Un REMUN à n niveaux est défini par :

- n niveaux ouverts.
- (n-1) transitions contenu-structure.

Les liens entre les niveaux sont représentés dans la figure D.16.

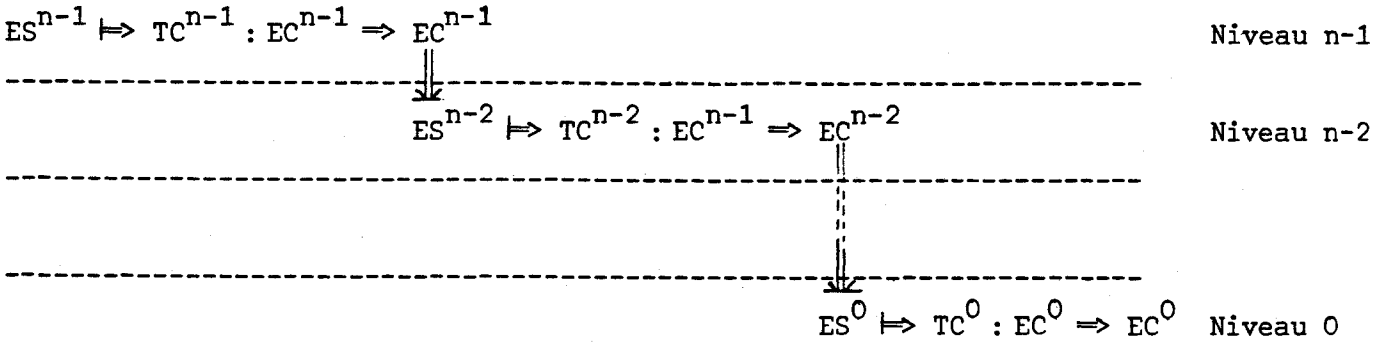


Figure D.16. : Représentation d'un REMUN.

VII. DEFINITION FORMELLE D'UN REMUN

Un REseau MUlti-Niveaux (REMUN) est défini par la donnée d'un quadruplet $(P, TC, TCS, \overline{ec_0})$ avec :

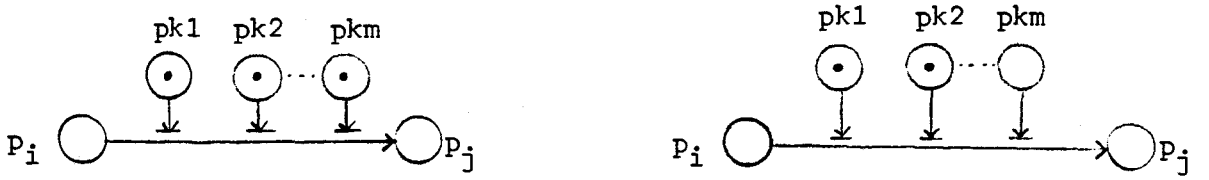
P : ensemble fini de places.
TC : ensemble fini de transitions de contenu, $TC \subseteq P \times P$.
TCS : ensemble fini de transitions contenu-structure, $TCS \subseteq P \times TC$.
$\overline{ec_0}$: état de contenu initial avec $\overline{ec_0} : p \to \{0, 1\}^p$ où $p = |P|$.

Règles d'évolution

Une transition de contenu $(p_i, p_j) \in TC$ peut se trouver à l'un des deux états structurels suivants :

- état actif si $\forall p_k \in P$ et $(p_k, (p_i, p_j)) \in TCS$, $ec(p_k) = 1$,
- état inactif sinon.

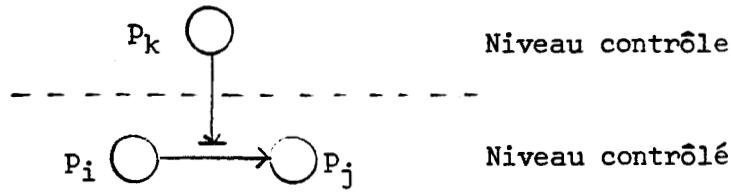
Ce dernier état structurel n'implique pas d'évolution ; nous nous intéresserons, alors, à l'état structurel actif.



(p_i, p_j) actif. (p_i, p_j) inactif.

Figure D.17. : Etats d'une transition de contenu.

Nous énoncerons les règles d'évolution pour la transition contenu-structure suivante :



p_k sera nommée "place contrôle".
(p_i, p_j) sera nommée "transition de contenu contrôlée".

Les règles d'évolution tiendront compte de la conservation du niveau contrôle, donc de l'état de contenu de p_k, et de la conservation du niveau contrôlé, donc de l'état de contenu de p_i.

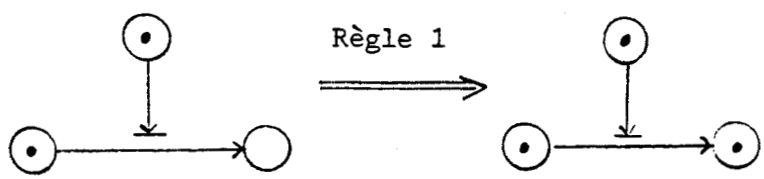
Quatre règles d'évolution seront alors distinguées :

- évolution avec conservation de $ec(p_k)$ et $ec(p_i)$,
- évolution avec conservation de $ec(p_k)$ et sans conservation de $ec(p_i)$,
- évolution sans conservation de $ec(p_k)$ et avec conservation de $ec(p_i)$,
- évolution sans conservation de $ec(p_k)$ et sans conservation de $ec(p_i)$.

La représentation graphique et algorithmique de chacune de ces règles seront présentées.

Règle 1 : Evolution avec conservation de $ec(p_k)$ et $ec(p_i)$.

Représentation graphique :



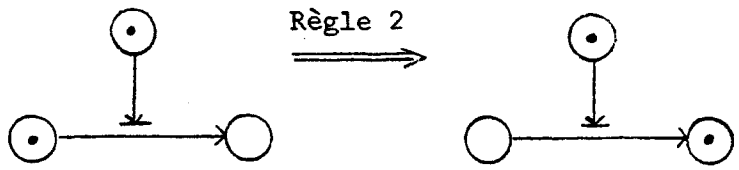
Représentation algorithmique :

```
Règle 1 (pi, pj)
  si pour tout (pk, (pi, pj)) ∈ TCS, ec(pk) = 1 alors
     ec(pj) := ec(pi)
  fsi
```

Cette règle d'évolution est utilisée au niveau de donnée, où les places sont associées à des éléments de mémorisation, comme ce qui a été montré au paragraphe V.5.

Règle 2 : Evolution avec conservation de $ec(p_k)$ et sans conservation de $ec(p_i)$.

Représentation graphique :



Représentation algorithmique :

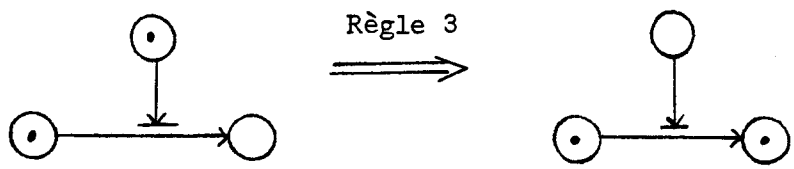
Règle 2 (p_i, p_j)
si pour tout $(p_k, (p_i, p_j)) \in TCS$, $ec(p_k) = 1$ alors
$ec(p_j) := ec(p_i)$;
$ec(p_i) := 0$
fsi

Les REMUN qui appliquent cette règle d'évolution permettent la modélisation de la partie contrôle d'un système logique, cette partie sera décomposée fonctionnellement en plusieurs niveaux.

Cette règle sera utilisée dans la suite de notre étude pour la modélisation du contrôle des systèmes.

Règle 3 : Evolution sans conservation de $ec(p_k)$ et avec conservation de $ec(p_i)$.

Représentation graphique :

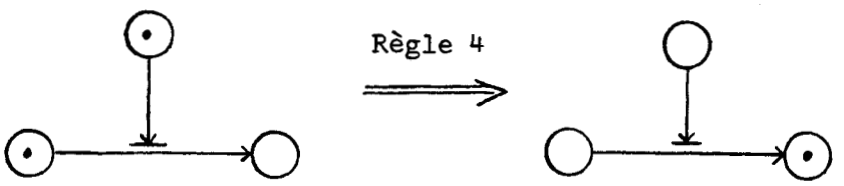


Représentation algorithmique :

Règle 3 (p_i, p_j)
si pour tout $(p_k, (p_i, p_j)) \in TCS$, $ec(p_k) = 1$ alors
$ec(p_j) := ec(p_i)$;
pour tout $(p_k, (p_i, p_j)) \in TCS$ faire $ec(p_k) := 0$
fsi

Règle 4 : Evolution sans conservation de $ec(p_k)$ et sans conservation de $ec(p_i)$.

Représentation graphique :



Représentation algorithmique :

Règle 4 (p_i, p_j)
  si $(ec(p_i) = 1)$ et (pour tout $(p_k, (p_i, p_j)) \in TCS$, $ec(p_k) = 1$) alors
    $ec(p_j) := 1$;
    $ec(p_i) := 0$;
    pour tout $(p_k, (p_i, p_j)) \in TCS$ faire $ec(p_k) := 0$
  fsi

Cette règle d'évolution est analogue à celle des réseaux de Pétri, présentée au II.6.1, comme le montre la figure ci-après.

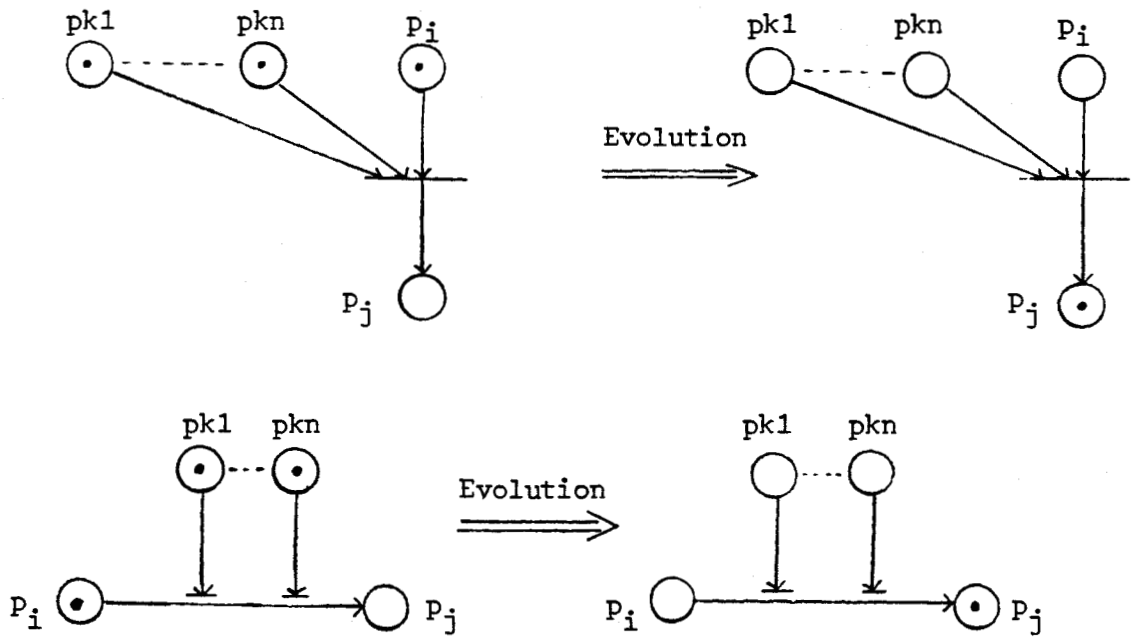


Figure D.18. : Analogie de la règle d'évolution des RdP avec la règle 4.

Nous utiliserons dans la suite de notre étude la règle 2, car une place du REMUN sera associée à un état, du système modélisé ; cette place pourra participer à l'évolution du système, tout en gardant le même état.

Afin de représenter textuellement un REMUN, il sera utile de proposer un langage de description des REMUN permettant la représentation structurelle et de contenu.

VIII. LANGAGE DE DESCRIPTION DES REMUN

Nous avons défini un langage de description, dont la syntaxe permet la représentation de la structure initiale d'un REMUN.

VIII.1. Présentation de LREMUN

La grammaire de ce langage (LREMUN) est présentée ci-dessous, sous la forme BACHUS-NAUR (BNF).

<desc. remun> ::= DEBREMUN <liste des niveaux> FREMUN
<liste des niveaux> ::= <desc. niveau> | <desc. niveau> <liste des niveaux>
<desc. niveau> ::= DEBNIV <num. niv.> <liste des places> FNIV
<num. niv.> ::= <entier>
<liste des places> ::= <desc. place> | <desc. place> <liste des places>
<desc. place> ::= DEBPLACE <num. place> <ec> <nom place>
<liste de transitions> FPLACE
<num. place> ::= <entier>
<ec> ::= 0|1
<nom place> ::= <identificateur>
<liste de transitions> ::= <transition> | <transition> <liste de transitions>
<transition> ::= <tc> | <tcs>
<tc> ::= C <num. niv.> <num. place>
<tcs> ::= S <num. niv.> <num. place> <num. niv.> <num. place>
<entier> ::= <chiffre> | <chiffre> <entier>
<chiffre> ::= 0|1|...|9
<identificateur> ::= <lettre> | <lettre> <identificateur>
<lettre> ::= A|B|...|Z

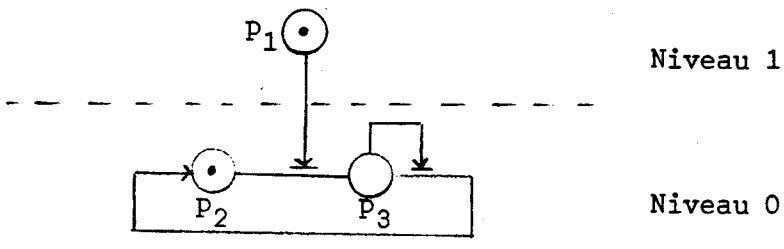
Pour illustrer la description des REMUNs à l'aide de LREMUN, nous donnons un exemple de description d'un REMUN simple.

VIII.2. Exemple de description de REMUN

Soit un REMUN $\mathcal{R}$ défini par $R = (P, TC, TCS, \overline{ec}_0)$ tels que :

$P = \{p_1, p_2, p_3\}$
$TC = \{(p_2, p_3), (p_3, p_2)\}$
$TCS = \{(p_1, (p_2, p_3)), (p_3, (p_3, p_2))\}$
$\overline{ec}_0 = (1\ 1\ 0)$

$\mathcal{R}$ est représenté graphiquement de la façon suivante.



La description textuelle de $\mathcal{R}$ à l'aide de LREMUN sera ainsi définie :

```
DEBREMUN
     DEBNIV  1
        DEBPLACE  1  1  P1  co numéro, ec, nom co
              S  0  1  0  2
        FPLACE
     FNIV
     DEBNIV  0
        DEBPLACE  1  1  P2
             C  0  2
        FPLACE
        DEBPLACE  2  0  P3
             C  0  1  S  0  2  0  1
        FPLACE
     FNIV
FREMUN
```

IX. SIMULATEUR DE REMUN

La modélisation dynamique et la simulation de REMUN consistent à déterminer les places actives au cours du temps.

Cette simulation a été développée pour les deux objectifs principaux suivants :

- Validation de l'outil de modélisation REMUN.
- Simulation du séquencement à différents niveaux de description.

Le simulateur génère un langage trace, comme le montre la figure D.19, ce langage définit les places actives à différentes étapes de simulation.

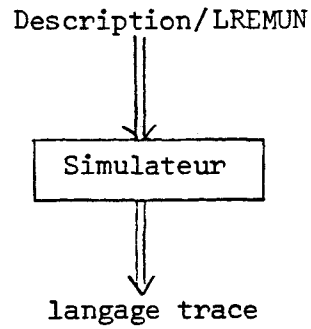
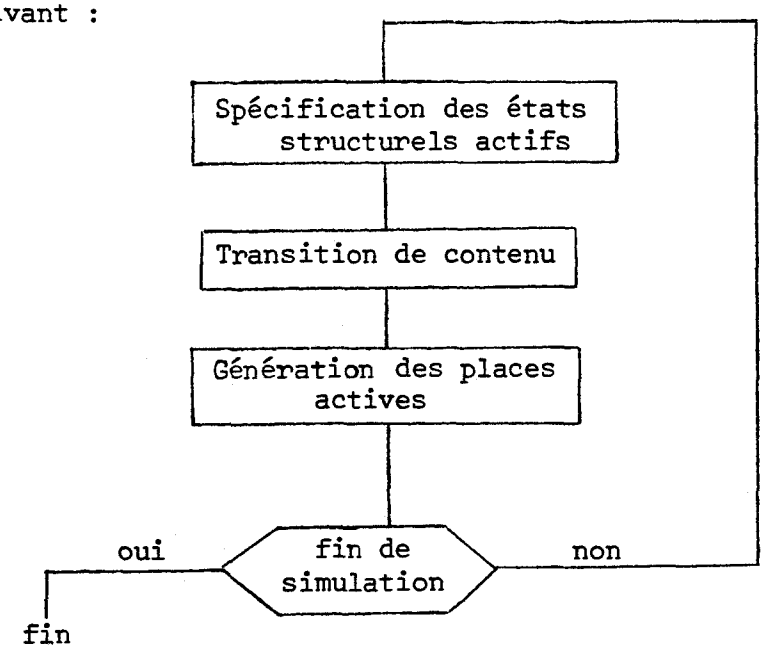


Figure D.19. : Simulateur de REMUN.

IX.1. Présentation du simulateur

D'une manière globale, le simulateur de REMUN apparaît comme le cycle répétitif suivant :



Les différents modules du simulateur sont présentés en annexe.

Nous présentons sur un exemple, l'utilisation des REMUN pour modéliser un système logique au niveau architecture système afin de le simuler.

IX.2. Exemple de simulation au niveau architecture système

Nous traitons le problème de synchronisation de deux processus producteur-consommateur avec un tempon unique.

Le processus consommateur prend, à tout instant, l'un des deux états suivants :

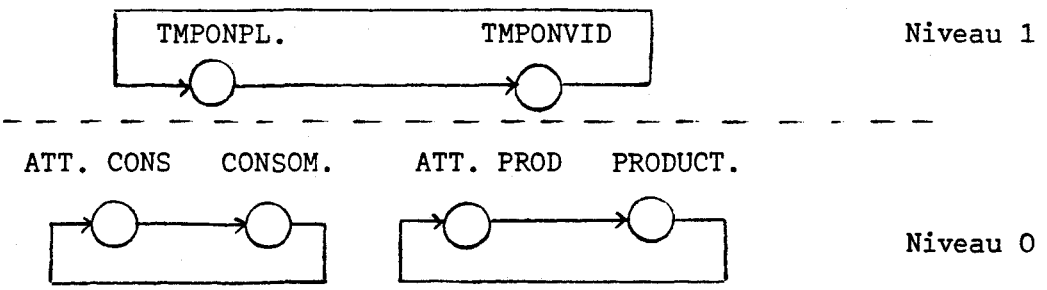
- attente de consommation ATT. CONS
- consommation CONSOM.

Idem pour le producteur, dont les états sont les suivants :

- attente de production ATT. PROD
- production PRODUCT.

Le tampon peut être - soit à l'état plein TMPONPL.
- soit à l'état vide TMPONVID.

A chaque état sera associé une place et les transitions de contenu entre les places apparaissent ainsi :



La décomposition du système en ces deux niveaux, est effectuée car chacun peut impliquer l'évolution de l'autre, en définissant les transitions contenu-structure (figure D.20.).

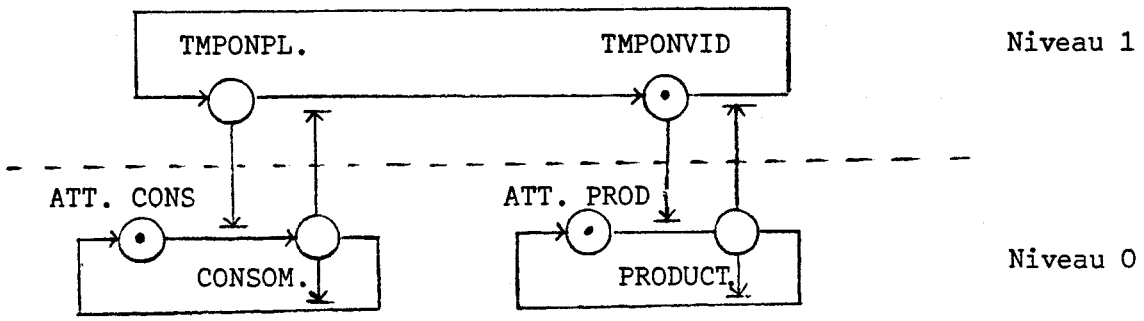


Figure D.20. : Le REMUN (Producteur-consommateur).

La description du REMUN, modélisant la synchronisation des deux processus, est la suivante :

```
DEBREMUN
    DEBNIV   1
        DEBPLACE  1 0   TMPONPL.   C 1 2   S 0 1 0 2   FPLACE
        DEBPLACE  2 1   TMPONVID   C 1 1   S 0 3 0 4   FPLACE
    FNIV
    DEBNIV   0
        DEBPLACE  1 1   ATT.CONS   C 0 2    FPLACE
        DEBPLACE  2 0   CONSOM.    C 0 1   S 0 2 0 1   S 1 1 1 2  FPLACE
        DEBPLACE  3 1   ATT.PROD   C 0 4    FPLACE
        DEBPLACE  4 0   PRODUCT.   C 0 3   S 0 4 0 3   S 1 2 1 1  FPLACE
    FNIV
FREMUN
```

Cette description est fournie au simulateur, qui imprime d'abord l'état initial du REMUN, d'une manière tabulaire, puis son comportement au cours du temps en ne générant que les places actives.

NIVEAU	NOEUD	NOM	EC	LIAIS. CONT-CONT.			LIAIS. CONT-STRUCT.			
				NIVEAU	NOEUD	ACTIVE	NIVEAU	NOEUD	NIVEAU	NOEUD
1										
	1	TMPONPL.	F							
				1	2	F				
							0	1	0	2
	2	TMPONVID	T							
				1	1	F				
							0	3	0	4
0										
	1	ATT.CONS	T							
				0	2	F				
	2	CONSOM.	F							
				0	1	F				
							0	2	0	1
							1	1	1	2
	3	ATT.PROD	T							
				0	4	T				
	4	PRODUCT.	F							
				0	3	F				
							0	4	0	3
							1	2	1	1

Figure D.21. : Modélisation statique.

```
*****************************************
ETAPE :  1
**********
NIVEAU  1
----------
   TMPONVID
NIVEAU  0
----------
   ATT.CONS    ATT.PROD

*****************************************
ETAPE :  2
**********
NIVEAU  1
----------
   TMPONVID
NIVEAU  0
----------
   ATT.CONS    PRODUCT.

*****************************************
ETAPE :  3
**********
NIVEAU  1

----------
   TMPONPL.
NIVEAU  0
----------
   ATT.CONS    ATT.PROD

*****************************************
ETAPE :  4
**********
NIVEAU  1
----------
   TMPONPL.
NIVEAU  0
----------
    CONSOM.     ATT.PROD

*****************************************
ETAPE :  5
**********
NIVEAU  1
----------
    TMPONVID
NIVEAU  0
----------
    ATT.CONS    ATT.PROD
```

Figure D.22. : Simulation du REMUN.

X. CONCLUSION

Afin d'utiliser les REMUN pour des applications de simulation et de conception de systèmes logiques, il est nécessaire de rendre leurs concepts transparents et d'offrir à l'utilisateur des primitives dont les nominations sont plus proches du matériel.

Initialement, nous avons visé la définition d'un langage de description multi-niveaux. Un tel langage doit disposer d'une collection de primitives, de tous les niveaux, allant du transistor jusqu'au processus.

Cette voie nous est apparue assez complexe dans le sens où l'utilisateur va disposer d'une multitude de symbolismes, d'où la difficulté d'appréhender le langage.

La démarche que nous avons alors suivie consiste à définir un langage pour chaque niveau de description :

LAS : Langage du niveau Architecture Système.
LAM : Langage du niveau Architecture Matérielle.
LI : Langage du niveau Instruction.
LTR : Langage du niveau Transfert de Registre
LL : Langage du niveau Logique.
LE : Langage du niveau Electrique.

A chaque niveau de description sera, donc, associé :

- un langage de description,
- un interpréteur I, d'une description (d'un système logique en ce langage) en LREMUN,
- un traducteur T, de niveau de description.

L'ensemble du système est montré dans la figure D.23.

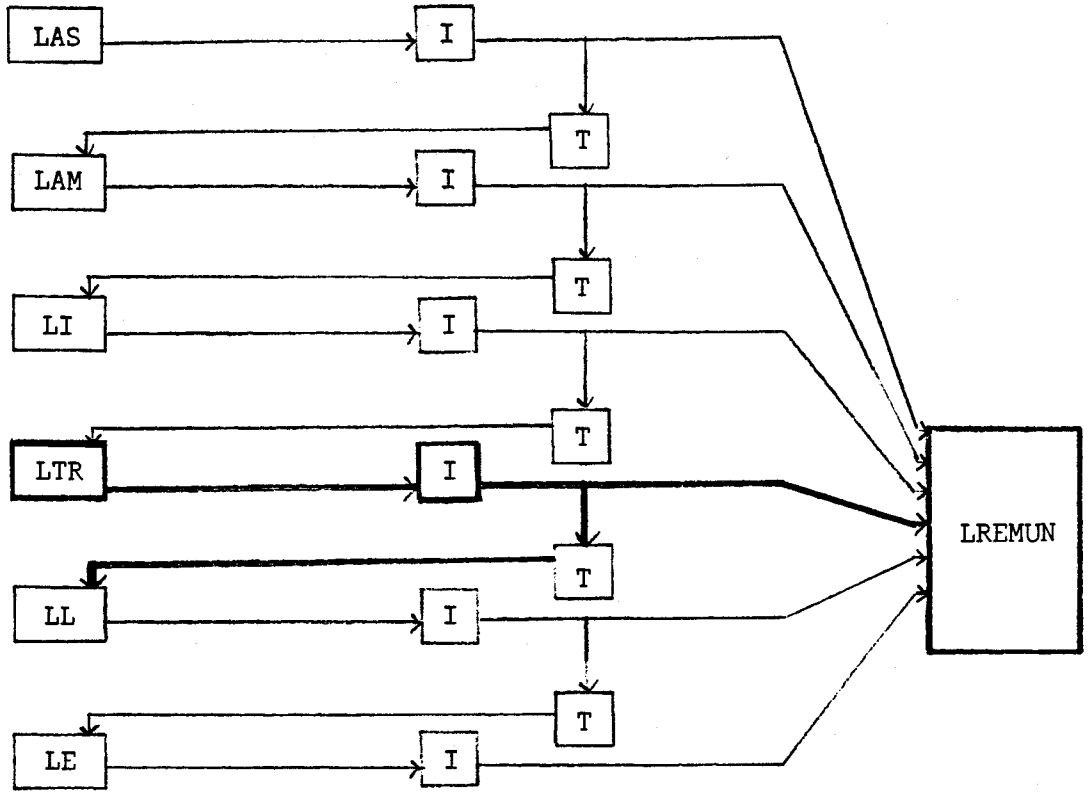


Figure D.23. : Système multi-niveaux.

Pour une application du système, nous avons réalisé tout ce qui figure en gras ci-dessus, c'est-à-dire :

- Un langage au niveau transfert de registre : LIDO.
- Un interpréteur (LIDO → LREMUN).
- Un traducteur (LREMUN → niveau logique).

CHAPITRE V

LIDO : LANGAGE INTERPRETE DE DESCRIPTION DES ORDINATEURS

PLAN

I. INTRODUCTION

Les réseaux multi-niveaux (REMUN) ont été développés pour modéliser un système logique à différents niveaux de description.

Dans l'optique d'appliquer cet outil de modélisation au niveau transfert de registre, nous proposons un langage de description de systèmes logiques bâptisé LIDO : Langage Interprété de Description des Ordinateurs.

LIDO est qualifié d'interprété car toute description à l'aide de ce langage peut être traitée par un interpréteur qui génère la description équivalente en LREMUN qui sera elle-même par la suite fournie au simulateur des REMUN pour effectuer la simulation fonctionnelle du système logique décrit initialement.

Notre propos n'est pas d'introduire de nouveaux symbolismes ou de nouvelles notations des langages de description, mais d'utiliser des symbolismes qui existent déjà dans ce domaine et qui s'adaptent le plus à l'approche des notions des REMUN.

Les caractéristiques principales que doit donc avoir le langage LIDO sont les suivantes :

1. Décomposition fonctionnelle des systèmes logiques en niveaux systémiques, chacun de ces niveaux sera associé à un mécanisme du langage qui sera défini par un ensemble d'objets fonctionnels munis de leurs connexions, connexions dont l'activation instantanée est déterminée par des états fonctionnels d'un autre mécanisme donc d'un autre niveau, ceci conduit à la deuxième caractéristique de hiérarchie structurelle.

2. Hiérarchie structurelle au sens des REMUN. Chaque mécanisme a un rôle fonctionnel d'activation structurelle donc des connexions d'un niveau subordonné.

La définition de LIDO ne doit pas faire intervenir le formalisme de REMUN, ce formalisme doit être transparent pour l'utilisateur.

II. NIVEAU TRANSFERT DE REGISTRE

Le système logique est décrit en terme de registres qu'il comporte, son comportement apparaît comme des transferts et des traitements de contenu de ces registres.

Les langages qui offrent une telle description sont de type RTL (Register Transfert Level : Niveau transfert de registre) ; ils sont souvent utilisés pour des fins de conception de systèmes logiques où ils spécifient la définition architecturale du système. L'intérêt de l'utilisation des langages de type RTL dans le domaine de la conception apparaît dans la finesse de la description structurelle de l'implémentation du chemin de données ainsi que de la partie de contrôle.

Une classification de ces langages, largement acceptée, a été proposée par BARBACCI [BAR]. Elle tient compte du caractère procédural de ces langages. Deux classes de langages de type RTL apparaissent ainsi :

Classe 1 : Les langages procéduraux ; où l'ordre d'exécution des opérations élémentaires est impliqué par leur ordre lexicographique dans la description.

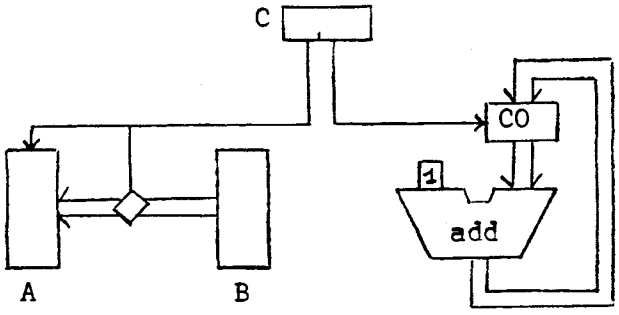
Classe 2 : Les langages non-procéduraux ; où l'ordre d'exécution des opérations élémentaires et leur ordre d'écriture sont indépendants.

Parmi les principaux avantages que présente la première classe, on trouve la facilité de leur utilisation et de leur interprétation dans le sens où ils sont plus proches des langages de programmation tels que PASCAL, APL,... etc... Nous citons l'exemple de AHPL qui est basé sur la notation d'APL. Or la simultanéîté d'exécutions d'opérations élémentaires, dans les systèmes logiques réels, nécessite la disposition de primitives pour exprimer ce parallélisme, ce qui a conduit à l'utilisation des concepts similaires à ceux qu'on trouve en ALGOL 68, Pascal Concurrent,... etc...

Exemple :

```
REGISTRE  % déclaration des registres %
    A(7 : 0) ; B(7 : 0) ; C(1 : 0) ; CO(15 : 0) ;
  DEBUT
    SI C(1) ALORS A := B FSI, %, : exécution simultanée %
    SI C(0) ALORS CO := CO + 1 FSI
  FIN
```

Le système décrit ci-dessus correspondra à la réalisation matérielle suivante :



La syntaxe des langages de programmation s'est avérée insuffisante pour la description des systèmes complexes ; ce qui a mené à l'extension syntaxique de ces langages.

Par exemple, le langage IRENE, qui est basé sur les notations de PASCAL, utilise de nouvelles primitives syntaxiques afin de décrire plus finement le comportement de système logique. C'est ainsi que pour décrire les conditions dont le système tient compte d'une manière permanente, le langage de description IRENE utilise le mot-clef **toutes les fois que** [JAH].

La multitude de mots-clefs introduits ainsi dans les langages procéduraux est le résultat de la non-décomposition des systèmes logiques en plusieurs niveaux systémiques.

C'est ainsi que nous avons opté pour un langage de description non procédural qui présente les caractéristiques de base des REMUN, à savoir la décomposition fonctionnelle en niveaux systémiques et la hiérarchie structurelle au sens des REMUN.

L'introduction de nouveaux symbolismes ou de nouvelles primitives a été largement critiquée par LIPOVSKI, du fait que :

"Si chacun parle son propre langage,
qui va écouter l'autre" [LIP].

C'est la raison pour laquelle nous avons largement utilisé les symbolismes d'un langage de description, à savoir CDL introduit par Y. CHU en 1974 [CHU], pour la définition de LIDO dont les principaux objectifs sont cités ci-dessous.

III. OBJECTIFS DU LANGAGE LIDO

Les objectifs visés pour la définition de LIDO sont de deux ordres :

- le premier est syntaxique ; à partir de l'analyse syntaxique d'une description, on peut extraire des informations concernant la structure du système décrit.

- le second est sémantique ; à partir de l'analyse sémantique, on extrait des informations concernant le comportement du système décrit.

Ces deux ordres sont illustrés dans la figure E.1.

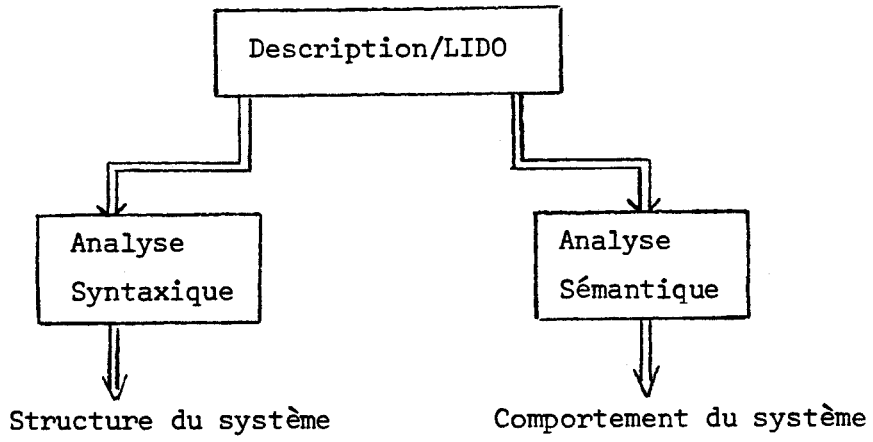


Figure E.1. : Extraction structurelle et comportementale.

Nous citerons ci-après les objectifs que nous avons visés qui sont d'ordre syntaxique et d'ordre sémantique.

III.1. Objectifs d'ordre syntaxique

La syntaxe de LIDO doit présenter les qualités suivantes :

- La simplicité de manipulation des objets.

- La clarté d'une description : La caractéristique non procédurale de LIDO a pour but l'écriture des opérations élémentaires en un ordre facilitant la lisibilité de la description.

- La flexibilité : Les règles de composition des objets doivent être souples, offrant à l'utilisateur la possibilité de définir plusieurs modèles architecturaux de systèmes logiques.

III.2. Objectifs d'ordre sémantique

La sémantique de LIDO doit présenter les qualités suivantes :

- La naturalité des objets manipulés par LIDO qui doivent correspondre aux objets réels du matériel.

- La facilité d'interprétation d'une description de LIDO en LREMUN, afin d'effectuer la simulation fonctionnelle.

Un système logique est décrit à l'aide de LIDO, en spécifiant les mécanismes qui le composent. L'ensemble de ces mécanismes est présenté dans le paragraphe suivant.

IV. LES MECANISMES DE LIDO

Un mécanisme est une partie du système (appelé aussi : niveau systémique) constituée d'objets qui peuvent être combinés entre-eux ou non.

Le langage LIDO présente d'une manière hiérarchique les mécanismes suivants :

- Mécanisme d'horloge.
- Mécanisme de conditions.
- Mécanisme séquenceur.
- Mécanisme décodeur.
- Mécanisme d'opérations.

Chacun de ces mécanismes a un rôle fonctionnel qui lui est propre.

IV.1. Mécanisme d'horloge

Le mécanisme d'horloge représente un nombre d'états fonctionnels, égal au nombre de phases d'horloge. A un instant donné, un seul état fonctionnel de l'horloge est actif.

Ce mécanisme a pour fonction, la production d'un flot d'activation pour synchroniser les autres mécanismes.

La description de ce mécanisme spécifie le nombre n de phases horloge, de la façon suivante :

"HORLOGE" n ;

Exemple :

"HORLOGE" 2 ;

Cette description spécifie une horloge biphasée, engendrant cycliquement les phases PHI1 et PHI2.

IV.2. Mécanisme de conditions

Ce mécanisme permet de décrire le vecteur d'indications de l'état interne du système. Les composantes de ce vecteur permettent l'activation structurelle du niveau séquenceur, elles correspondront alors à des conditions de transitions.

Le mécanisme de conditions définit alors le compte-rendu de l'exécution d'une opération élémentaire.

IV.3. Mécanisme séquenceur

Ce mécanisme représente, d'une manière abstraite, le séquencement du système logique, à l'aide d'un graphe de transition. Il génère un flot d'activation des opérations élémentaires décrites dans le mécanisme d'opérations.

Le mécanisme séquenceur décrit le comportement du système logique, en précisant les états fonctionnels, les conditions de transition entre ces états et le flot d'activation des opérations élémentaires.

Ce mécanisme doit permettre l'activation simultanée de plusieurs opérations.

IV.4. Le mécanisme décodeur

Le flot d'activation généré par le mécanisme séquenceur peut être codé ; le mécanisme décodeur a pour fonction le décodage de ce flot, puis l'activation proprement dite des opérations élémentaires.

IV.5. Le mécanisme d'opérations

Le mécanisme d'opérations définit toutes les opérations élémentaires de traitement des données.

Trois types d'opérations seront distinguées :

- opération de transfert,
- opération arithmétique,
- opération logique.

L'exécution d'une opération élémentaire correspond à un traitement de contenu des éléments de mémorisation que nous introduisons ci-dessous.

V. ELEMENTS DE MEMORISATION

Un élément de mémorisation est un objet porteur d'informations, contenant une valeur.

En LIDO, tout élément de mémorisation doit être déclaré avant son utilisation dans la description.

Deux types d'éléments de mémorisation seront distingués :

- les registres,
- les mémoires.

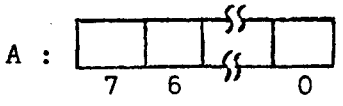
V.1. Les registres

Tout registre est nommé, et défini comme un intervalle de bits en précisant ses deux bornes.

Exemple de déclaration de registres :

```
"REGISTRE"
     A(7 : 0) ; B(7 : 0) ;
     RI(7 : 0) ; RA(15 : 0) ; CO(15 : 0) ;
```

A est le nom d'un registre de longueur 8 avec 7 pour borne supérieure et 0 pour borne inférieure.



Il est possible d'utiliser une partie d'un registre, appelée sous-registre.

Les sous-registres doivent être déclarer après leurs registres primaires.

Exemple de déclaration de sous-registres :

```
"SOUS-REGISTRE"
    COH = CO(15 : 8) ;
    COB = CO(7 : 0) ;
    COP = RI(7 : 3) ;
```

Le contenu initial d'un registre est précisé juste après sa déclaration, entre %...% sinon il est considéré comme nul.

Exemple :

```
"REGISTRE"
   A(7 : 0) ; CO(15 : 0) ;  % 0000 0000 0001 0000 B %
```

Le contenu initial de CO est donné ici en binaire ; il est possible aussi de le préciser en hexa-décimal % 10 H % ou en décimal % 16 D %.

V.2. Mémoire

Une mémoire est déclarée comme un tableau à deux dimensions en déterminant la taille des mots, l'adresse du début et l'adresse de fin.

Exemple de description de mémoires :

```
"MEMOIRE"
   RAM(0 : 255, 8) ;
   ROM(156 : 1023, 8) ;
```

VI. DESCRIPTION DES MECANISMES

VI.1. Mécanisme d'horloge

La description du mécanisme d'horloge consiste a spécifier le nombre n de phases d'horloge, qui seront nommées dans la description PHI1, PHI2,..., PHIn.

Cette description sera définie ainsi :

"HORLOGE" n ;

Exemple :

"HORLOGE" 2 ;

Cet exemple correspond à une horloge bi-phasée.

VI.2. Mécanisme de conditions

A chaque condition sera associée une expression booléenne, de la façon suivante :

"CONDITION"
nom-condition 1 : expression booléenne 1 ;
nom-condition 2 : expression booléenne 2 ;
⋮
nom-condition i : expression booléenne i ;
⋮

Les expressions booléennes sont décrites à l'aide de notations de comparaisons données dans le tableau suivant.

<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
<>	différent de
=	égal à

Comparateurs de LIDO.

Exemple de description de mécanisme de conditions :

```
"CONDITION"
  C1 : A > B ;
  C2 : B(0) = 1 B ;
```

On peut effectuer des tests, même au niveau du bit, comme le montre la condition C2 qui teste l'égalité du bit de poids faible de au 1 Booléen.

VI.3. Mécanisme séquenceur

La description du mécanisme séquenceur correspond à la représentation textuelle d'un automate. Une transition d'un état à un autre peut être conditionnée ou non par les composantes du mécanisme de conditions.

Deux types de transitions peuvent alors être décrites :

- transition conditionnée :

```
NS : /C/ S1, S2,..., SN - NO ;
```

- transition non conditionnée :

```
NS : S1, S2,..., SN - NO ;
```

avec NS : Noeud Source de la transition,

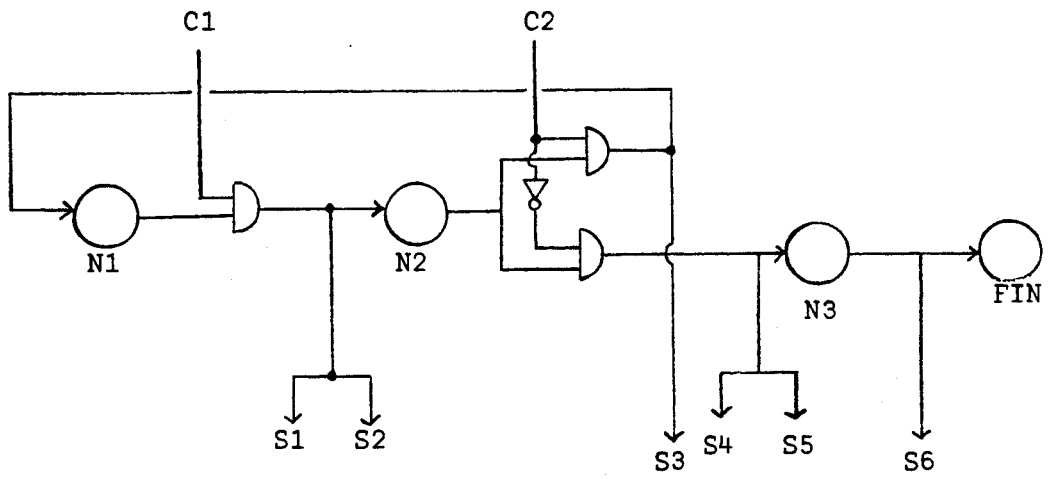
C : Une condition générée par le mécanisme des conditions,

S1, S2,..., SN : Flot d'activation des opérations élémentaires,

NO : Noeud Objet de la transition.

Exemple :

Soit l'automate de MEALY décrit graphiquement ci-dessous.



La description de ce séquenceur présentera quatre états N1, N2, N3 et FIN ; ce dernier est l'état final qui correspondra au mot clé "FIN". Cette description sera textuellement définie à l'aide de LIDO de la façon suivante.

```
"SEQUENCEUR"
   N1 : /C1/ S1, S2 - N2.
   N2 : /C2/ S3 - N1 ;
        /C2'/ S4, S5 - N3.
   N3 : S6 - "FIN".
```

' : est le symbole d'un inverseur.

VI.4. Mécanisme décodeur

Le mécanisme décodeur traite le contenu d'un registre R, en associant aux combinaisons binaires de R des sorties.

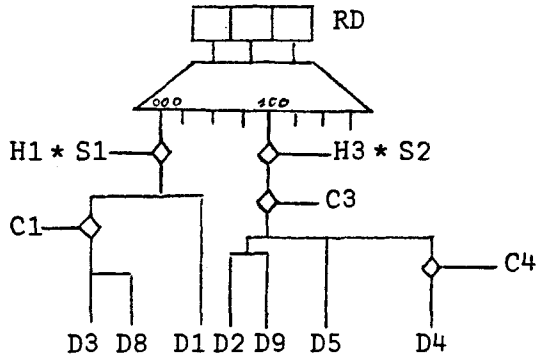
Les sorties du décodeur peuvent être conditionnées par le flot généré par les autres mécanismes.

Pour chaque flot généré par le mécanisme décodeur, on doit spécifier :

- le flot de condition,
- le contenu du registre.

Exemple :

Soit le système de décodage suivant :



La description textuelle de ce décodeur à l'aide de LIDO est ainsi définie :

```
"DECODEUR" RD(2 : 0) ;
    H1 * S1 : (000) /C1/ (D3, D8), D1 ;
    H3 * S2 : (100) /C3/ (D2, D9), D5, /C4/ D4 ;
```

VI.5. Mécanisme d'opérations

Ce mécanisme associe au flot généré par les mécanismes séquenceur et décodeur, les opérations élémentaires qui sont de trois types. On donnera les opérateurs de base de LIDO.

- Opérateur de transfert, noté ainsi <—, permet de transférer le contenu d'un élément de mémorisation à un autre.

- Opérateurs arithmétiques : les différentes notations de ces opérateurs figurent dans le tableau ci-dessous.

@ +	addition arithmétique
@ -	soustraction arithmétique
@ *	multiplication arithmétique
@ /	division arithmétique

Les opérateurs arithmétiques sont précédés de @ pour ne pas les confondre avec les opérateurs logiques.

- Opérateurs logiques : les différentes notations de ces opérateurs figurent dans le tableau ci-dessous.

+	ou logique
*	et logique
,	inverseur
-	ou exclusif
/	traitement d'une chaîne de bits

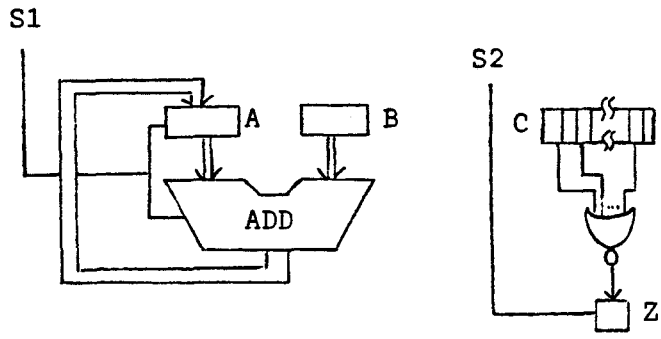
Il est possible d'effectuer des combinaisons de ces opérateurs afin de décrire de nouveaux opérateurs tels que le NAND ou le NOR...

+' : NOR

*' : NAND,... etc...

Exemple de description du mécanisme d'opérations :

Soit un chemin de données représenté ci-dessous.



La description textuelle, de cette partie du système, à l'aide de LIDO est ainsi définie :

```
"OPERATION"
    S1 : A ← A @ + B ;
    S2 : Z ← /+' C ;
```

La syntaxe du langage LIDO est présentée en annexe sous la forme de BACKUS-NAUR (BNF).

VII. INTERCONNEXION DES MECANISMES

Dans ce paragraphe, on va s'intéresser aux connexions structurelles entre les différents mécanismes que peut comporter un système logique.

Une connexion, entre deux mécanismes M1 et M2, peut être sous l'une des trois formes suivantes :

- émétrice (M1 → M2),
- réceptrice (M1 ← M2),
- ou bidirectionnelle (M1 ↔ M2).

Le rôle d'une connexion est de transférer un flot d'informations d'un mécanisme à un autre, suivant l'un des sens cités ci-dessous.

Pour tout système logique on distinguera les trois types de flots suivants :

- Flot d'activation : a pour but l'activation d'une opération élémentaire.

Les mécanismes sources de ce flot sont :

- le mécanisme d'horloge,
- le mécanisme séquenceur,
- le mécanisme décodeur.

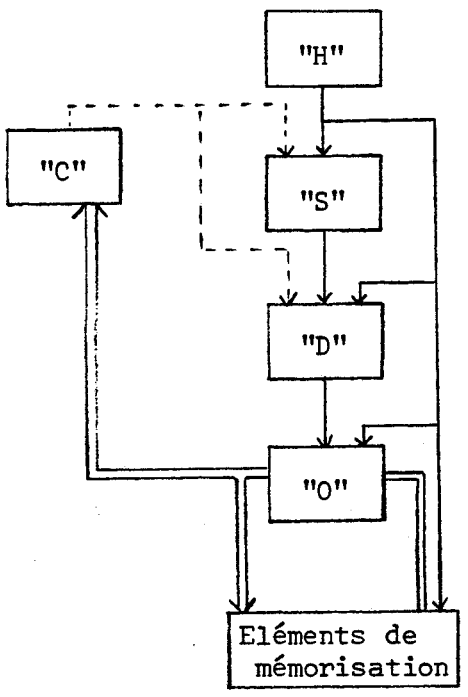
- Flot de conditions détermine l'état conditionnel du système d'une manière instantannée et active les transitions entre les états fonctionnels du mécanisme séquenceur.

Le mécanisme source de ce flot est le mécanisme de conditions.

- Flot de données : traité par le mécanisme d'opérations. Ce traitement correspond à l'exécution proprement dite d'une opération élémentaire.

Ce sont les éléments de mémorisation (registres et mémoires) qui sont sources du flot de données.

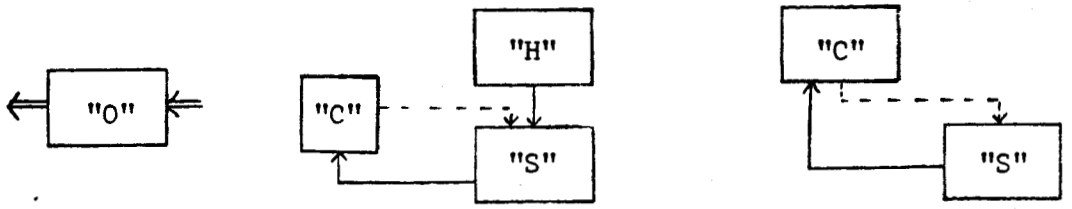
Les connexions entre les mécanismes sont illustrées dans le schéma E.2.



"H" : Mécanisme d'horloge.
"C" : Mécanisme de conditions.
"S" : Mécanisme séquenceur.
"D" : Mécanisme décodeur.
"O" : Mécanisme d'opérations.
——— : Flot d'activation.
- - - : Flot de condition.
═══ : Flot de données.

Schéma E.2. : Interconnexion des mécanismes.

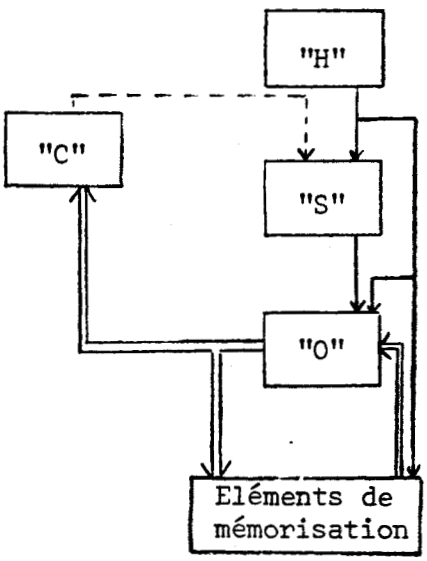
Le nombre de mécanismes, que peut comporter un système logique, est compris entre un et cinq mécanismes ; comme le montrent les schémas partiels de la figure ci-dessous.



a. Circuit combinatoire.

b. Circuit séquentiel synchrone.

c. Circuit séquentiel asynchrone.



d. Système logique sans décodeur

Schéma E.3. : Combinaisons de mécanismes.

VIII. EXEMPLE DE DESCRIPTION

Le système logique, qui sera décrit à l'aide de LIDO, est un diviseur. Il sera détaillé au fur et à mesure de la présentation des différents modules du simulateur.

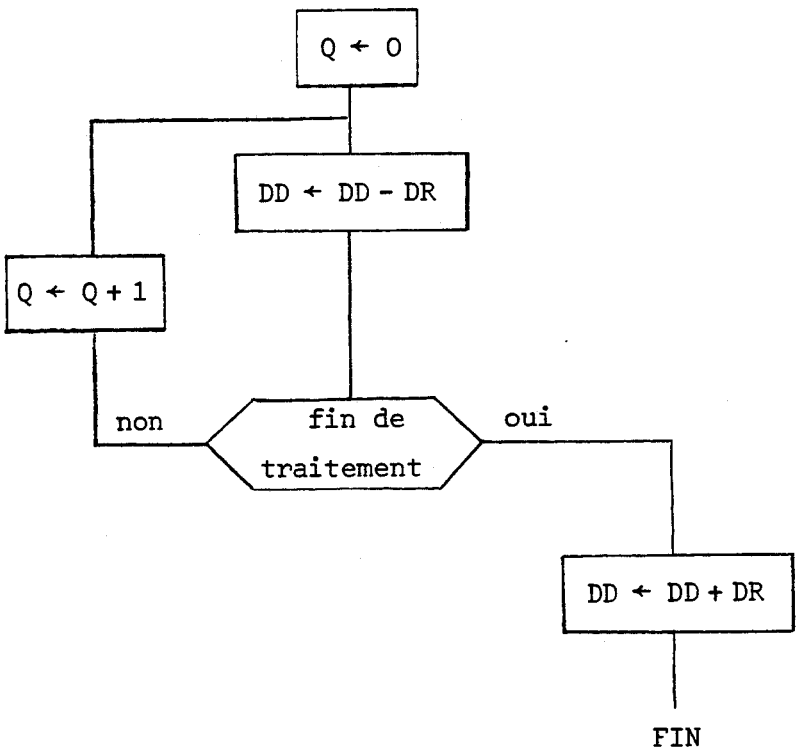
Le diviseur présenté contiendra six registres, dont trois à contenu variable (DD : Dividande, DR : Diviseur et Q : Quotient) et trois autres à contenu constant (CUN, CZERO et CTEST).

Le mécanisme d'horloge spécifiera deux phases.

Deux conditions seront distinguées :

- CF : Condition de Fin.
- NONCF : l'inverse de CF.

Le mécanisme séquenceur décrira l'organigramme suivant.



Les registres sont initialement chargés après leurs déclarations

Le mécanisme d'opérations définit toutes les opérations qu'on trouve dans l'organigramme ci-dessus, à savoir :

Q ← 0 qui sera nommé SI,
DD ← DD - DR qui sera nommée SS,
DD ← DD + DR qui sera nommée SA,
et Q ← Q + 1 qui sera nommée SC.

La description de ce diviseur sera la suivante :

```
%   déclaration  des  registres
 à contenu constant ou variable %

"REGISTRE"

  DD(8:0);     % 000001101 B %
  DR(7:0);     %  00000011 B %
  Q(7:0);      %  00000000 B %
  CUN(7:0);    %  00000001 B %
  CZERO(7:0); %  00000000 B %
  CTEST(7:0); %  10000000 B %


%  phases d'horloge  %

"HORLOGE" 2 ;


%  définition des conditions  %

"CONDITION"

  CF : CTEST ( DD;
  NONCF : CTEST )= DD;


%  définition du graphe de controle  %

"SEQUENCEUR"

  N1     :  SI - N2 .   % initialisation %
  N2     :  SS - N3 .
  N3     : /CF/ SA - "FIN";
           /NONCF/ SC -N2.


%  définition des opérations élémentaires  %

"OPERATION"

 SI   : Q (- CZERO;
 SA   : DD (- DD @+ DR ;
 SS   : DD (- DD @- DR ;
 SC   : Q (- Q @+ CUN ;
$
```

Le mécanisme séquenceur défini dans cet exemple génère directement le flot d'activation des opérations élémentaires, ce flot n'est pas alors codé, c'est la raison pour laquelle le mécanisme décodeur n'y est pas défini.

Pour toute description de système logique, en LIDO, est associée un REMUN permettant essentiellement la modélisation de son contrôle.

Nous présentons ci-après le passage d'une description de LIDO en LREMUN.

IX. INTERPRETATION DE LIDO EN LREMUN

La description en LREMUN associée à un système logique donné consiste à modéliser les différents mécanismes de contrôle suivants :

- Mécanisme d'horloge.
- Mécanisme de conditions.
- Mécanisme séquenceur.

A chacun de ces mécanismes sera associé un niveau formant ainsi un réseau à trois niveaux (figure E.3.). Le REMUN obtenu détermine les opérations élémentaires à activer pour chaque étape de simulation.

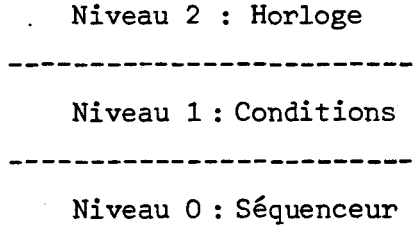


Figure E.3. : Association du REMUN à un système logique.

Nous montrons dans la suite le passage de chacun de ces mécanismes de LIDO en niveaux et leurs descriptions en LREMUN.

IX.1. Niveau d'horloge

Soit la description du mécanisme d'horloge à n phases :

"HORLOGE" n ;

Le niveau 2, définissant cette description, sera graphiquement représenté ainsi :

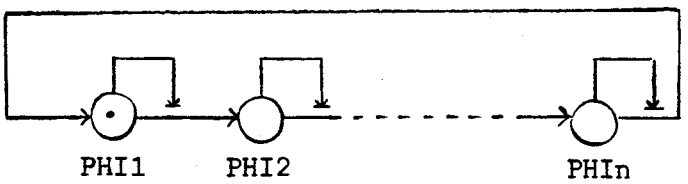


Figure E.4. : Représentation graphique du niveau d'horloge.

Le niveau d'horloge présente alors :

- n places, nommés PHI1, PHI2,..., PHIn
- n transitions de contenu telles que :
  pour i $\in$ [1, n] on a : $tc_i : ec(PHI_i) \rightarrow ec(PHI_{(i+1) \bmod n})$
- n transitions contenu-structure telles que :
  $tcs_i : ec_i \rightarrow (PHI_i, PHI_{(i+1) \bmod n})$ avec $ec_i = ec(PHI_i)$

IX.2. Niveau de conditions

Chaque condition C_i est définie par une expression booléenne E_i.

Le passage du mécanisme de conditions, décrit en LIDO, au niveau 1 du REMUN, consiste à associer à chaque condition C_i une place, nommée C_i, dont l'état de contenu à chaque étape de simulation est déterminée par son expression booléenne E_i, de la façon suivante :

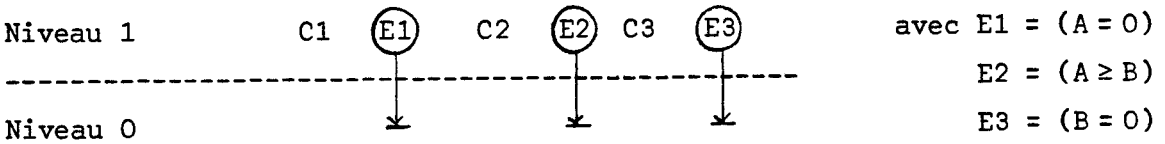
$$ec(C_i) = E_i.$$

Les transitions contenu-structure servent à activer le niveau subordonné 0 permettant ainsi les transitions de contenu d'un état fonctionnel du mécanisme séquenceur à un autre état. Ces transitions contenu-structure seront donc spécifiées par le mécanisme séquenceur.

Exemple :

"CONDITION"
C1 : A = 0 ;
C2 : A >= B ;
C3 : B = 0 ;

Le niveau 1 sera ainsi défini :



IX.3. Niveau séquenceur

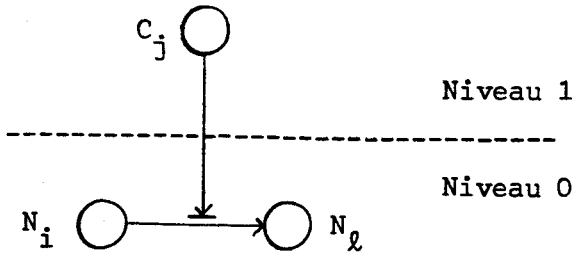
Le mécanisme séquenceur est défini par un ensemble d'états fonctionnels, dont les transitions entre-eux sont activées par les mécanismes d'horloge et/ou de conditions. Chacune de ces transitions permet l'activation parallèle d'un ensemble d'opérations élémentaires.

Le passage du mécanisme séquenceur, décrit en LIDO, au niveau 0 du REMUN, consiste à associer à chaque état fonctionnel N_i une place nommée N_i, les transitions de contenu ainsi que leurs activations sont déterminées de la façon suivante :

Soit la description définie ci-dessous :

N_i : /C_j/ S_k - N_ℓ.

Le passage au REMUN sera ainsi graphiquement défini :



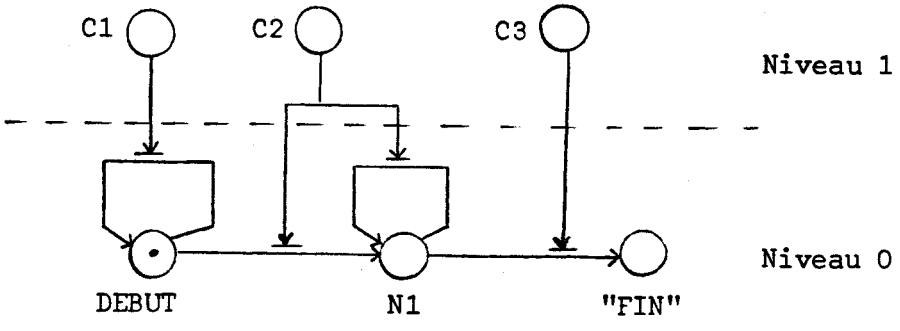
L'opération élémentaire S_k s'activera si $ec(N_i) = 1$ et $ec(C_j) = 1$, elle sera alors associée à N_i et C_j.

Exemple :

Soit la description du mécanisme séquenceur suivant :

```
"SEQUENCEUR"
    DEBUT : /C2/ S1, S2 - N1 ;
            /C1/ S3 - DEBUT.
       N1 : /C2/ S2 - N1 ;
            /C3/ S4 - "FIN".
```

Dans cette description, trois états fonctionnels sont définis (DEBUT, N1, et "FIN"). Le niveau 0 sera alors formé de trois places, ainsi :



Les associations (Places ; opérations élémentaires) sont les suivantes :

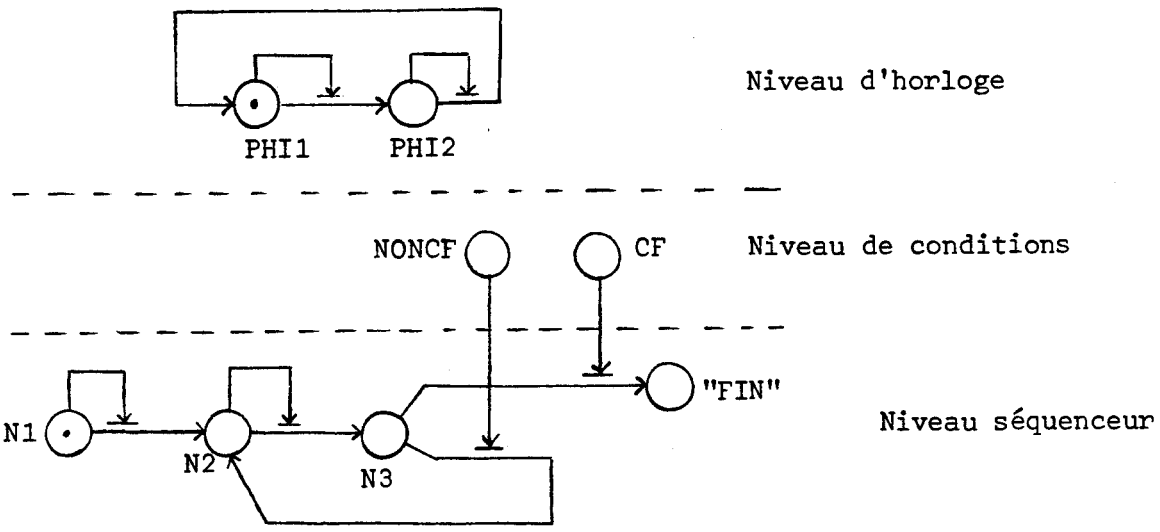
(DEBUT, C2 ; S1, S2)
(DEBUT, C1 ; S3)
(N1, C2 ; S2)
(N1, C3 ; S4)

La première association indique l'activation simultanée des opérations élémentaires S1 et S2 si on a :

ec(DEBUT) = ec(C2) = 1.

IX.4. Exemple

Nous présentons le REMUN associé au diviseur décrit au paragraphe 8.



avec

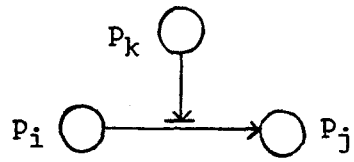
ec(CF) = (CTEST < DD),
ec(NONCF) = (CTEST >= DD).

Les associations (places ; opérations élémentaires) sont les suivantes :

(N1 ; SI)
(N2 ; SS)
(N3, CF ; SA)
(N3, NONCF ; SC)

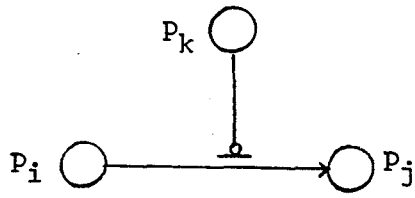
Remarque : La complémentarité que présente les places CF et NONCF conduit à l'extension des REMUN au sens de l'activation positive-négative des états structurels.

- Activation positive :



La transition de contenu (p_i, p_j) est à l'état actif si $ec(p_k) = 1$.

- Activation négative :

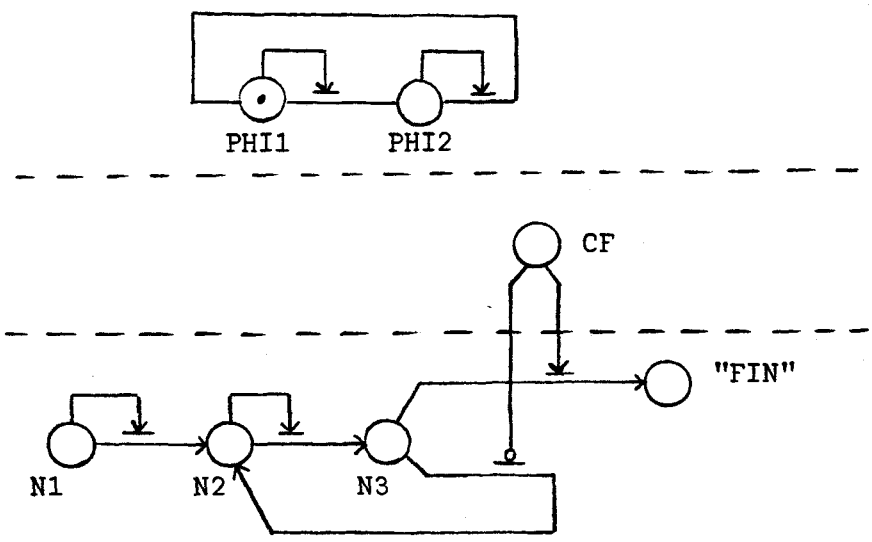


: Représentation de l'activation négative.

La transition de contenu (p_i, p_j) est à l'état actif si $ec(p_k) = 0$.

La notion de contrôle positif - négatif est utilisée dans le langage de description NEGLAN [BAK].

Pour les REMUN, cette notion permet la suppression des places complémentaires, ce qui est le cas pour CF et NONCF dans l'exemple. Le REMUN à activation positive-négative qui correspond à cet exemple est le suivant.



Les associations (places ; opérations élémentaires) sont les suivantes :

(N1 ; SI)
(N2 ; SS)
(N3, CF ; SA)
(N3, -CF ; SC)

La notation - représente l'activation négative.

X. PASSAGE DU REMUN AU NIVEAU LOGIQUE

La représentation d'un système logique au niveau logique consiste à définir le nombre d'éléments de mémorisation et de portes logiques qu'il comporte puis établir les connexions entre-eux.

Le passage du niveau de description transfert de registre au niveau électrique se fera par l'intermédiaire du REMUN généré à partir de la description en LIDO.

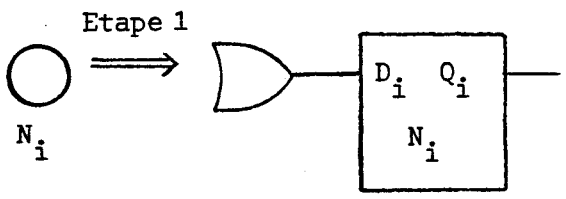
Ceci consistera alors à transcrire matériellement un REMUN en terme de primitives logiques.

X.1. Réalisation cablée

Nous présentons une méthode simple qui s'établit en cinq étapes.

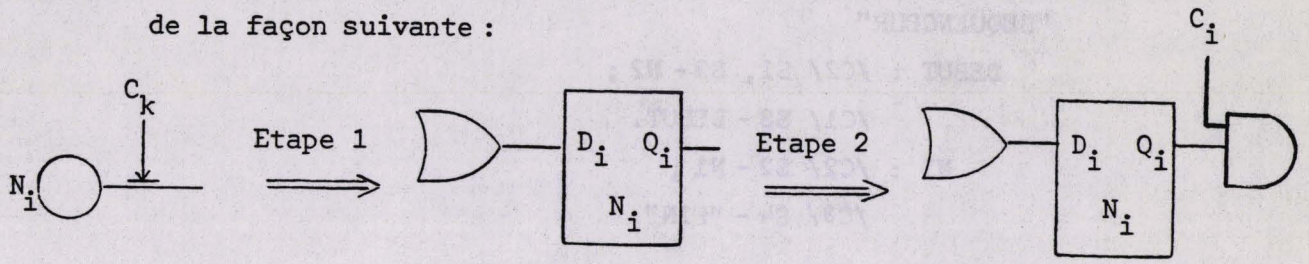
Etape 1 : Association d'un élément de mémorisation et d'une porte "OU" à chaque place du séquenceur.

L'élément de mémorisation peut être une bascule D, connectée à la porte "OU" ainsi :



Le système est défini ainsi d'une manière semi-structurelle, sans spécification de connexions entre ses primitives.

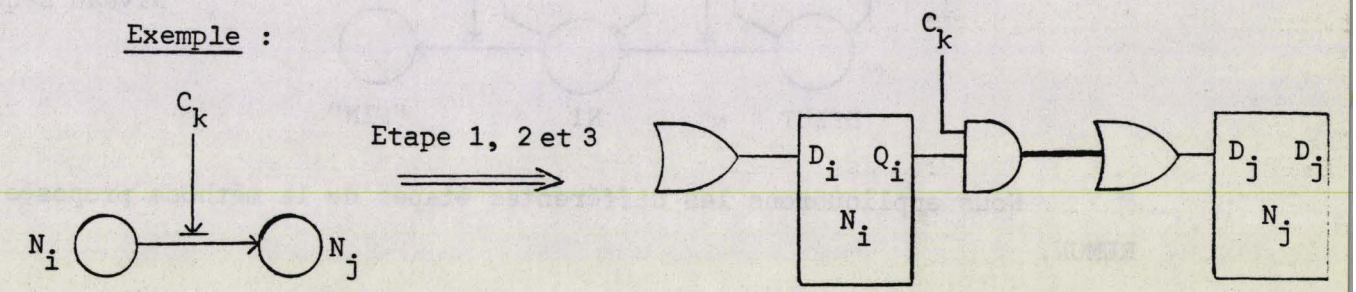
Etape 2 : Association d'une porte "ET" à toute transition contenu-structure, de la façon suivante :



Etape 3 : Etablissement matériel des transitions de contenu.

Il s'agit d'associer matériellement à une transition de contenu une connexion, comme ce qui est représenté en gras ci-dessous.

Exemple :



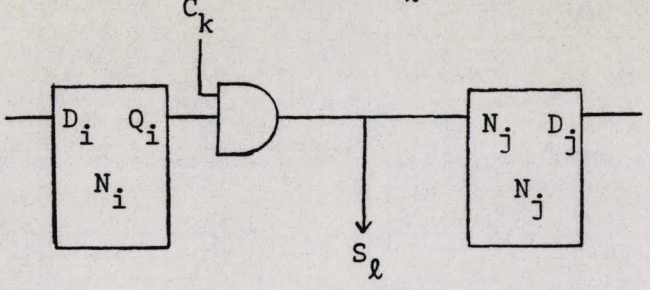
Etape 4 : Elimination de portes logiques à une seule entrée.

Etape 5 : Spécification des sorties S_i du séquenceur à l'aide de la table (Places ; opérations élémentaires).

Exemple :

Soit le triplet $(N_i, C_k ; S_\ell)$ contenu dans la table.

La sortie d'activation S_ℓ sera défini par la porte "ET", ainsi :

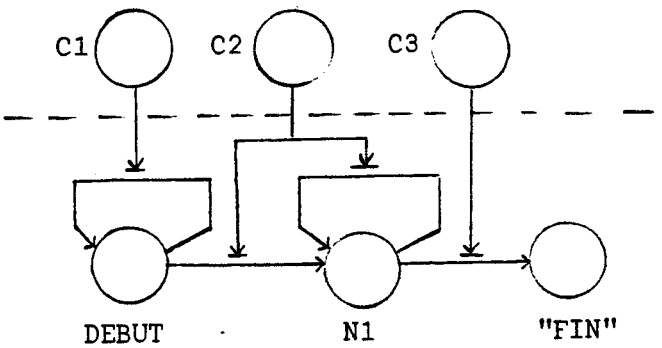


X.2. Exemple

Nous reprenons l'exemple du séquenceur décrit au paragraphe IX.3.

```
"SEQUENCEUR"
   DEBUT : /C2/ S1, S2 - N2 ;
           /C1/ S3 - DEBUT.
      N1 : /C2/ S2 - N1 ;
           /C3/ S4 - "FIN".
```

Le REMUN généré pour cette description était le suivant :

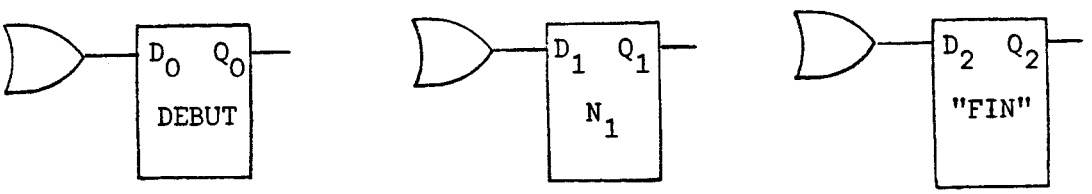


Niveau de conditions

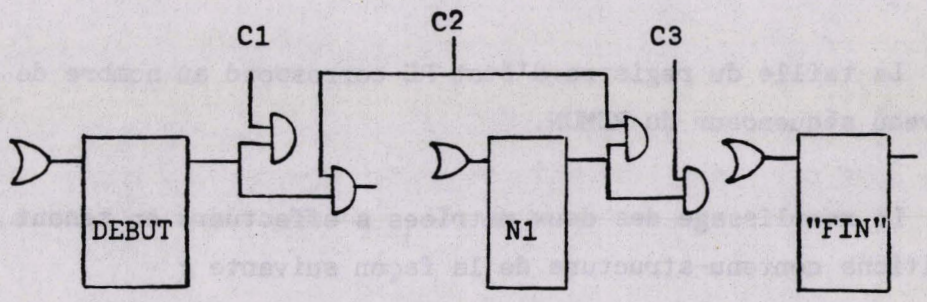
Niveau séquenceur

Nous appliquerons les différentes étapes de la méthode proposée à ce REMUN.

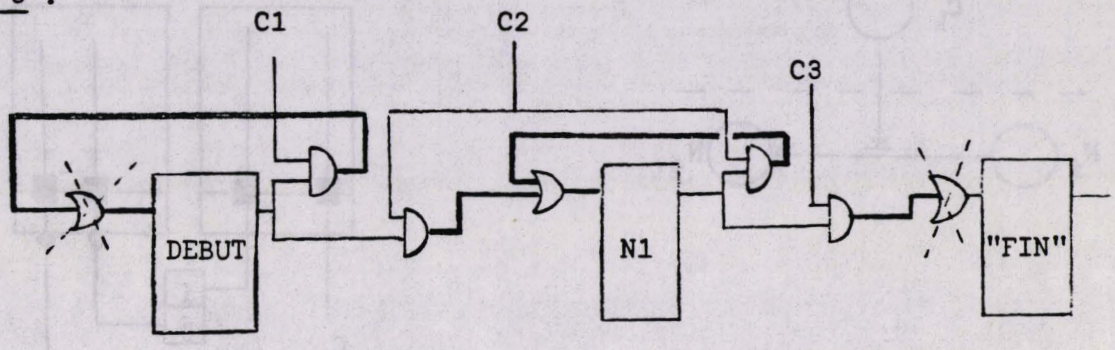
Etape 1 :



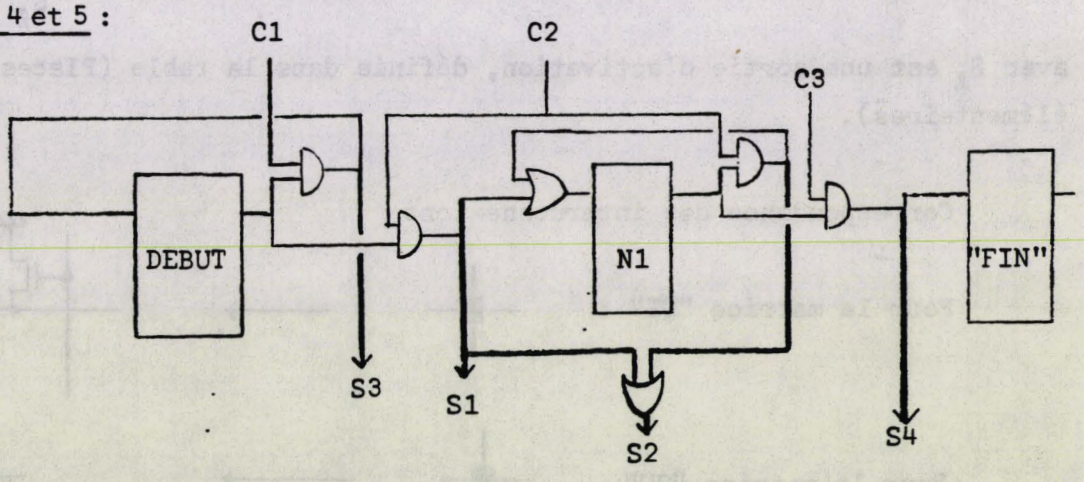
Etape 2 :



Etape 3 :

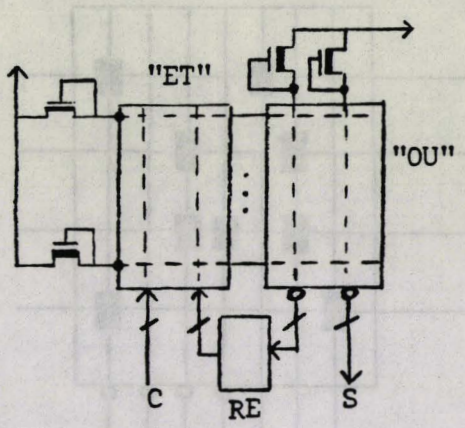


Etapes 4 et 5 :



X.3. Réalisation à l'aide de PLA

La réalisation à l'aide de PLA consiste à définir le contenu des matrices "ET" et "OU", elle est organisée ainsi :



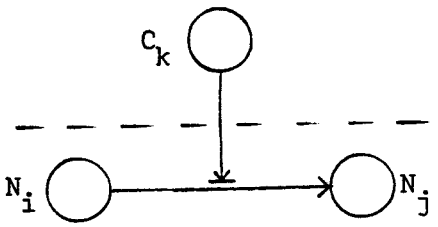
où C : ensemble de conditions,

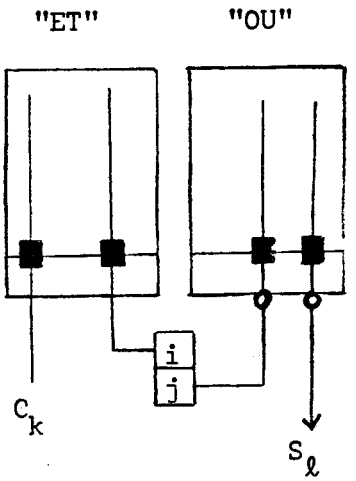
S : ensemble de sorties,

RE : Registre d'Etat.

La taille du registre d'état RE correspond au nombre de places dans le niveau séquenceur du REMUN.

Le remplissage des deux matrices s'effectuera en tenant compte des transitions contenu-structure de la façon suivante :

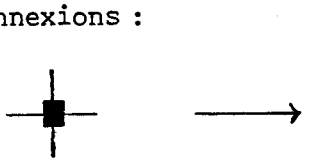




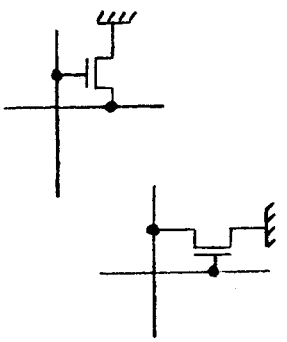
avec S_ℓ est une sortie d'activation, définie dans la table (Places ; opérations élémentaires).

Correspondance des interconnexions :

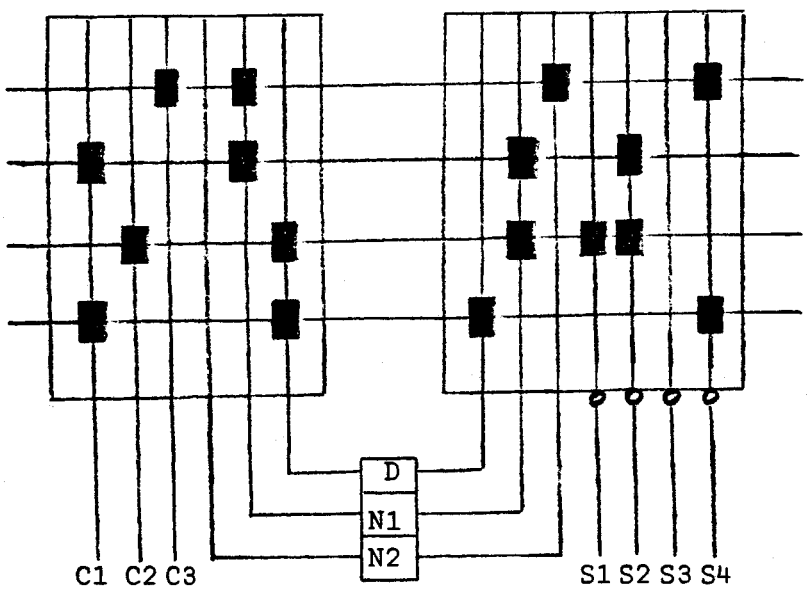
Pour la matrice "ET" :



Pour la matrice "OU" :

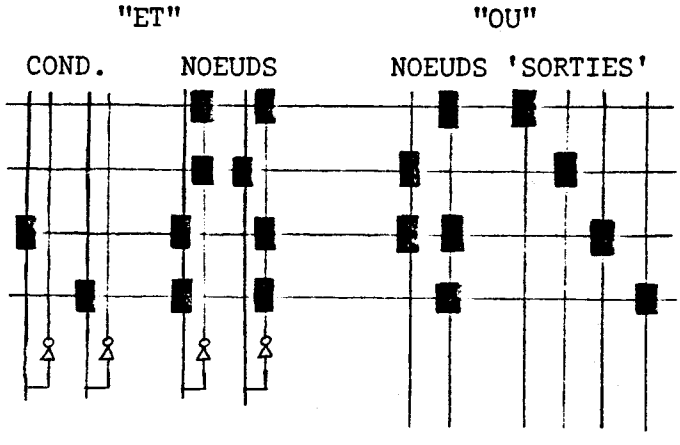


Nous reprenons l'exemple de la réalisation cablée.



Un générateur automatique de PLA a été réalisé, le problème qui est posé est celui de minimisation de matrices.

On présente dans la figure ci-après un exemple de génération automatique des matrices "ET" et "OU".



XI. SIMULATEUR LIDO

La simulation d'un système logique, décrit à l'aide de LIDO, a pour but la vérification fonctionnelle de ce système. Le simulateur LIDO est composé des modules suivants :

- Analyseur syntaxique.
- Détecteur d'erreurs conceptuelles.
- Interpréteur : réalisant les fonctions suivantes :
  - . génération de la description en LREMUN,
  - . génération de tables de contenu,
  - . génération de la table d'association (Places ; opérations).
- Simulateur de REMUN défini au chapitre précédent.
- Ensemble de modules d'exécution d'opérateurs élémentaires.

L'enchaînement de ces modules est présenté dans la figure E.5.

XII. EXEMPLE DE SIMULATION

L'exemple exposé ici correspond au diviseur décrit auparavant.

Les résultats de la simulation se décomposent en cinq parties :

1. Génération des tables de contenu des éléments de mémorisation.
2. Description et modélisation du REMUN.
3. Association (Places actives ; opérations élémentaires).
4. Liste des opérations élémentaires.
5. Résultats de la simulation proprement dite.

Chaque étape de simulation comprend :

- la génération des places actives,
- l'activation des opérations élémentaires,
- l'impression des contenus des éléments de mémorisation.

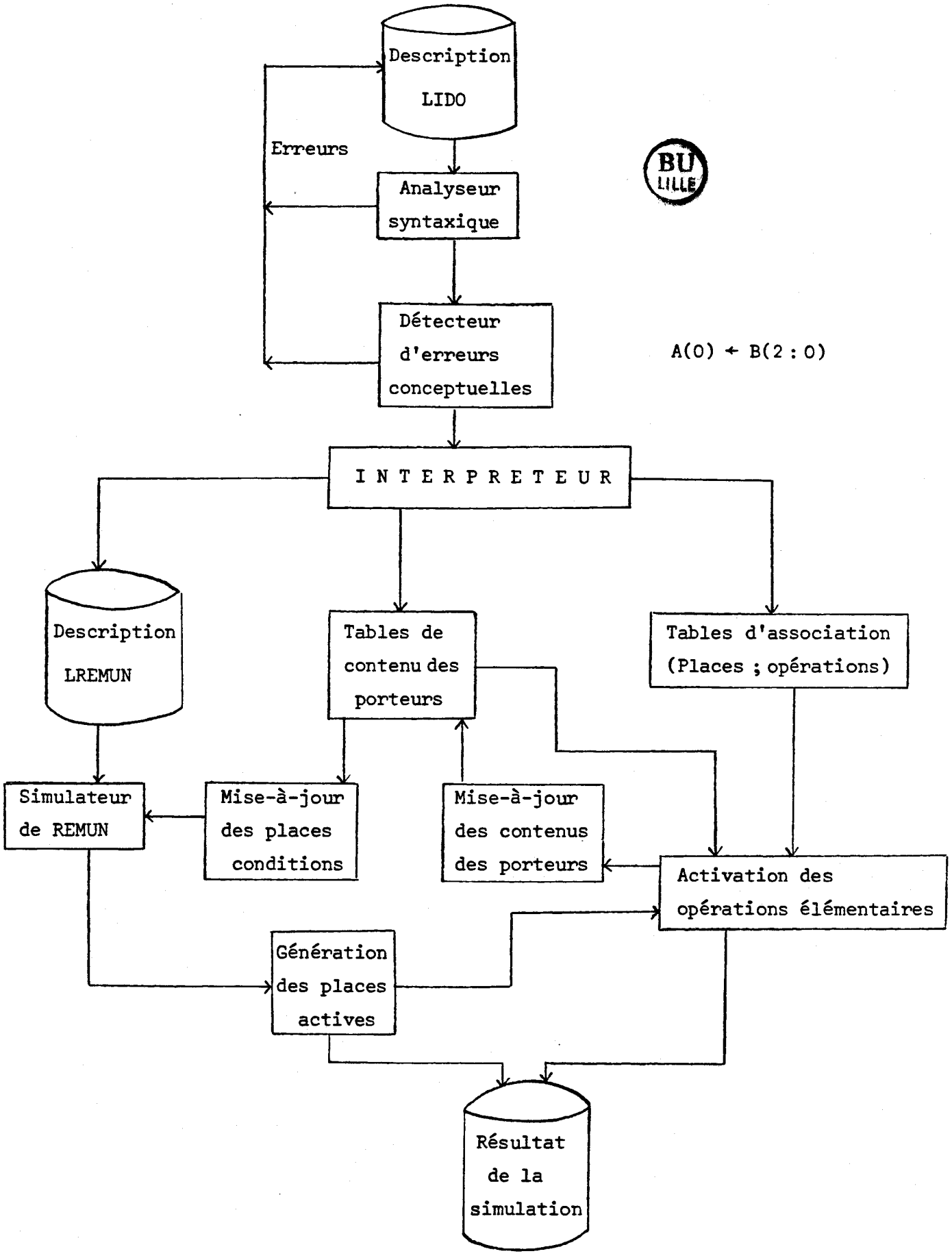


Figure E.5. : Enchaînement des modules du simulateur.

```
GENERATION  DES TABLES  DE  CONTENU
   DES  ELEMENTS DE MEMORISATION
==================================

DD       000001011
DR        00000011
Q         00000000
CUN       00000001
CZERO     00000000
CTEST     10000000


 DESCRIPTION DU REMUN
 ====================

DEBREMUN
  DEBNIV  2
    DEBPLACE  1  1  PHI1
        C  2  2
        S  2  1  2  2
    FPLACE
    DEBPLACE  2  0  PHI2
        C  2  1
        S  2  2  2  1
    FPLACE
  FNIV
  DEBNIV  1
    DEBPLACE  3  0  CF     S  0  7  0  0  FPLACE
    DEBPLACE  4  0  NONCF  S  0  7  0  6  FPLACE
  FNIV
  DEBNIV  0
    DEBPLACE  5  1  N1
        C  0  6
        S  0  5  0  6
    FPLACE
    DEBPLACE  6  0  N2
        C  0  7
        S  0  6  0  7
    FPLACE
    DEBPLACE  7  0  N3  C  0  0  FPLACE
    DEBPLACE  8  0  "FIN"  FPLACE
  FNIV
FREMUN

ASSOCIATION  (PLACES ACTIVES ; OPERATIONS ELEMENTAIRES) :
========================================================

(N1 ; SI)
(N2 ; SS)
(N3,CF ; SA)
(N3,NONCF ; SC)
("FIN" ; )


LISTE DES OPERATIONS ELEMENTAIRES :
===================================

OP. :  RES. <- OPD1    NOM-OP. OPD2

SI      Q       CZERO
SA      DD      DD      é+      DR
SS      DD      DD      é-      DR
SC      Q       Q       é+      CUN
```

MODELISATION DU REMUN

NIVEAU	NOEUD	NOM	EC	LIAIS. CONT-CONT.			LIAIS. CONT-STRUCT.			
				NIVEAU	NOEUD	ACTIVE	NIVEAU	NOEUD	NIVEAU	NOEUD
2										
	1	PHI1	T							
				2	2	F				
							2	1	2	2
	2	PHI2	F							
				2	1	F				
							2	2	2	1
1										
	3	CF	F							
							0	7	0	0
	4	NONCF	F							
							0	7	0	6
0										
	5	N1	T							
				0	6	F				
							0	5	0	6
	6	N2	F							
				0	7	F				
							0	6	0	7
	7	N3	F							
				0	0	F				
				0	6	F				
	8	"FIN"	F							

```
**********************************************************
************  DEBUT  DE  LA  SIMULATION  ***************
**********   DU  SYSRTEME  LOGIQUE  DECRIT   ***********
************   EN  LANGAGE  L.I.D.O.  ******************
*****************************************VERS 28/2/85*

   CONTENUS INITIAUX
     DES REGISTRES
   -----------------

   DD       000001011 <-------- dividande, qui contiendra le reste à la fin de la
                                 simulation.
   DR       00000011  <-------- diviseur.
   Q        00000000  <-------- quotient.
   CUN      00000001
   CZERO    00000000
   CTEST    10000000

================  E T A P E  : 0  ====================

   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   X X X X     GENERATION DES PLACES ACTIVES     X X X X
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI1

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N1

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    ACTIVATION   DES    OPERATIONS      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Q      <-  CZERO


-- CONTENU  DES   REGISTRES --

DD       000001011
DR       00000011
Q        00000000
CUN      00000001
CZERO    00000000
CTEST    10000000
```

```
================  E T A P E  : 1  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     GENERATION DES PLACES ACTIVES       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI2

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N2

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION   DES    OPERATIONS      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

DD     <-  DD       @-       DR


-- CONTENU  DES   REGISTRES --

DD       000001000
DR       00000011
Q        00000000
CUN      00000001
CZERO    00000000
CTEST    10000000

================  E T A P E  : 2  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     GENERATION DES PLACES ACTIVES       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI1

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N3
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION   DES     OPERATIONS       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Q       <-  Q         e+        CUN


-- CONTENU  DES    REGISTRES --

DD        000001000
DR        00000011
Q         00000001
CUN       00000001
CZERO     00000000
CTEST     10000000

=================  E T A P E  :  3  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     GENERATION DES PLACES ACTIVES       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI2

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N2

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION   DES     OPERATIONS       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

DD      <-  DD        e-        DR


-- CONTENU  DES    REGISTRES --

DD        000000101
DR        00000011
Q         00000001
CUN       00000001
CZERO     00000000
CTEST     10000000
```

```
================  E T A P E  : 4  ================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    GENERATION DES PLACES ACTIVES      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI1

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N3

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    ACTIVATION   DES    OPERATIONS     X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Q      <-  Q       @+       CUN


-- CONTENU  DES   REGISTRES --

DD       000000101
DR       00000011
Q        00000010
CUN      00000001
CZERO    00000000
CTEST    10000000

 ================  E T A P E  : 5  ================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    GENERATION DES PLACES ACTIVES      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI2

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N2
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION   DES     OPERATIONS       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

DD      <-  DD        @-        DR


-- CONTENU  DES    REGISTRES --

DD        000000010
DR        00000011
Q         00000010
CUN       00000001
CZERO     00000000
CTEST     10000000

=================  E T A P E  : 6  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     GENERATION DES PLACES ACTIVES       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


  NIVEAU  2
 -----------
    PHI1

  NIVEAU  1
 -----------
    NONCF

  NIVEAU  0
 -----------
    N3

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION   DES     OPERATIONS       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Q         <-Q         @+        CUN


-- CONTENU  DES    REGISTRES --

DD        000000010
DR        00000011
Q         00000011
CUN       00000001
CZERO     00000000
CTEST     10000000
```

```
================  E T A P E  : 7  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    GENERATION DES PLACES ACTIVES      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


 NIVEAU  2
-----------
   PHI2

 NIVEAU  1
-----------
   NONCF

 NIVEAU  0
-----------
   N2

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    ACTIVATION   DES    OPERATIONS     X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

DD       (-DD       è-       DR


-- CONTENU  DES   REGISTRES --

DD       111111111
DR       00000011
Q        00000011
CUN      00000001
CZERO    00000000
CTEST    10000000

================  E T A P E  : 8  ==================

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X    GENERATION DES PLACES ACTIVES      X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


 NIVEAU  2
-----------
   PHI1

 NIVEAU  1
-----------
   CF

 NIVEAU  0
-----------
   N3
```

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X X X X     ACTIVATION    DES     OPERATIONS       X X X X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

DD      (-  DD        @+        DR


-- CONTENU  DES   REGISTRES --

DD        000000010 <-------le reste
DR        00000011
Q         00000011<-------le quotient
CUN       00000001
CZERO     00000000
CTEST     10000000

  NIVEAU  2
 -----------
    PHI2

  NIVEAU  1
 -----------
    CF

  NIVEAU  0
 -----------
    "FIN"
************************************************************
*******   F I N    D E  S I M U L A T I O N  **********
************************************************************
```

XIII. CONCLUSION

Nous avons montré dans ce qui précède qu'il était possible d'interpréter en LREMUN, les mécanismes de contrôle de LIDO, à savoir :

- le mécanisme d'horloge,
- le mécanisme de conditions,
- et le mécanisme séquenceur.

Cette interprétation a été facile à réaliser, du fait que le langage LIDO permet une décomposition fonctionnelle en plusieurs niveaux systémiques ou mécanismes, d'un système logique.

Une description du MC 6800 a été effectuée ; elle a amené à des extensions des objets, manipulés par le langage, du type :

- latchs (en distinction avec les registres),
- terminaux (pour la définition des bus, broches,...).

Cette description, nous a permis aussi de confirmer :

- la facilité de l'utilisation de LIDO,
- la clarté de description de systèmes complexes,
- la facilité de correction des erreurs de description.

En ce qui concerne le simulateur, il a été remarqué l'utilité de comprimer ses résultats et d'offrir à l'utilisateur un moyen intéractif lui permettant la sélection des résultats qui l'intéresse.

CONCLUSION GENERALE

Le résultat principal du travail présenté dans cette thèse est d'avoir montré l'intérêt du développement d'un outil de modélisation permettant de représenter d'une manière abstraite un système logique à différents niveaux de description.

La définition d'un tel outil vise essentiellement la facilité de passage entre les niveaux de description. Les REseaux MUlti-Niveaux (REMUN) est l'outil proposé pour ce but ; nous avons montré leurs applications à différents niveaux et détaillé celle du niveau transfert de registre, pour lequel le langage de description LIDO a été proposé.

La réalisation de l'interpréteur, de LIDO en REMUN, a été confrontée à l'interprétation des niveaux systémiques opératoires, "OPERATION", "MEMOIRE", "REGISTRE" et "DECODEUR" qui sont restés décrits d'une manière comportementale.

Les études effectuées dans cette thèse permettent de définir quelques directions de travail pour le développement d'un système intégré multi-niveaux d'aide à la conception :

- développer les REMUN afin de permettre la modélisation du contrôle positif-négatif d'une part et les transitions temporisées d'autre part,
- définir un langage de description multi-niveaux se basant sur une base syntaxique adaptée à la réalisation matérielle et facile à utiliser,
- réaliser une interprétation de cette base syntaxique en REMUN développé,
- enfin, réaliser des outils logiciels automatisant la génération de masques.

La finalité de ce type d'outils étant leur utilisation effective par les concepteurs de systèmes logiques, il est important d'offrir un produit facile à utiliser et possédant le minimum de symbolismes et de notations.

ANNEXE 1

Cette première annexe regroupe les principaux modules utilisés pour la simulation des REseaux MUlti-Niveaux (REMUN) :

Module 1 : Spécification des états structurels actifs.

Module 2 : Transition de contenu.

Spécification des états structurels actifs ;
co ce module consiste à déterminer les états
structurels qui sont actifs en testant les
états de contenu des places qui les activent co

pour tout $(p_i, p_j) \in TC$ faire

$$\text{Actif } (p_i, p_j) \leftarrow \bigcap_{(p_k, (p_i, p_j)) \in TCS} ec(p_k)$$

fait

Transition de contenu ;
co ce module effectue les transitions de contenu
des états structurels actifs co

pour tout (p_i, p_j) actif faire

$ec(p_j) \leftarrow ec(p_i)$;
$ec(p_i) \leftarrow$ faux

fait

Simulateur de REMUN ;

Création de la structure de données du REMUN.

tt que non (fin de simulation) faire
    spécification des états structurels actifs ;
    transition de contenu
fait

****
* ANNEXE 2 *
****

LA SYNTAXE DE LIDO

LA SYNTAXE DU LANGAGE LIDO

Pour la description syntaxique de LIDO, on a utilisé la forme de Backus-Naur (BNF).

En effet, les constructions syntaxiques seront notées entre ⟨ ⟩.
::= signifiera "peut etre réécrit en".
! apparaissant à droite d'une production, signifiera "ou".
[ ] délimitront les constructions répétées.
On définira alors la syntaxe de LIDO ainsi:

1-LANGAGE ALPHABET:

```
⟨ALPHA⟩        ::= A ! B ! C ! D ! E ! F ! G ! H ! I ! J ! K ! L ! M ! N !
                   O ! P ! Q ! R ! S ! T ! U ! V ! W ! X ! Y ! Z ! 0 ! 1 !
                   2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9 ! + ! * ! - ! , ! ; ! . !
                   = ! ' ! " ! : ! ( ! ) ! / ! @
```

2-NOMS RESERVES:

```
⟨NOMR⟩         ::= "REGISTRE" ! "SOUSREGISTRE" ! "MEMOIRE" ! "OPERATIONS"
                   ! "DECODEUR" ! "SEQUENCEUR" ! "HORLOGE" ! "CONDITIONS"
                   ! "FIN"
```

3-ELEMENTS MEMOIRES:

```
⟨REGISTRE⟩     ::= "REGISTRE" [ ⟨NOMREG.PRIM⟩ ⟨NUM.DES.BITS⟩ ; ]

⟨NOMREG.PRIM⟩  ::= ⟨NOM⟩

⟨NUM.DES.BITS⟩::= (⟨B1⟩:⟨B2⟩)

⟨NOM⟩          ::= ⟨LETTRE⟩ [ ⟨LETTRE⟩ ! ⟨CHIFFRE⟩ ]

⟨B1⟩ ::= ⟨B2⟩  ::= ⟨NB.DECIMAL⟩

⟨LETTRE⟩       ::= A ! B ! C ! D ! E ! F ! G ! H ! I ! J ! K ! L ! M ! N !
                   O !P ! Q ! R ! S ! T ! U ! V ! W ! X ! Y ! Z

⟨CHIFFRE⟩      ::= 0 ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9

⟨NB.DECIMAL⟩   ::= ⟨CHIFFRE⟩ [ ⟨CHIFFRE⟩ ]

⟨SOUREGISTRE⟩  ::= "SOUSREGISTRE" [ ⟨NOM.DU.SS.REG⟩ = ⟨NOMREG.PRIM⟩
                                                      ⟨NUM.DES.BITS⟩ ; ]

⟨NOM.DU.SS.REG⟩::= ⟨NOM⟩
```

```
⟨MEMOIRES⟩      ::= "MEMOIRE" [ ⟨NMEM⟩ ( ⟨NBREMOTS⟩ , ⟨LONGMOT⟩ ) ; ]

⟨NMEM⟩          ::= ⟨NOM⟩

⟨NBREMOTS⟩      ::= ⟨K1⟩ : ⟨K2⟩

⟨K1⟩  ::= ⟨K2⟩  ::= ⟨NB.DECIMAL⟩

⟨LONGMOT⟩       ::= ⟨NB.DECIMAL⟩
```

APPELS D'ELEMENTS MEMOIRES

```
⟨APPEL.REG⟩     ::= ⟨NOMREG.PRIM⟩    ⟨NUM.DES.BITS⟩ !
                    ⟨NOM.DU.SS.REG⟩  ⟨NUM.DES.BITS⟩

⟨APPEL.MEM⟩     ::= ⟨NMEM⟩ ( ⟨ADR⟩ ) ! ⟨NMEM⟩ ( ⟨REG.ADR⟩ )

⟨ADR⟩           ::= ⟨NB.DECIMAL⟩

⟨REG.ADR⟩       ::= ⟨NOMREG.PRIM⟩ ! ⟨NOM.DU.SS.REG⟩
```

4-HORLOGE

```
⟨HORLOGE⟩       ::= "HORLOGE" ⟨NB.DE.PHASES⟩ ;

⟨NB.DE.PHASES⟩  ::= ⟨NB.DECIMAL⟩
```

5-CONDITIONS

```
⟨CONDITIONS⟩    ::= "CONDITIONS" [ ⟨NOM.COND⟩ : ⟨EXP.BOOL⟩ ; ]

⟨NOM.COND⟩      ::= ⟨NOM⟩

⟨EXP.BOOL⟩      ::= ⟨COMP.BOOL⟩ ! ⟨COMP.BOOL⟩ + ⟨COMP.BOOL⟩

⟨COMP.BOOL⟩     ::= ⟨FACT.BOOL⟩ ! ⟨FACT.BOOL⟩ * ⟨FACT.BOOL⟩

⟨FACT.BOOL⟩     ::= ⟨RELATION⟩ ! ⟨RELATION⟩ ⟨OP.NEG⟩

⟨RELATION⟩      ::= ⟨ARG⟩ ⟨OP.REL.COMP⟩ ⟨ARG⟩

⟨OP.REL.COMP⟩   ::= ⟨REL.SIMP⟩ ! ⟨REL.SIMP⟩ ⟨OP.NEG⟩

⟨ARG⟩           ::= ⟨ARG.SIMP⟩ ! ⟨ARG.SIMP⟩ ⟨OP.NEG⟩ ! ⟨NB.BINAIRE⟩

⟨ARG.SIMP⟩      ::= ⟨APPEL.REG⟩ ! ⟨APPEL.MEM⟩

⟨OP.NEG⟩        ::= '

⟨REL.SIMP⟩      ::= ( ! ) ! =

⟨NB.BINAIRE⟩    ::= ⟨BIT⟩ [ ⟨BIT⟩ ]
```

6-SEQUENCEUR

```
⟨SEQUENCEUR⟩    ::= "SEQUENCEUR" [ ⟨DESC.TRANS⟩ ]

⟨DESC.TRANS⟩    ::= ⟨N.DEP⟩ : [ ⟨DESC.ARC⟩ ; ] ⟨DESC.ARC⟩ .

⟨DESC.ARC⟩      ::= ⟨ARC.COND⟩ ! ⟨ARC.NON.COND⟩ ! ⟨ARC.VIDE⟩

⟨ARC.COND⟩      ::= / ⟨NOM.COND⟩ / ⟨SIGN.SEQ⟩ - ⟨N.AR⟩

⟨ARC.NON.COND⟩  ::= ⟨SIGN.SEQ⟩ - ⟨N.AR⟩

⟨ARC.VIDE⟩      ::= - ⟨N.AR⟩

⟨SIGN.SEQ⟩      ::= [ ⟨N.SIGNAL⟩ , ] ⟨N.SIGNAL⟩

⟨N.SIGNAL⟩      ::= ⟨NOM⟩

⟨N.AR⟩          ::= ⟨NOM.NOEUD⟩ ! "FIN"

⟨N.DEP⟩         ::= ⟨NOM.NOEUD⟩

⟨NOM.NOEUD⟩     ::= ⟨NOM⟩

⟨NOM.COND⟩      ::= ⟨NOM⟩
```

7-DECODEUR

```
⟨DECODEUR⟩      ::= "DECODEUR" [ ⟨REG.DECOD⟩ , ] ⟨REG.DECOD⟩ ;
                        [ ⟨VALIDATION⟩ : [ ( ⟨CODE⟩ ) ]
                                         [ , ⟨SIGN.CONT⟩ ] ; ]

⟨REG.DECOD⟩     ::= ⟨APPEL.REG⟩

⟨VALIDATION⟩    ::= ⟨SIGN.VAL⟩ ! ⟨SIGN.VAL⟩ * ⟨VALIDATION⟩

⟨SIGN.VAL⟩      ::= ⟨PHASE.H⟩ ! ⟨N.SIGN⟩

⟨PHASE.H⟩       ::= PHI ⟨NB.DECIMAL⟩

⟨CODE⟩          ::= ⟨NB.BIN.VAR⟩ !
                    ( ⟨NB.BIN.VAR⟩ ) ⟨SIGN.COND⟩ ( [ ⟨SIGN.CONT⟩ , ]
                      ⟨SIGN.CONT⟩ )
                    ! ⟨SIGN.COND⟩ ( [ ⟨SIGN.CONT⟩ , ] ⟨SIGN.CONT⟩ )

⟨NB.BIN.VAR⟩    ::= ⟨BIT⟩ ! ⟨BIT⟩ ⟨NB.BIN.VAR⟩ ! ⟨VAR⟩ ⟨NB.BIN.VAR⟩
                    ! ⟨VAR⟩

⟨VAR⟩           ::= X

⟨SIGN.COND⟩     ::= /⟨NOM.COND⟩/

⟨SIGN.CONT⟩     ::= ⟨NOM⟩
```

8-OPERATIONS

⟨OPERATIONS⟩ ::= "OPERATIONS" ⟨CORPS.OP⟩

⟨CORPS.OP⟩ ::= [⟨SIGN.ACT⟩ : ⟨OP⟩;]

⟨SIGN.ACT⟩ ::= ⟨SIGNAL⟩ * ⟨SIGNAL.ACT⟩ ! ⟨SIGNAL⟩

⟨SIGNAL⟩ ::= ⟨SIGN.COND⟩ ! ⟨SIGN.CONT⟩ ! ⟨PHASE.H⟩

⟨OP⟩ ::= ⟨NOM.OP⟩ : ⟨EXP.OP⟩

⟨NOM.OP⟩ ::= ⟨NOM⟩

⟨EXP.OP⟩ ::= ⟨MBRE.G⟩ ⟨- ⟨MBRE.D⟩

⟨MBRE.G⟩ ::= ⟨APPEL.REG⟩ ! ⟨APPEL.MEM⟩

⟨MBRE.D⟩ ::= ⟨MBRE.D.DIAD⟩ ! ⟨MBRE.D.MON⟩

⟨MBRE.D.DIAD⟩ ::= ⟨APPEL⟩ ⟨OPR⟩ ⟨APPEL⟩

⟨MBRE.D.MON⟩ ::= ⟨APPEL⟩

⟨APPEL⟩ ::= ⟨APP.R.M⟩ ! ⟨APP.R.M⟩ '

⟨APP.R.M⟩ ::= ⟨APPEL.REG⟩ ! ⟨APPEL.MEM⟩

⟨OPR⟩ ::= + ! * ! - ! @+ ! @- ! @*

BIBLIOGRAPHIE

[ALA] R. ALALI, C. DURANTE, J.J. MERCIER.
"MODFLO : Un Modèle Fonctionnel de Description de Systèmes Logiques et de Flots d'Informations".

[AMB] P. AMBLARD.
"Conception temporelle sûre de circuits intégrés complexes".
Thèse de 3ème cycle, INPG, Juin 1983.

[ANC1] F. ANCEAU.
"Real Assumption Concerning IC Design".
Advanced Course on VLSI Architecture. Univ. of Bristol (OK), 19-30 July 1982.

[ANC2] F. ANCEAU.
"LSI-Processor Architecture and Design".
Advanced Course on VLSI Architecture. Univ. of Bristol (OK), 19-30 July 1982.

[ASH] E.A. ASHCROFT.
"Lucid, a Nonprocedural Language with Iteration".
CACM, July 1977, Vol. 20, Number 7.

[ASH] H. ATLAN.
"Organisation en Niveau Hiérarchique dans les systèmes vivants".
Université Paris VI.

[ATL] J.L. BAER.
"Models for the Design, Simulation, and Performance of Distributed-Function Architecture".
Lake Arrowhead Workshop 1973.

[BAE] J.L. BAER.
"Computer Architecture".
Computer, October 1984.

[BAK] P. BAKOWSKI.
"An Essay to introduce a unified theory of computative forms".
Conference on Methodologie for Computer System Design, I.F.I.P. WG 10.1, 15-17 September 1983.

[BAR] M.R. BARBACCI.
"A Comparison of Register Transfer Languages for Describing Computers and Digital Systems".
IEEE Trans. on Computers, Vol. C-24, n° 2, Feb. 1975.

[BOR] D. BORRIONE.
"Langages de description de systèmes logiques. Propositions pour une méthode formelle de définition".
Thèse d'Etat, USMG, France, Juillet 1981.

[BUR] R.M. BURGER and All.
"The Impact of ICs on Computer technology".
Computer, October 1984.

[CLE] W.M. VAN CLEEMPUT and H. OFEK.
"Design Automation for Digital Systems".
Computer, October 1984.

[CHI] C. CHICOIX and All.
"EPISODE : A set of tools oriented at logic integreted circuit design, verification and testing".
PROC. of the ESSCIRC 76, Toulouse, Sept. 1976.

[CHS] S. CHUQUILLANQUI - T. PERES SEGOVIA.
"POALA, un système CAO pour l'optimisation topologique et le dessin automatique des masques de PLA complexes".
Acte du Congrès de l'AFCET, 17-19 Novembre 1982.

[CHU1] Y. CHU.
"Introduction CDL".
Computer, December 1974.

[CHU2] Y. CHU.
"Why do we need Computer Hardware Description Languages ?".
Computer, Vol. 7, n° 2, Decembre 1974.

[CLE] W.M. VAN CLEEMPUT and All.
"Design Automation for Digital Systems".
Computer, October 1984.

[COR] D. CORBEL, J.C. GENTINA, C. VERCAUTER.
"Généralisation des Réseaux de Pétri".
A.I. 83 lasted Symposium, Lille, March 1983.

[DAS] S. DASGUPTA and M. OLAFSSON.
"Towards a Family of Languages for the Design and Implementation of Machine Architectures".
IEEE, 1982.

[DAV] R. DAVIS and H. SHROBE.
"Representing Structure and Behavior of Digital Hardware".
Computer, October 1983.

[DUR] A. DURET.
"Participation à la conception et à la réalisation en LSI, de la partie opérative d'une machine intégrée".
INPG, Grenoble, Thèse de 3ème cycle, Décembre 1979.

[ETI] R. ETIENNE.
"Etude des Méthodologies de conception, outils de synthèse et de génération automatiques de parties contrôle de microprocesseurs".
Thèse de 3ème cycle, INPG, Juin 1983.

[FAR] M.C. FARLAND.
"An Abstract Model of Behavior for Hardware Descriptions".
IEEE Trans. on Computers, Vol. C-32, n° 7, July 1983.

[FUS] A. FUSAOKA and M. HIRAYAMA.
"Compiler chip : A Hardware Implementation of Compiler".
ACM, March 1982.

[FLY] M. FLYNN.
"Some Computer Organization and their effectiveness".
IEEE Trans. on Computers, Vol. C-21, n° 9, September 1972.

[GIR] C. GIRAULT.
"Réseaux de Pétri et Synchronisation de Processus".
n° I.P. 78-02. Institut de Programmation CNRS - ERA 592.

[GLU] V. GLUSHKOV, A. LETICHEVSKU.
"Theory of Algorithms and Discrete Processors".
In : Advances in Information Sciences (Ed. Julius Tou), 1964.

[HIL] F.J. HILL and All.
"Structure Specification with a Procedural Hardware Description Language".
IEEE Trans. on Computers, Vol. C-30, n° 2, Feb. 1981.

[JAH] K. JAHIDI.
"Langage IRENE, Definition et exemple de description".
Mémoire de DEA, INSIMAG, Juin 1982.

[KRY] A.J. KRYGIEL.
"Synchronous Nets for Single Instruction Stream-Multiple Data Stream Computers".
IEEE, 1981.

[KLU] W.E. KLUG.
"An Unifying Approach to Computer System Modeling Based on Petri Nets".
IFIP W.G. 10.1. Conference on Methodology for Computer System Design, Lille, Sept. 1983.

[KUN] J. KUNTZMANN.
"Théorie des Réseaux".
Dunod, 1972.

[LEC] J. LECOURVOISIER, A. PUISSOCHET.
"CASSIOPE : Un système intégré pour la CAO de VLSI".
Proceedings of MICAD'84.

[LIM] W. LIM.
"HISD. A Structure Description Language".
CACM, Nov. 1982, Vol. 25, n° 11.

[LIP1] G.P. LIPOVSKI.
"Hardware Description Languages : Voices from the Tour of Basel".
Computer, June 1977.

[LIP2] G.J. LIPOVSKI.
"Developments and Directions in Computer Architecture".
Computer, August 1978.

[MAR] E.D. MARVIN and W. GWIN.
"CAD Systems for IC Design".
IEEE. CAD, Vol. 1, January 1982.

[MAI] F. MAISON.
"Une vue globale des problèmes liés à la conception des VLSI".
TSI, 1983.

[MEA] C. MEAD and L. CONWAY.
"Introduction to VLSI systems".
Addison-Wesley, Publishing Compagny. 1980.

[MEN] J.R. MENAND and M. BECKER.
"Modeling a Multiprocessor Architecture".
IEEE Trans. on Software Engineering, Vol. SE-9, n° 2, March 1983.

[MER1] J. MERMET.
"An Integreted CAD System for logical and electronic circuits".
CHDL and their applications. North-Holland publishing Compagny.
IFIP, 1981.

[MER] J.J. MERCIER.
"Contribution à la modélisation fonctionnelle des structures logiques".
Thèse d'Etat, 19 Décembre 1980, Univ. des Sciences et Techniques du Languedoc.

[MER2] J. MERMET.
"Etude méthodologique de la conception assistée par ordinateur des systèmes logiques : CASSANDRE".
Thèse d'Etat, 10 Avril 1973, Univ. Scirntifique et Médicale de Grenoble.

[MIL] R. MILLADI, G. SERRERO.
"OASIS. An Automatic design tool for symbolic layout of Circuits".
Proc. of the ESSCIRC, 1981, Frībourg, Oct. 1981.

[MIN] IMS T4 24 Transputer.
INMOS Reference Manual.

[MOA] M. MOALLA,J. DULOU, J. SIFAKIS.
"Réseaux de Pétri Synchronisés".
RAIRO Automatique, Vol. 12, n° 2, 1978.

[NEM] M. NEMMOUR.
"Formalisme DELTA : Un outil de description logique pour la synthèse automatique dans la conception des machines séquentielles synchrones".
Thèse de 3ème cycle, INPG, Décembre 1981.

[PAW1] A. PAWLAK.
"Microprocessor systems modeling with MODLAN".
20eme Design Automation Conference, pp. 804-811, Miami, 27-29 Juin 1983.

[PAW2] A. PAWLAK.
"MODLAN. A language for multilevel description and modeling of digital systems".
IFIP 81, Computer Hardware Description Languages and their Applications.

[PAT1] D.A. PATTERSON.
"Reduced Instruction Set Computer".
C-ACM, Vol. 28, n° 1, January 1985.

[PAT2] J.L. PATERSON and Th. H. BREDT.
"A comprison of models of parallel computation".
Information Processing 74. North-Holland, Publishing Compagny, 1974, pp. 466-470.

[PET] C.A. PETRI.
"Introduction to general Net Theory".
G.M.D. Bonn.

[POP] J. POPIEL and A.T. ROSINSKI.
"Multilevel Simulation of LSI Digital Circuits.
Polish Academy of Sciences. Warsan.

[RAY] T.C. RAYMOND.
LSI/VLSI Design Automation".
Computer, July 1981.

[RIG] J.P. RIGANATI and P.B. SCHNECK.
"Supercomputing".
Computer, October 1984.

[SHI] S.G. SHIVA.
"Computer Hardware Description Language - A tutorial".
Proceeding of IEEE, Vol. 67, n° 12, December 1979.

[SU] S. SU.
"Hardware Description Language Applications : An Introduction and Prognosis".
Computer, June 1977.

[TOU] B. TOURSEL.
"Contribution à l'étude de l'architecture des ordinateurs : proposition pour un nouveau mode de fonctionnement".
Thèse d'Etat, Unviersité de Lille 1, Septembre 1979.

[WIL] M.Y. WILKES.
"Keynote Adress : Size, Power and Speed".
C-ACM, 1983.

[ZAC] M. ZACHARIDES.
"MAS : Realisation d'un langage d'aide à la description et à la conception des systèmes logiques".
Thèse de 3ème cycle, INPG, Septembre 1977.

RESUME

La complexité des systèmes logiques à très grande échelle d'intégration (VLSI) a rendu coûteuses leur simulation ainsi que leur conception, d'où la nécessité de disposer de langages de description et d'outils de modélisation, associés aux niveaux de description, allant du niveau architecture système jusqu'au niveau d'implémentation. Or le passage d'un niveau à un autre est d'autant plus difficile que les outils de modélisation associés sont incompatibles.

Ce travail présente à la fois un outil de modélisation (REseaux MUlti-Niveaux : REMUN) et un langage de description (LIDO : Langage Interprété de Description des Ordinateurs).

Une première partie concerne la définition de REMUN dont le but est de permettre la modélisation d'un système logique à tous les niveaux de description. Un simulateur de cet outil a été réalisé afin d'évaluer le comportement du système modélisé.

Une seconde partie présente l'application de cet outil au niveau transfert de registre, un langage de description de systèmes logiques, LIDO, a été défini. Un interpréteur, de LIDO en REMUN a été réalisé afin de simuler le comportement des systèmes logiques au cours du temps.

MOTS CLES : Systèmes VLSI, Outils de modélisation, Niveaux de description, Langage de description de systèmes logiques, Mode de description, Décomposition fonctionnelle, Hiérarchie structurelle, Système d'aide à la conception.