



HAL
open science

Conception et réalisation d'un [sic] architecture multi-microprocesseur flexible : application au contrôle de processus industriel

Hussein Habannakeh-Midani

► **To cite this version:**

Hussein Habannakeh-Midani. Conception et réalisation d'un [sic] architecture multi-microprocesseur flexible : application au contrôle de processus industriel. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1979. Français. NNT : . tel-00289017

HAL Id: tel-00289017

<https://theses.hal.science/tel-00289017>

Submitted on 19 Jun 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
Génie Informatique

par

HABANNAKEH—MIDANI Hussein



CONCEPTION ET REALISATION D'UN ARCHITECTURE
MULTI-MICROPROCESSEUR FLEXIBLE;
APPLICATION AU CONTROLE DE PROCESSUS INDUSTRIEL



THESE SOUTENUE LE 28 MAI 1979 DEVANT LA COMMISSION D'EXAMEN

Monsieur	BOLLIET	Président
Madame	SAUCIER	Examineurs
Messieurs	PRUNET	
	ROMAND	
	BOUDILLON	
	BOUTTAZ	
	CASPI	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET

M. Georges LESPINARD

PROFESSEURS TITULAIRES

MM. BENOIT Jean	Electronique - automatique
BESSON Jean	Chimie minérale
BLOCH Daniel	Physique du solide - cristallographie
BONNETAIN Lucien	Génie chimique
BONNIER Etienne	Métallurgie
* BOUDOURIS Georges	Electronique - automatique
BRISSONNEAU Pierre	Physique du solide - cristallographie
BUYLE-BODIN Maurice	Electronique - automatique
COUMES André	Electronique - automatique
DURAND Francis	Métallurgie
FELICI Noël	Electronique - automatique
FOULARD Claude	Electronique - automatique
LANCIA Roland	Electronique - automatique
LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
LESPINARD Georges	Mécanique
MOREAU René	Mécanique
PARIAUD Jean-Charles	Chimie - physique
PAUTHENET René	Electronique - automatique
PERRET René	Electronique - automatique
POLOUJADOFF Michel	Electronique - automatique
TRAYNARD Philippe	Chimie - physique
VEILLON Gérard	Informatique fondamentale et appliquée
* en congé pour études	

PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuël	Electronique - automatique
BOUVARD Maurice	Génie mécanique
COHEN Joseph	Electronique - automatique
GUYOT Pierre	Métallurgie physique
LACOUME Jean-Louis	Electronique - automatique
JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM.	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	ZADWORNY François	Electronique - automatique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	CHARTIER Germain	Electronique - automatique
	CHIAVERINA Jean	Biologie, biochimie, agronomie
	IVANES Marcel	Electronique - automatique
	LESIEUR Marcel	Mécanique
	MORET Roger	Physique nucléaire - corpusculaire
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	SABONNADIÈRE Jean-Claude	Informatique fondamentale et appliquée
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie Physique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahim	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)
E.N.S.E.E.G.

MM.	BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

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

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

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

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	Centre national recherche scientifique

A mes parents

Avec ma profonde affection

Ce travail a été effectué dans le cadre de l'Equipe "Conception et Sécurité des Systèmes" du Laboratoire Associé au C.N.R.S. (LA7), ENSIMAG, Grenoble. La réalisation pratique s'est faite dans le cadre de l'atelier de microinformatique de Grenoble.

Je remercie vivement Monsieur L. BOLLINET, Professeur à l'Université de Grenoble, pour l'honneur qu'il me fait en acceptant de présider le Jury de cette thèse.

Que Madame G. SAUCIER, Maître de Conférence à l'Institut National Polytechnique de Grenoble, me permette de lui exprimer ma sincère reconnaissance pour les conseils judicieux et éclairés ainsi que les encouragements qu'elle n'a cessé de me prodiguer afin de mener à bien les recherches qu'elle m'avait confiées.

Mes remerciements vont aussi à :

Monsieur PRUNET, Maître de Conférence à l'Université des Sciences et Techniques de Montpellier,

Monsieur ROMAND, Ingénieur en Chef de la Sté CROUZET,

Monsieur BODILLON, de la Sté OPTION,

de bien vouloir participer au Jury de cette thèse.

Je remercie spécialement Monsieur R. BOUTTAZ pour l'aide technique et l'environnement amical que j'ai trouvés dans l'atelier de micro-informatique.

Je tiens à remercier chaleureusement les membres de l'équipe "Conception et Sécurité des Systèmes" pour leur collaboration, en particulier Monsieur P. CASPI et Monsieur J. PULOU, chercheurs au CNRS, de leur aide précieuse et les conseils constants et efficaces qu'ils m'ont apportés au cours de mes travaux.

Enfin, je remercie celles et ceux qui ont travaillé à la dactylographie et au tirage de cette thèse, en particulier Madame G. DUFFOURD pour son chaleureux dévouement.

TABLE DES MATIERES

INTRODUCTION	
I - OBJECTIF DE L'ETUDE	1
II - PRESENTATION DU DOCUMENT	2
<u>CHAPITRE I</u> : GENERALITES SUR LES ARCHITECTURES MULTIMICROPROCESSEUR ET LA TOLERANCE AUX PANNES	3
I - 1. Architectures multimicroprocesseur	4
I - 1.1. Méthodes de communication entre processeurs	4
I - 1.2. Types d'architecture	7
I - 2. Tolérance aux pannes	8
I - 2.1. Définition de la tolérance aux pannes	8
I - 2.2. Système à haute sécurité - système à haute disponibilité	8
I - 2.3. Intérêt des structures modulaires	9
<u>CHAPITRE II</u> : DEFINITION DU SYSTEME	14
II - 1. Présentation	15
II - 2. Eléments du choix d'une structure	15
II - 3. Présentation et choix de l'architecture retenue	19
II - 3.1. Adressage et mémoire logique	19
II - 3.2. Mémoire physique et protection	19
II - 3.3. Choix de cette architecture	23
<u>CHAPITRE III</u> : DESCRIPTION DETAILLEE DU SYSTEME	24
III - 1. Résolution des conflits d'accès mémoire par multiplexage temporel	25
III - 1.1. Accès mémoire du M6800	25
III - 1.2. Multiplexage temporel	27
III.1.2.1. Système biprocesseur	27
III.1.2.2. Système quadriprocesseur	28
III.1.2.3. Généralisation	28
III.1.2.4. Solution mixte	31
III.1.2.5. Réalisation du multiplexage temporel	34

III - 2. Protection mémoire et allocation	38
III - 2.1. Protection de l'écriture en ROM et en RAM	38
III - 2.2. Protection fine et allocation des pages	41
III - 3. Architecture réalisée	44
III - 3.1. Liaisons processeurs - mémoire	44
III - 3.2. Implantation et découpage	44
III - 3.3. Sélection et organisation du banc	45
III - 3.4. Système d'interruption	45
<u>CHAPITRE IV : MISE EN OEUVRE DE LOGICIEL SUR LE SYSTEME "4M"</u>	49
IV - 1. Interpréteur de Réseau de Petri	50
IV - 1.1. Généralité	50
IV - 1.2. Implémentation d'un interpréteur de Réseau de Petri Synchronisé sur "4M"	51
IV - 1.3. Description du logiciel	53
IV - 2. Exemple de l'automatisation d'une centrale à béton	55
IV - 2.1. Description d'une centrale à béton	55
IV - 2.2. Modélisation sous forme de Réseau de Petri	57
IV.2.2.1. Découpage en chaîne d'asservissement	57
IV.2.2.2. Définition des différentes tâches	57
IV.2.2.3. Modélisation des tâches	59
IV - 2.3. Asservissement d'une chaîne balance	61
IV.2.3.1. Principe d'une pesée	61
IV.2.3.2. Principe du séquenceur balance	61
IV - 2.4. Principe du séquenceur central	64
IV - 2.5. Entrée/sortie	64
<u>CHAPITRE V : ETUDE DE LA TOLERANCE AUX PANNES</u>	67
V - 1. Introduction	68
V - 2. Examen de l'architecture	68
V - 3. Horloge et organes de multiplexage temporel	69
V - 3.1. Détection - localisation - remplacement	69
V - 3.2. Masquage	72

<u>CHAPITRE VI : EXTENSIBILITE</u>	75
VI - 1. Extensibilité intrinsèque : système "8M"	76
VI - 2. Architectures moléculaires fondées sur le système "4M"	80
VI - 2.1. Proposition (1)	80
VI - 2.2. Proposition (2)	80
VI - 2.3. Proposition (3) : DMA	80
VI - 3. Généralisation	85
VI - 3.1. Définition	85
VI - 3.2. Exemple	86
VI - 3.3. Condition nécessaire et suffisante d'existence	88
CONCLUSION	89
<u>ANNEXE A : PROGRAMMATION DIRECTE DE L'EXEMPLE DE LA CENTRALE A BETON</u>	91
A - 1. Introduction	92
A - 2. Algorithmes	94
A - 3. Listings	96
<u>ANNEXE B : Nomenclature des principaux circuits intégrés utilisés</u>	106
BIBLIOGRAPHIE	107

INTRODUCTION

I - OBJECTIFS DE L'ETUDE

Ce travail traite de l'étude et de la réalisation d'un système multimicroprocesseur. Il importe tout d'abord de préciser la gamme et le domaine d'application de la machine que nous nous sommes proposés de réaliser. En effet, si à leur origine les microprocesseurs étaient destinés à des applications de taille limitée (automatismes, calcul de bureau, entrées-sorties de systèmes plus puissants), très vite des structures multimicroprocesseurs ont été conçues de façon à concurrencer des calculateurs d'usage général "minis" [MICRAL] ou "moyens" [MAZARE].

Notre projet est différent et vise plutôt la gamme des ordinateurs de contrôle de processus (automatismes industriels, avionique), sans dispositifs matériels de traduction externe, d'adressage, de mémoire, d'interruptions autres que ceux des microprocesseurs couramment disponibles sur le marché. Notre projet peut donc se résumer de la façon suivante :

Etudier une machine :

- qui réalise une extension simple et peu coûteuse des microprocesseurs d'usage courant,
- qui soit performante , en ce sens qu'elle utilise au mieux les performances des composants précités,
- qui jouisse de propriétés de tolérance aux pannes telles que cette machine puisse être facilement adaptée à divers objectifs de sûreté de fonctionnement (disponibilité, sécurité), objectifs dont l'importance en contrôle de processus n'est plus à souligner,
- enfin, il nous a semblé important, afin d'étendre éventuellement sa gamme d'applications, d'étudier une machine qui soit extensible.

II - PRESENTATION DU DOCUMENT

Après avoir, dans un premier chapitre, présenté des généralités sur les architectures multiprocesseurs et sur la tolérance aux pannes, nous décrivons les choix successifs que nous avons effectués pour parvenir à la définition du système que nous proposons. Le choix fondamental qui a été effectué à ce stade porte sur le mécanisme de partage de mémoire. Nous avons, en effet, choisi une méthode de multiplexage temporel. Si cette méthode présente certains avantages, rapidité, coût peu élevé, bonne adaptation au microprocesseur choisi, ses inconvénients sont a priori importants : elle semble imposer l'existence d'un point "dur" du point de vue de la tolérance aux pannes, et son extensibilité paraît très limitée. En effectuant ce choix nous avons cherché à montrer comment on peut tirer profit des avantages de cette méthode tout en remédiant à ses inconvénients.

Ce choix fondamental étant fait, le chapitre III est consacré à la description détaillée de la machine réalisée dite "4M". Cette machine est un quadri processeur qui utilise au mieux les performances du microprocesseur de base choisi : le Motorola M 6800.

Au chapitre IV, un logiciel spécialisé pour la commande de processus industriels, en cours d'implémentation sur la machine "4 M", est présenté. Ce logiciel consiste en un interpréteur de systèmes décrits directement en réseaux de Petri interprétés suivant l'approche MAS.

Le cas particulier de la commande d'une centrale à béton étudiée dans cette approche est présentée.

Au chapitre V, les propriétés de tolérance aux pannes de la machine réalisée sont examinées. On montre que, mis à part les circuits réalisant le multiplexage temporel, (horloge, génération de phase) qui constituent un point dur, dans le pire des cas une panne rend indisponible la moitié de la machine, ce qui apparaît comme satisfaisant. En ce qui concerne le point dur, on justifie partiellement le choix fait en II en montrant que ce point dur est petit, donc sa fiabilité est bonne, et qu'elle peut être encore améliorée, en augmentant la qualité des composants employés ou en utilisant des techniques de redondance.

Enfin, au chapitre VI, la question de l'extensibilité est étudiée. On montre que si l'extensibilité intrinsèque est limitée, on peut cependant construire facilement des réseaux de machine 4M communicant par mémoires communes. Ces réseaux peuvent présenter une grande diversité puisqu'ils peuvent être caractérisés par des graphes n -coloriables ($n = 4$ pour 4M).

CHAPITRE I

GENERALITES SUR LES ARCHITECTURES MULTIMICROPROCESSEUR

ET LA TOLERANCE AUX PANNES

-0-0-0-

I - 1. ARCHITECTURES MULTIMICROPROCESSEUR

Il nous semble que les architectures multimicroprocesseur peuvent se caractériser de deux points de vue : le type de communications entre processeurs et le type de spécialisation des processeurs, ces deux aspects étant naturellement liés.

I - 1.1. Méthodes de communication entre processeurs

Deux grands principes de communication semblent se dégager :

a) Communications par interface [2]

Chaque processeur a sa mémoire et ses unités d'entrées/sorties propres : l'ensemble forme un microcalculateur. Ces microcalculateurs sont reliés entre eux par connexion directe de leurs interfaces d'entrée/sortie. On peut ainsi construire des réseaux réguliers ou irréguliers. Les caractéristiques de cette méthode sont :

- Elle est facilement extensible et peu coûteuse pour des structures de communication simples. Cependant, son prix croît vite lorsque l'on enrichit la structure de communication. Ainsi un réseau complet n nécessite $n(n-1)$ ports.
- D'autre part, cette méthode est lente (débit d'information faible) car elle fait appel au déroulement d'un algorithme de communication réparti sur les deux calculateurs communicants. L'échange d'un mot entre deux calculateurs nécessite, en général, l'exécution de plusieurs instructions.

b) Communications par mémoire commune et partage de bus

Cette méthode consiste à connecter plusieurs microprocesseurs sur un même bus supportant des mémoires et des interfaces d'entrée/sortie. Le conflit d'accès au bus des processeurs peut être résolu de diverses façons. On ne s'intéresse ici qu'aux dispositifs matériels simples d'allocation du bus. Ces dispositifs sont encore appelés "arbitres de bus". Leurs caractéristiques communes sont :

- La transparence au niveau logiciel : chaque processeur ignore les autres processeurs; il affiche des adresses et si le bus n'est pas libre, le déroulement de l'instruction en cours est bloqué (de diverses façons, selon le type de processeur) jusqu'à ce que le bus lui soit alloué.

- Le débit de communication obtenu est fonction du nombre de processeurs, de leur débit de requêtes et du débit des mémoires ou interfaces du bus. Si ces éléments sont correctement équilibrés, le débit de communication peut être très bon. A la limite, chaque processeur peut travailler à sa vitesse propre sans perte de performance.

Deux types d'arbitres peuvent se concevoir :

- . *Arbitres synchrones par multiplexage temporel* : le temps est divisé en tranches dont la durée est égale au temps d'accès élémentaire au bus. Chaque tranche est allouée à un processeur avec une périodicité égale au nombre de processeurs.

Cette méthode est simple et peu coûteuse. De plus, dans le cas où les processeurs ont des cycles lecture/écriture périodique, et où l'on peut faire coïncider les deux périodes (Fig.III.1.2), on obtiendra des performances très élevées (III-1.).

En revanche, cette méthode présente une extensibilité limitée.

En effet, si on accroît le nombre de processeurs partageant le même bus, les performances vont chuter. Cet inconvénient est cependant partagé par toutes les méthodes. Un autre inconvénient spécifique de cette méthode porte sur la possibilité d'étendre le nombre de bus partagés par les processeurs au delà de l'unité. En effet, lorsqu'il y a plusieurs bus partagés (structure matricielle) il paraît difficile de définir une période d'accès d'un processeur à un bus. (Cela en effet dépend des programmes exécutés par chaque processeur). Les séquences d'allocation de chaque bus devront être identiques (Fig.III.1.5). A chaque instant, un seul processeur sera actif sur un seul bus.

Les processeurs continueront à travailler à leur vitesse propre, mais les bus eux seront insaturés et il ne sera pas possible d'augmenter le taux d'occupation des bus, même en augmentant le nombre de processeurs.

D'autre part, du point de vue de la tolérance aux pannes, l'allocateur apparaît comme un point dur du système. En effet, on a vu que même lorsque l'on multiplie les bus partagés, ceux-ci sont régis par la même séquence d'allocation, et donc par le même allocateur. Toute panne de cet allocateur entraînera donc la défaillance du système entier.

. *Arbitres asynchrones* [17] : L'arbitre reçoit les requêtes des processeurs et alloue le bus à un processeur en fonction de celles-ci. Plusieurs types d'arbitres ont été décrits dans la littérature :

- *priorité tournante (Daisy Chaining)* : c'est une méthode peu coûteuse et efficace [18] . Les processeurs sont organisés en boucle et chaque processeur passe la priorité à son voisin lorsqu'il n'utilise pas, ou a fini d'utiliser, le bus. Si les processeurs sont indifférenciés à cycle de lecture/écriture périodique ou quasi périodique on aboutira à une utilisation presque aussi bonne, à la fois du bus et des processeurs que dans le cas du multiplexage temporel; en effet, les processeurs se synchroniseront très vite en déphasant leurs cycles de lecture/écriture.
- *priorité fixe* : cette méthode ne se justifie que lorsque les processeurs et la longueur de leurs accès sont très différenciés. Elle doit se combiner avec une autre méthode lorsqu'il y a des groupes de processeurs similaires.
- *file d'attente* : cette méthode est optimale en ce sens qu'elle minimise l'espérance du temps d'attente de chaque processeur. En revanche, elle est plus coûteuse en matériel et ne se justifie guère que lorsque l'on considère des bus très saturés, ou des cycles de lecture/écriture très aléatoires [à la limite processus poissonnien]. Or, nous avons vu que l'on cherche en général à adapter au mieux le nombre et les performances des processeurs et celles des mémoires et interfaces. D'autre part, pour les micro-processeurs, les cycles de lecture/écriture sont plutôt périodiques ou presque périodiques qu'aléatoires.

Il apparaît donc que les arbitres asynchrones peuvent fournir de bonnes performances. Ils seront cependant plus coûteux en matériel que le multiplexage temporel.

L'extensibilité en nombre de processeurs est limitée par les performances des mémoires et unités d'entrée/sortie, l'extensibilité en nombre de bus conduit très vite à une utilisation insuffisante des organes passifs. En revanche, on peut accroître simultanément le nombre de processeurs et le nombre de bus avec une utilisation satisfaisante des deux types d'organe actifs et passifs.

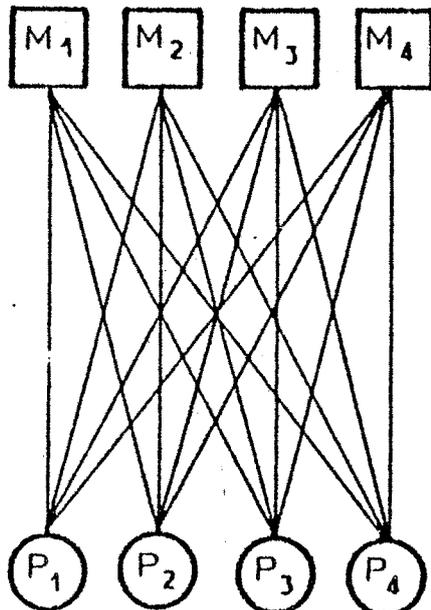
Enfin, cette méthode présente d'excellentes propriétés de tolérance aux pannes puisque chaque bus partagé est muni de son arbitre et que tous ces arbitres sont indépendants.

I - 1.2. Types d'architecture

Outre le mode de connexion, les architectures multimicro-processeur se caractériseront par :

- La structure d'interconnexions : celle-ci peut se représenter par un graphe simple dans le cas de communications processeur à processeur, par un graphe biparti dans le cas de communications par mémoire commune. Les graphes considérés pourront être orientés ou non. En général, les calculateurs d'usage général auront un graphe plus "régulier" que les calculateurs spécifiques (réseau complet, arbre). Par exemple, un réseau complet 4x4 avec communication par mémoire commune correspond à ce que l'on appelle la structure matricielle :

Graphe Biparti



Représentation Matricielle

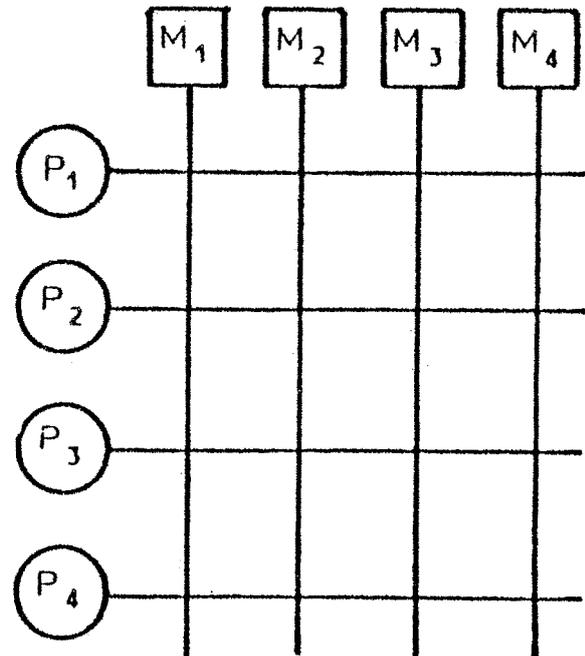


Figure I.1.

- *La spécialisation des processeurs* : les processeurs peuvent être banalisés ou spécialisés en fonction de tâches précises : processeurs d'entrée-sortie, processeurs de contrôle, processeurs opératifs, arithmétiques, processeurs maître et esclaves, Le choix dépend là encore du caractère plus ou moins spécialisé de l'architecture à réaliser.

I - 2. TOLERANCE AUX PANNES

I - 2.1. Définition de la tolérance aux pannes

Le terme "tolérance aux pannes" regroupe plusieurs notions (cf. les actes des Congrès FTC des dernières années [3][4][5] et diverses thèses [6][7]).

Cependant, on peut dégager deux notions importantes :

- la sécurité, ou probabilité qu'une panne ne provoque pas d'erreurs (elle peut cependant conduire à l'arrêt du système) ;
- la disponibilité, ou probabilité qu'une panne ne provoque pas l'arrêt du système (elle peut cependant provoquer un fonctionnement erroné transitoire).

Après un rappel sur les différentes techniques permettant d'améliorer l'une ou l'autre de ces grandeurs, nous étudierons la facilité de mise en oeuvre de ces techniques dans des structures modulaires.

I - 2.2. Système à haute sécurité - Système à haute disponibilité

On peut classer les systèmes tolérant les pannes en deux classes :

- Les systèmes à haute sécurité (avionique, espace, ...) dont les sorties doivent avoir une faible probabilité d'être erronées ; les erreurs sont donc détectées, dès leur occurrence, par redondance massive (duplication) ou sélective (codes détecteurs). La disponibilité de tels systèmes peut être améliorée par masquage des erreurs, c'est-à-dire là où les premières pannes ne conduisent pas à l'arrêt du système (TMR, NMR pour la redondance massive ; codes correcteurs pour la redondance sélective). Cette option est coûteuse en matériel et n'est utilisée que si l'application l'exige.

- Les systèmes à haute disponibilité (télécommunication, téléréservation, ...) qui doivent avoir une faible probabilité d'être hors service ; avant la détection d'une panne, les sorties peuvent toutefois être transitoirement erronées.

La disponibilité peut être assurée :

- + soit par l'utilisation de rechanges : après détection d'une panne dans un composant, un composant semblable (ou un ensemble de composants pouvant réaliser les mêmes fonctions) est mis en service. Cette méthode peut s'appliquer indifféremment aux architectures classiques et aux architectures distribuées ; dans le deuxième cas, le composant de rechange sera généralement une unité. Cette méthode est coûteuse en matériel : matériel de rechange utilisé rarement.
- + soit par la possibilité de dégradation progressive : un sous-système en panne est déconnecté et le système continue à fonctionner avec le matériel restant, après reconfiguration logicielle. Cette méthode est la moins coûteuse et s'applique surtout aux architectures distribuées ; mais elle implique une diminution des performances du système après panne.

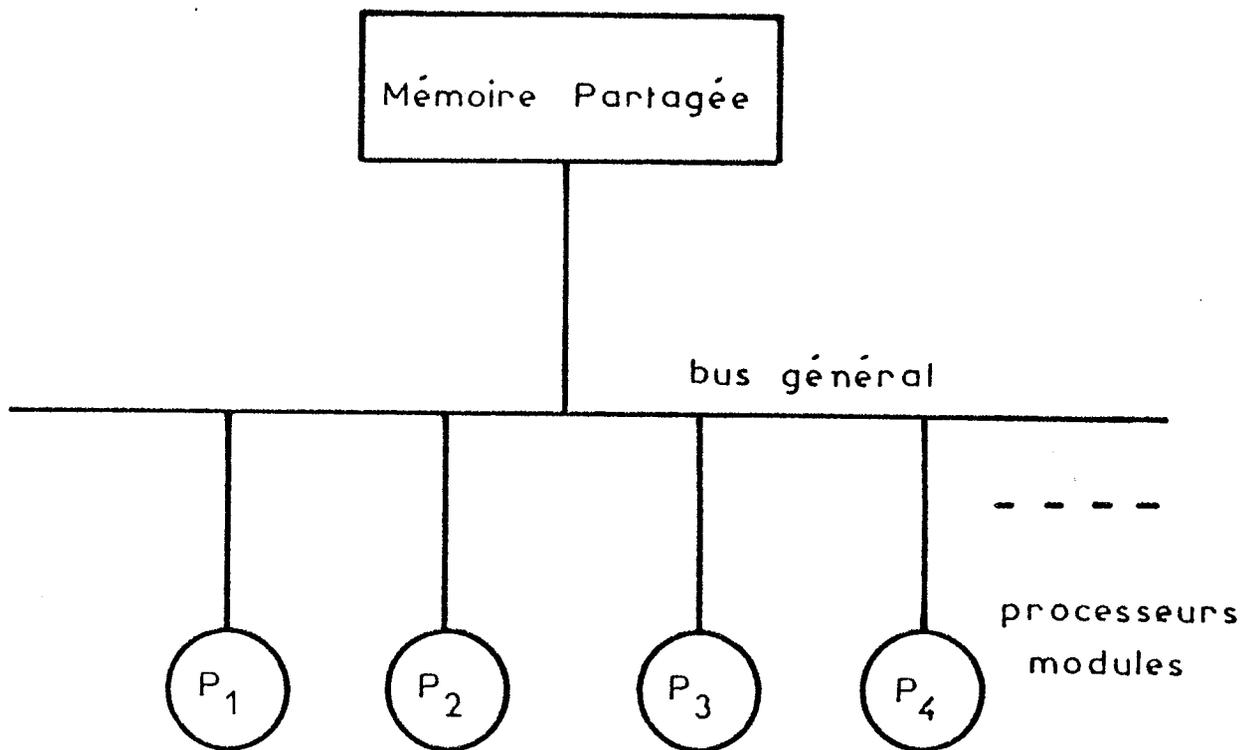
I - 2,3. Intérêt des structures modulaires

a) Structures modulaires

Un système multiprocesseur est un système composé de processeurs interconnectés; un système multiprocesseur modulaire est un système dans lequel les processeurs sont banalisés :

- ils sont du même type,
- ils ont accès aux mêmes ressources.

Chaque processeur peut donc exécuter toutes les fonctions du système; la puissance désirée est obtenue par le travail simultané de tous les microprocesseurs. Un système modulaire général est représenté en (Fig.I.2). Les systèmes modulaires peuvent être opposés aux systèmes distribués [7], dans lesquels chaque processeur est spécialisé dans l'exécution d'une ou plusieurs fonctions.



Systeme Multiprocesseurs Modulaire

Figure I.2.

b) Replication dans les structures modulaires

L'implémentation de redondance est relativement simple et efficace dans un système multiprocesseurs modulaire;

- une redondance au niveau d'un système complet est difficile à maîtriser (synchronisation, propagation des erreurs) et peu efficace au niveau de la fiabilité [7];
- une redondance au niveau fin (registres, UAL..) n'est pas comparable avec l'intégration croissante des composants et nécessite, en fait, l'étude de composants "à la demande". A l'heure actuelle, la solution raisonnable est d'implémenter la redondance au niveau des composants "LSI" disponibles (microprocesseurs, banc de mémoire,...), donc à un niveau moyen.

Dans un système modulaire, deux ou plusieurs modules seront donc groupés pour implémenter une redondance. Ce groupement peut être :

- figé : par du matériel supplémentaire (comparateur pour la duplication, voteur pour la triplification);
- dynamique par le logiciel : à des phases déterminées du programme, deux ou trois processeurs comparent leurs résultats. Les processeurs travaillent en redondance sont choisis par le logiciel, donc ce choix peut varier suivant la dégradation du système et l'avancement de ces tâches.

Les degrés de redondances "réalistes" au niveau interprocesseurs sont :

- la duplication
- la triplification

c) Dégradation progressive dans les structures modulaires

Un système admet une dégradation progressive en présence de pannes lorsqu'il peut être automatiquement reconfiguré, logiciellement et matériellement après la détection d'une panne du matériel, sans avoir recours à du matériel en réserve [7].

Dans un système modulaire, la reconfiguration matérielle consiste à supprimer l'unité défectueuse; la reconfiguration logicielle consiste simplement à répartir la charge primitive sur les modules restants; le degré de parallélisme diminue, ainsi que les performances du système qui restent cependant opérationnelles.

d) Dégradation progressive dans les structures modulaires redondantes

Un système admettant une dégradation progressive doit assurer une bonne détection des pannes, pour atteindre un certain niveau de sécurité (pas d'erreurs) ou de disponibilité (erreurs transitoires seulement).

Nous étudierons ici la méthode la plus systématique de détection : la redondance. Les systèmes étudiés seront donc des systèmes modulaires, où une redondance est implémentée, admettant une dégradation progressive (algorithme flexible d'allocation des tâches aux groupes de processeurs).

On distingue quatre organisations possibles :

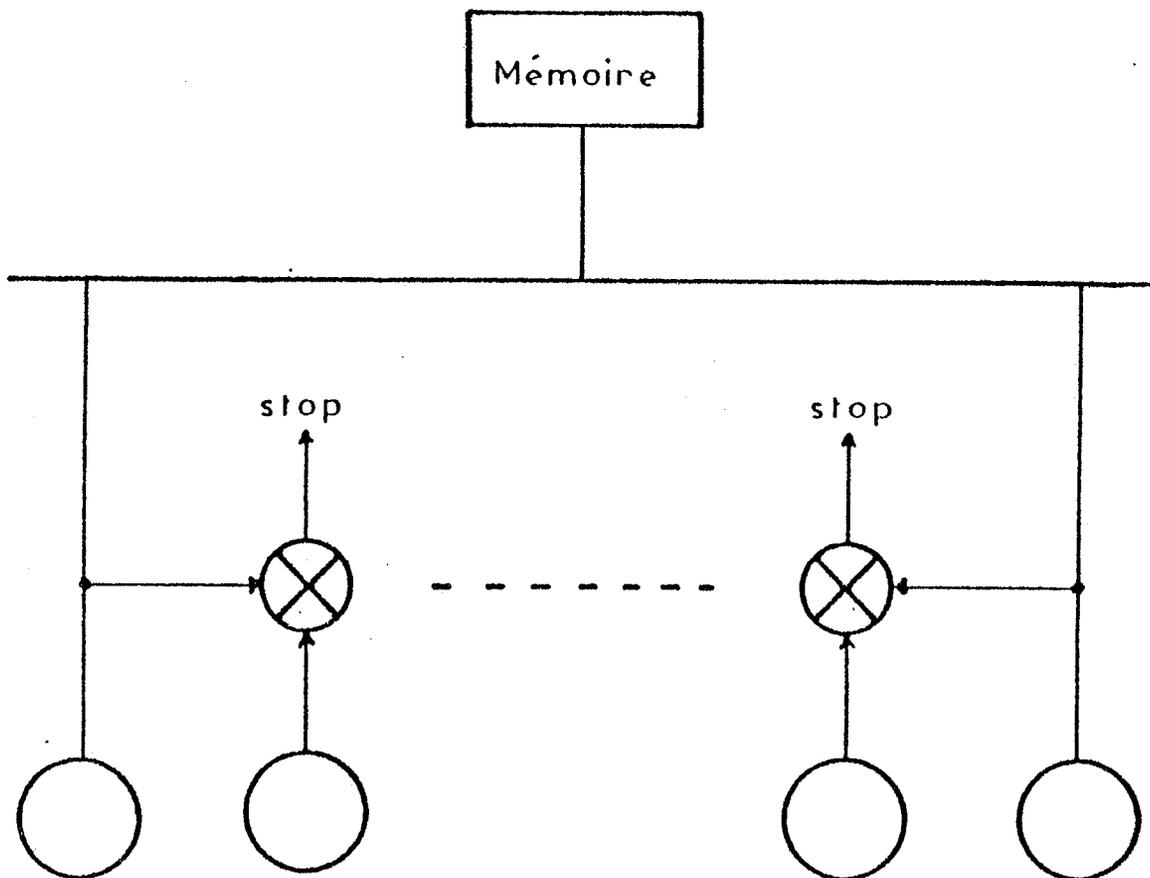
La duplication figée : employée dans [8] [9] [10], consiste à jumeler par construction deux processeurs (Fig. I.3); quand il y a désaccord, on supprime les deux processeurs jumelés sans chercher lequel est en panne. Une panne élimine donc deux processeurs.

La duplication dynamique [11] : consiste à former des couples de processeurs par logiciel; en cas de désaccord, on fait appel à un troisième processeur pour déterminer le processeur en panne; celui-ci est supprimé. Le processeur restant est jumelé à un processeur libre s'il y en a; si non il est déclaré libre. S'il y a n couples à l'origine, après une panne il n'en reste que $n-1$; après deux pannes il y a toujours $n-1$ couples (au lieu de $n-2$ pour la duplication figée).

La triplification figée [12] : consiste à déterminer des triades de processeurs à la construction; un voteur majoritaire compare les sorties des trois processeurs; s'il y a désaccord, le résultat choisi est le résultat "majoritaire". Après une panne, soit la triade est supprimée, soit elle continue à fonctionner avec deux processeurs (en duplication). S'il y a n triades au départ, après une panne il en reste $n-1$ (ou $n-1$ et un couple); après deux pannes $n-2$ (ou $n-2$ et 2 couples).

La triplification dynamique [13] : consiste à former des triades par un choix logiciel; s'il y a désaccord, le processeur en panne est déterminé par vote majoritaire, puis supprimé. S'il existe un processeur libre, une triade est reformée; si non les deux processeurs restants sont déclarés libres et mis en réserve). Après trois pannes on dispose encore de $n-1$ triades sur les n dont on disposait initialement; de plus, on interdit tout fonctionnement en duplication qui nécessiterait une stratégie de point de reprise et re-exécution.

Les performances de tels systèmes, en tenant compte de la dégradation progressive de puissance, ont été analysées et comparées en [7]. Il en ressort que la duplication dynamique est une solution intéressante à la fois par ses performances et son coût.



Système à duplication figée

Figure I.3.

CHAPITRE II

DEFINITION DU SYSTEME

-0-0-0-

II - 1. PRESENTATION

La définition complète d'une architecture multimicroprocesseurs est le fruit de la suite des choix effectués durant sa conception. Dans ce chapitre nous présenterons cette suite de choix en respectant l'ordre chronologique dans lequel ils se sont posés. Pour chacun d'entre eux nous indiquerons un éventail de solutions possibles, ainsi que les motivations de nos décisions. Nous avons classé ces choix sous deux rubriques :

- choix relatifs aux principes mêmes de la structure
- choix d'une architecture particulière répondant à ces principes.

Par les premiers nous définissons une "famille de machine", par les seconds nous sélectionnons dans cette famille l'architecture effectivement expérimentée.

II - 2. ELEMENTS DU CHOIX D'UNE STRUCTURE

a) Choix d'un matériel

Les architectures visées sont constituées par l'interconnexion de divers types d'éléments (mémoires, processeurs, interface d'Entrée/sortie,..

Deux solutions vont alors s'affronter :

- 1) choix de quelques représentants de chacun des types précités et définition de leur interconnexion éventuelle au coup par coup;
- 2) choix d'une interface normalisée pour chacun des types précédents et définition d'interconnexion standard entre ces types ne tenant compte que de cette interface normalisée.

Une fois ce travail effectué, l'introduction d'un nouvel élément d'un certain type au sein d'une telle structure se limitera à la réalisation, autour de cet élément, de circuit d'adaptation à l'interface normalisée de son type.

Cette approche est notamment celle proposée par J. Nicoud [14] et par Faiman [15]. Elles se résument en la définition de bus standards normalisant les échanges mémoires - processeurs (MUBUS, MUMS).

Remarquons que la deuxième approche est souvent réservée à des petits systèmes destinés à l'enseignement; plusieurs exemples de microprocesseurs peuvent alors être étudiés et comparés (système DOLPHIN).

Cette solution demande une étude complète du matériel disponible, étude qui semble en dehors de notre propos. Elle conduit également à une complication du matériel utilisé et à une multiplication des boîtiers (circuits d'adaptation), contradictoires avec notre objectif de tolérance aux pannes.

De plus, l'utilisation effective d'un microprocesseur nécessite un important matériel d'aide au développement. L'absence où le petit nombre de ces matériels risque de limiter dans la pratique les possibilités offertes par cette deuxième solution.

En outre, nous nous intéresserons à des gammes de matériels aux performances relativement proches qui seront encore uniformisées par l'emploi d'interfaces normalisées.

Cette deuxième solution, séduisante au premier abord, a donc été abandonnée d'une part à cause des objectifs poursuivis et d'autre part à cause de l'environnement limité dans lequel cette étude devrait être menée.

La présence dans cet environnement d'un outil de mise au point EXORCISER 6800 nous a conduit à choisir le microprocesseur 6800.

b) Choix d'un mécanisme de communication mémoires ↔ processeurs

Il s'agit d'un point crucial de la structure Mazaré [1]: il conditionne à la fois :

- * les performances fonctionnelles de la structure
 - . débit d'échange entre éléments
 - . facilités de communication
 -
- * les performances opérationnelles (sûreté)
 - . possibilité de test mutuel
 - . possibilité de reconfigurations
 -
- * les possibilités d'extension de la structure.

Parmi les objectifs poursuivis figurent la non spécialisation de la machine et son aspect haute performance (introduction). Ces objectifs nous imposent la transparence au niveau logiciel et nous suggèrent un mécanisme de dialogue interprocesseurs par mémoires communes.

La structure recherchée apparaît alors comme un groupe de processeurs se partageant un ensemble de bus auxquels sont connectés les éléments passifs de la structure (mémoires interfaces d'entrée/sortie) (Figure II.1).

On aboutit ainsi à une structure de type matricielle (Cf. Chap. I.1) dont la tolérance aux pannes a déjà été étudiée sur de nombreux exemples (SIFT, CMMP), [13][12].

c) Choix d'un mécanisme d'allocation

Ici intervient un choix fondamental pour l'orientation de ce travail. En effet, nous avons choisi d'utiliser un mécanisme d'allocation par multiplexage temporel, en dépit de ces nombreux inconvénients (Cf. Chap. I.1.1),

- point dur du point de vue de la tolérance aux pannes
- faible extensibilité.

Les raisons de ce choix sont les suivantes :

* Tout d'abord les arbitres asynchrones ont déjà été utilisés de façon intensive dans les systèmes multiprocesseurs existants, et leurs possibilités sont actuellement bien connues. En revanche, le multiplexage temporel n'a guère été utilisé que dans des structures restreintes du type bi-processeur.

* D'autre part, cette solution s'adapte bien au microprocesseur choisi M 6800 étant donné le caractère périodique de ses cycles de lecture écriture. On peut donc espérer bâtir un allocateur simple, compact au point de vue matériel et facile de mise au point.

Il nous a donc paru intéressant de choisir ce type de mécanisme et ainsi de fonder ce travail sur l'étude approfondie de cette technique et sur la recherche de solutions originales, tant du point de vue du nombre de processeurs partageant un bus, que des possibilités d'extensions à des réseaux complexes.

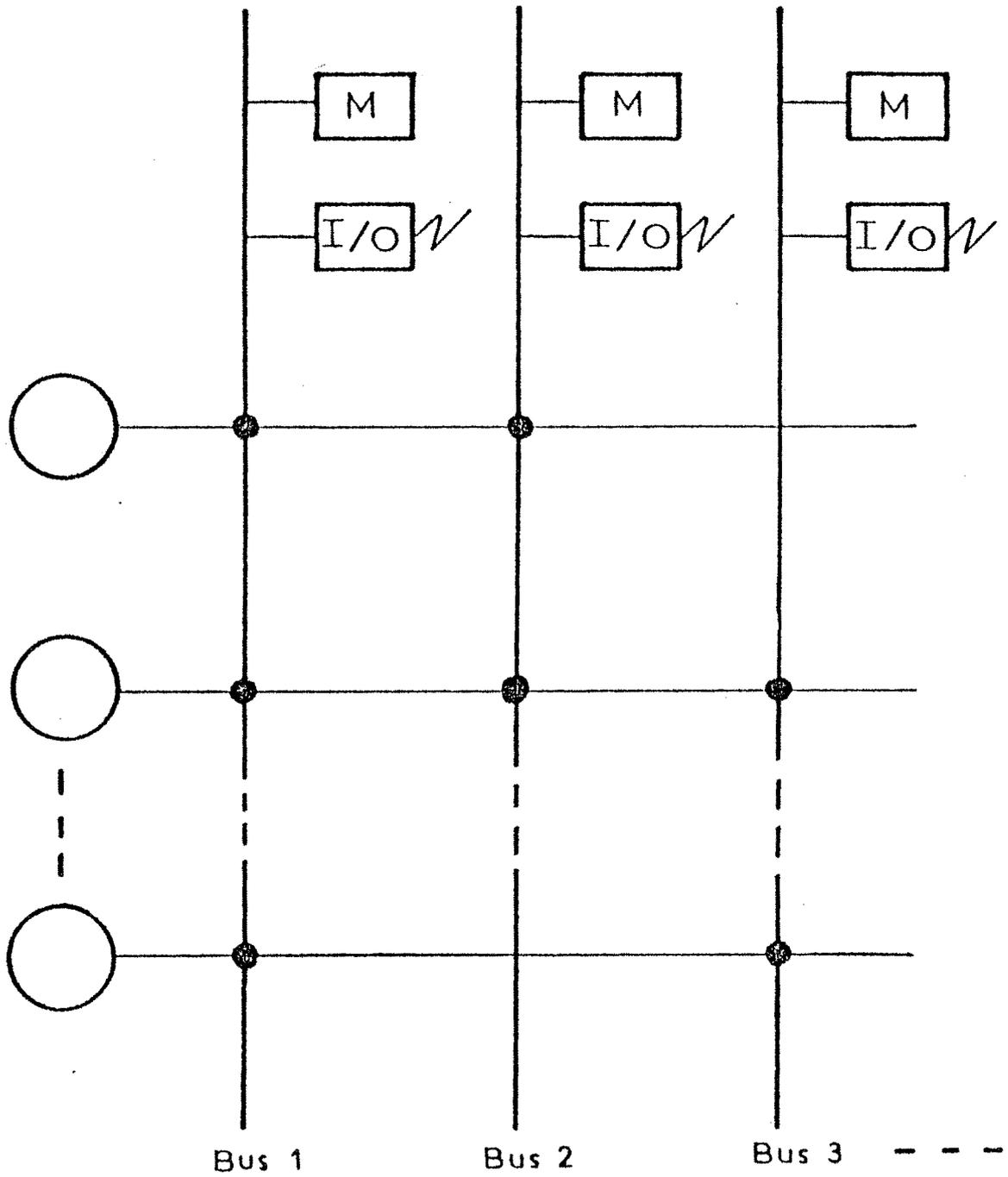


Figure II.1.

II - 3. PRESENTATION ET CHOIX DE L'ARCHITECTURE RETENUE

La machine "4 M" est un quadriprocesseur. La rapidité des communications inter processeurs est assurée par des zones communes aux espaces d'adressage de plusieurs processeurs.

II - 3.1. Adressage et mémoire logique

Les quatre processeurs de "4 M" sont répartis en deux groupes : (A_1, A_2) et (B_1, B_2) respectivement.

Chaque processeur accède à un espace mémoire de 64K octets, mais les deux processeurs d'un même couple se partagent un même espace mémoire. Les deux espaces d'adressage correspondant au deux groupes de processeurs ne sont pas disjoints; ils se recouvrent sur une zone de 32K octets (Fig.II.2).

L'espace mémoire total de "4 M" se répartit donc en 3 zones :

- 1) zone M_2 propre au groupe (A_1, A_2)
- 2) Zone M_1 partagée par les deux groupes
- 3) Zone M_3 propre au groupe (B_1, B_2) .

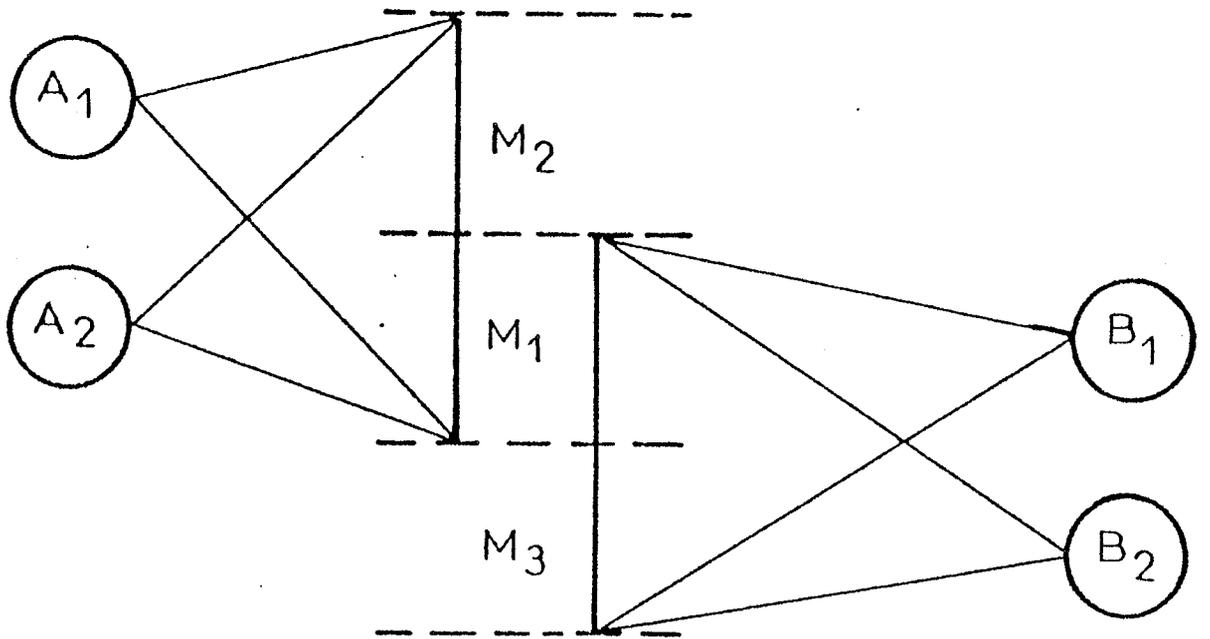
Ces diverses zones peuvent être implémentées totalement ou partiellement, au niveau physique, suivant la configuration désirée.

Au niveau logique, ces zones de mémoires se distinguent par les ensembles de processeurs qui y ont respectivement accès. Par contre, au niveau physique, ces mêmes zones mémoires vont se distinguer par les différents temps d'accès que leur imposera la structure "4 M".

II - 3.2. Mémoire physique et protection

La version de base de la machine "4 M" devait être orientée vers les hautes performances. Cet objectif se traduit, au niveau de la réalisation, par deux caractéristiques :

- * vitesse maximale des processeurs
- * partage mémoire transparent au niveau logique (logiciel).



Mémoire Logique de "4M"

Figure II.2.

Ce second point imposait donc l'existence d'un mécanisme de partage des diverses zones mémoires entre les processeurs. Nous avons choisi un multiplexage temporel de l'accès à l'espace physique.

Le premier point conduit alors, d'après le paragraphe précédent, à des circuits mémoires ayant des temps d'accès différents suivant la zone mémoire à laquelle elle appartient :

- * un quart de cycle machine pour la zone M_1 (4 processeurs se partagent cette zone),
- * un demi cycle machine pour les zones M_2 et M_3 (un seul groupe de processeurs se partage chacune de ces zones).

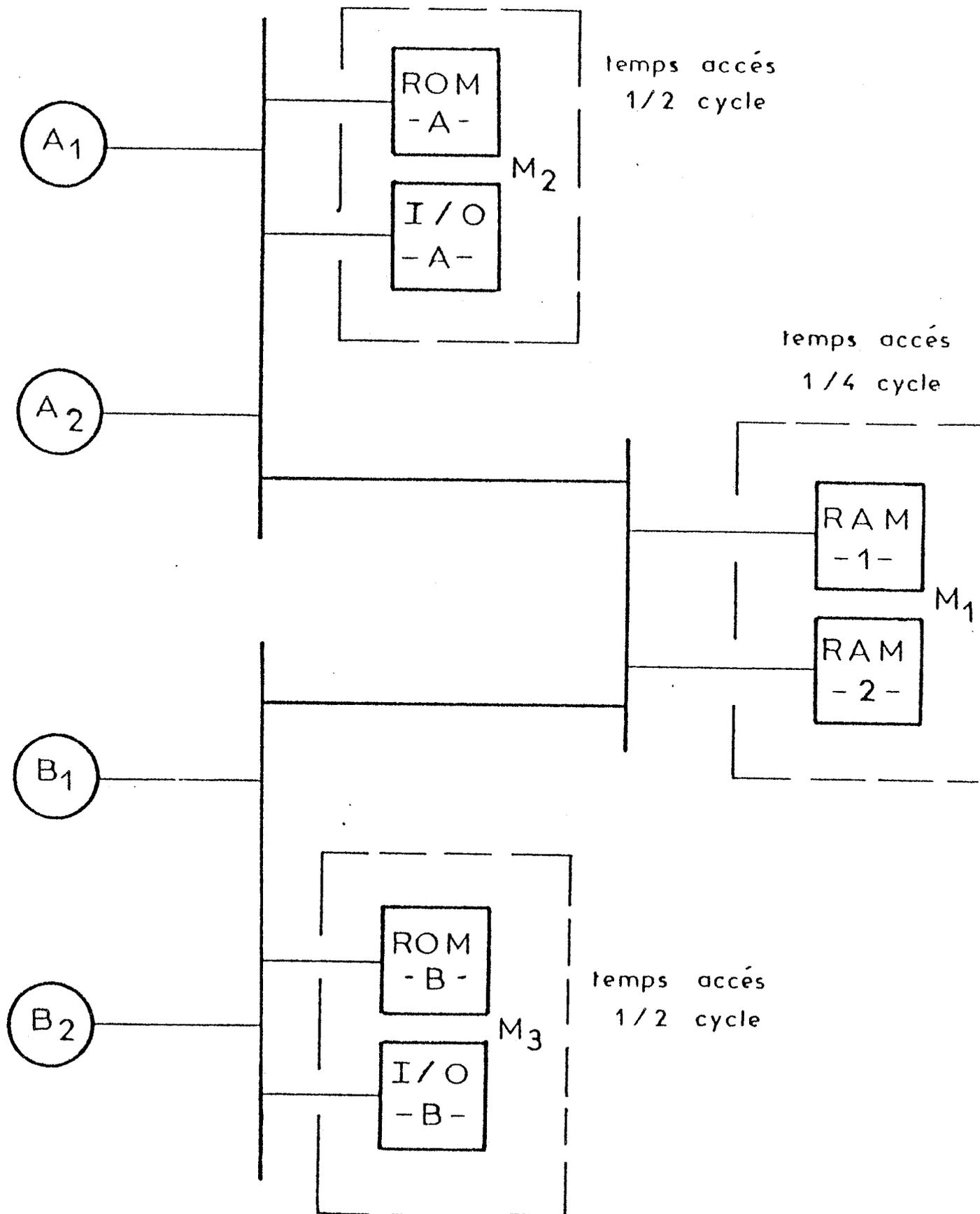
Ces rapports temps accès/temps de cycle restent tout à fait cohérent avec les circuits disponibles actuellement. (La description complète de ce mécanisme est donnée au paragraphe III-1.).

Dans notre réalisation physique de ce système la zone M_1 est entièrement formée par des mémoires à accès direct (RAM) alors que les zones M_2 et M_3 ne contiennent que de la mémoire morte (ROM) et des interfaces d'entrée/sortie (Fig. II.3). Mais ce choix peut être facilement modifié si l'on possède des circuits de mémoires ou d'entrée/sortie ayant des temps d'accès compatibles avec la zone d'implantation désirée.

Un système de protection a également été implanté. Cette protection concerne la mémoire logique. Cet espace d'adressage est divisé en pages (taille variable suivant les versions de "4 M" et suivant la zone considérée : M_2 M_3 ou M_1). Pour chaque page et pour chaque processeur adressant cette page deux variables logiques indiquent les modes d'accès autorisés (lecture, écriture) pour le processeur concerné. Suivant l'option choisie, ces variables peuvent être, soit fixées par câblage, soit adressables et donc modifiables dynamiquement par les processeurs.

Ce dispositif classique de protection répond à deux objectifs :

- isolation mutuelle de certains programmes,
- facilite la gestion des bus mémoire (par exemple inhibition des accès au ROM en écriture) (Chapitre III).



Mémoire Physique du Système Réalisé

Figure II.3.

II - 3.3. Choix de cette architecture

Le choix le plus important effectué concerne la limitation à quatre du nombre de processeurs.

- 1) Cette limitation était imposée, au début de notre étude, par les circuits couramment disponibles dans le commerce et en particulier par les rapports temps de cycle microprocesseur, temps d'accès mémoire atteignables. Aujourd'hui, compte tenu de l'avance technologique, on peut penser à des "octo-processeurs" (voire même plus).
- 2) Ce nombre de processeurs reste assez élevé pour valider, de façon réaliste, ces structures sur le plan matériel.
- 3) Un des objectifs de notre étude était la réalisation d'une structure capable d'assurer une certaine tolérance aux pannes. Quatre processeurs et une spécialisation par logiciel suffiront à essayer de façon significative la plupart des solutions classiques de tolérance aux pannes (TMR, Bi duplex, 4 MR,...) (I.2.).

La division en plusieurs zones de mémoire de temps d'accès différent nous est imposée par le besoin d'utiliser divers matériel de temps d'accès différent (RAM, ROM, interface d'Entrée/sortie).

Néanmoins, celle-ci nous suggère un fonctionnement par paire qui est très favorable, tant à la haute sécurité (comparaison au sein d'une paire) qu'à la haute disponibilité (localisation des pannes au sein d'une paire).

CHAPITRE III

DESCRIPTION DETAILLEE DU SYSTEME

-0-0-0-

III - 1. RESOLUTION DES CONFLITS D'ACCES MEMOIRE PAR MULTIPLEXAGE TEMPOREL

Le mode d'accès mémoire du microprocesseur 6800 facilite grandement la mise en oeuvre de ce multiplexage.

III - 1.1. Accès mémoires du M 6800

Tout le séquençement interne du MC 6800 se synchronise sur deux signaux d'horloge externe : les signaux $\phi 1$ et $\phi 2$.

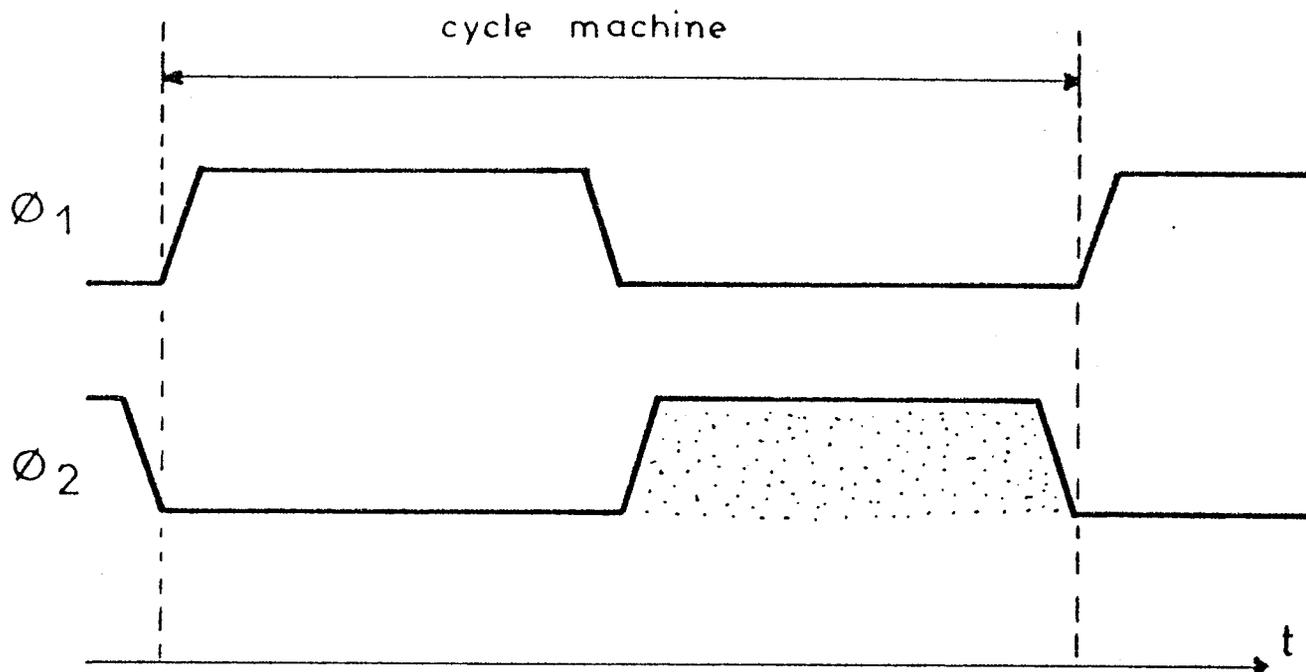


Figure III.1.1.

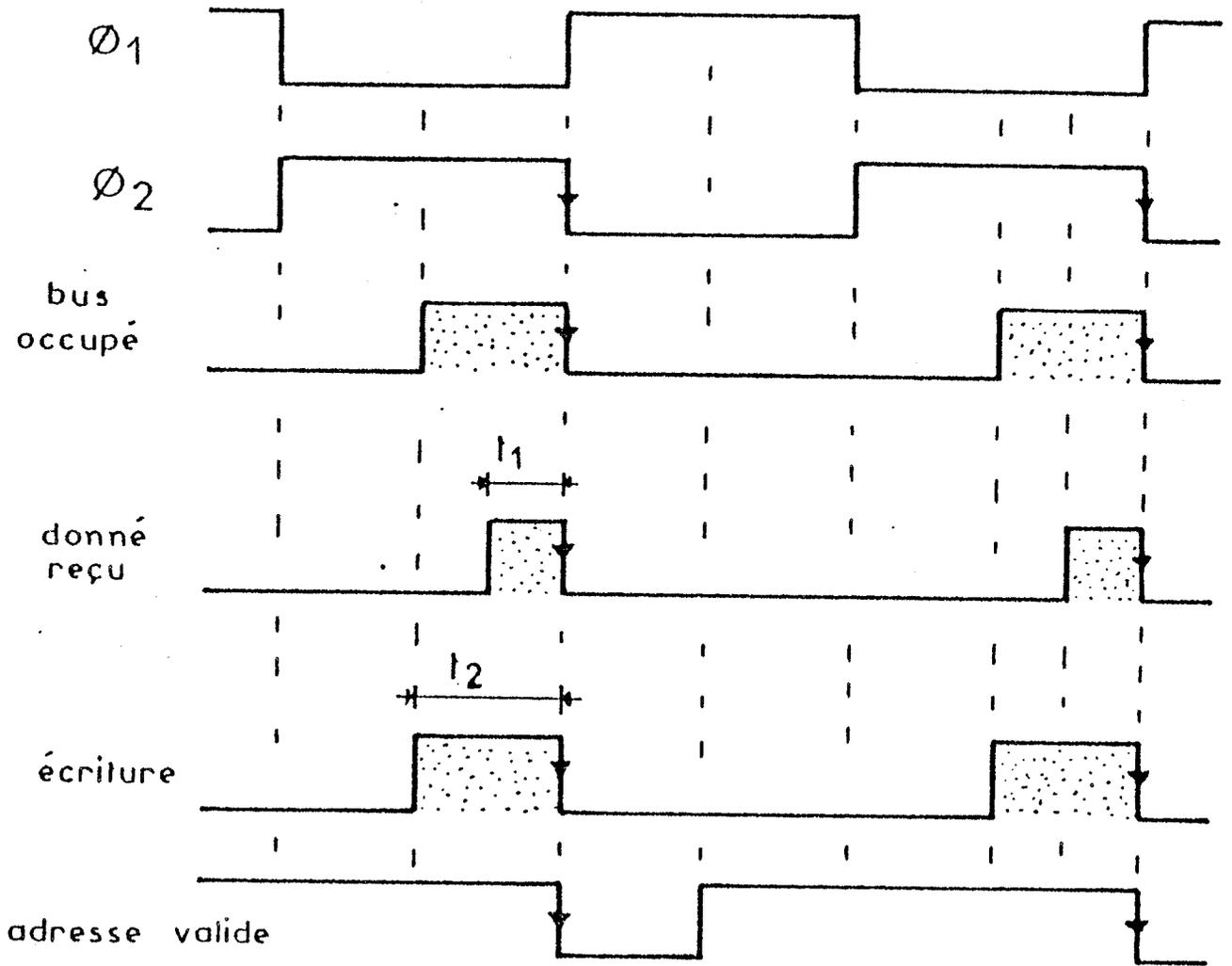


Figure III.1.2.

Ces signaux sont périodiques, identiques mais déphasés d'une demi-période l'un par rapport à l'autre.

Le fonctionnement (usuel) du processeur est formé d'une suite d'exécution d'instructions de son répertoire.

Chaque exécution d'instruction comprend un nombre entier de cycles machine successifs. Chaque cycle comprend deux phases ϕ_1 et ϕ_2 indiquée par l'état "haut" de leur horloge respective ϕ_1 et ϕ_2 (Fig.III.1.1).

Tous les accès-mémoire sont compris dans un seul cycle machine. L'adresse mise en cause est présente dès le milieu de la phase ϕ_1 . S'il s'agit d'une lecture, l'organe sélectionné (mémoire ou interface d'entrée/sortie) doit présenter la valeur du mot adressé un huitième de cycle avant la sortie de la phase ϕ_2 du cycle machine (voir Fig. III.1.2)

S'il s'agit d'une écriture, le processeur positionne la valeur à écrire pratiquement à partir du milieu de la phase ϕ_1 . Cet intervalle est donc pleinement disponible pour le cycle écriture de l'organe adressé.

III - 1.2. Multiplexage temporel

III - 1.2.1. Système biprocesseur

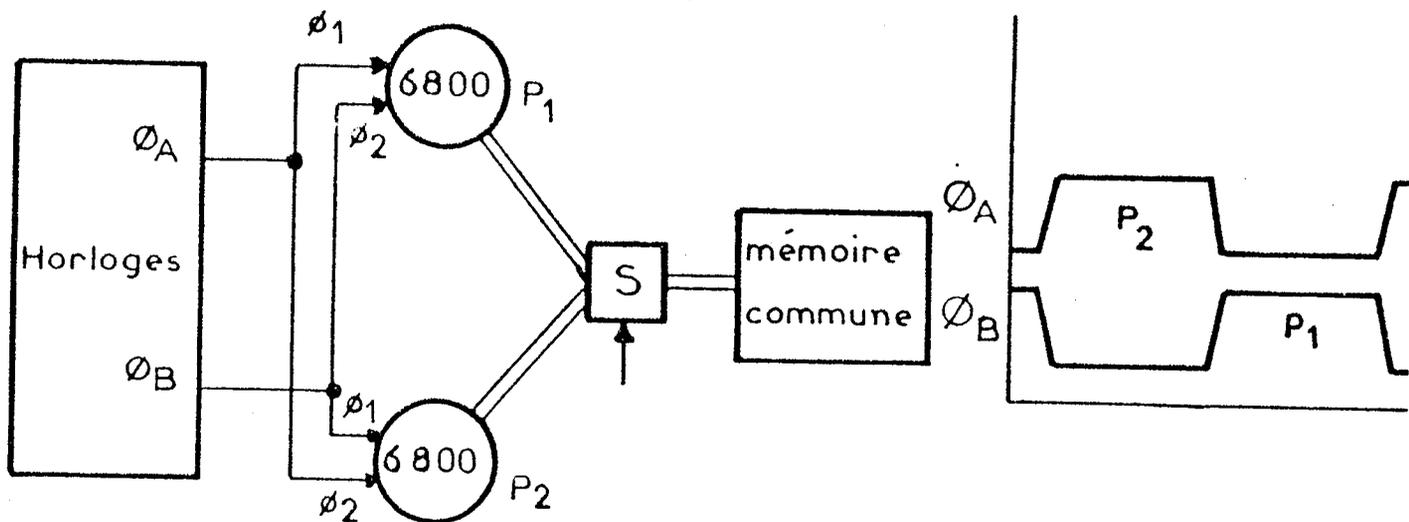


Figure III.1.3.

Considérons le système représenté par la figure III.1.3. :

- La phase Φ_1 du processeur P1 correspond à la phase Φ_2 du processeur P2 et réciproquement.
- Le multiplexeur/démultiplexeur S permet de connecter la mémoire commune au processeur P1 si Φ_B est à l'état haut et au processeur P2 si Φ_A est à l'état haut.

Les deux processeurs seront donc électriquement reliés à la mémoire commune durant la phase Φ_2 de chacun de leur cycle machine.

Si la mémoire a un temps d'accès inférieur à la durée d'une phase, soit environ un demi-cycle machine, les deux processeurs accéderont alternativement à la mémoire commune sans être ralentis par les accès de l'autre processeur.

III - 1.2.2. Système quadriprocesseurs

On peut remarquer, sur la figure III.1.4, que si les mémoires utilisées ont un temps d'accès inférieur à un quart de cycle machine, la généralisation du mécanisme précédent à quatre processeurs est possible.

Il suffit, pour cela, que les horloges du processeur i ($i \in \{1,2,3,4\}$) soient retardées d'un quart de cycle machine par rapport aux horloges correspondantes du calculateur $[i-1]$ modulo-4.

Les accès mémoires des processeurs, effectués tous dans la seconde moitié des phases Φ_2 correspondantes, ne se recouvriront pas dans le temps. Le multiplexeur/démultiplexeur S se bornera à allouer le bus mémoire, successivement à chaque processeurs, tous les quart de cycle (Fig.III.1.5).

III - 1.2.3. Généralisation

On peut envisager des généralisations à plus de quatre processeurs, de ce dispositif. De telles structures impliquent l'utilisation de circuits mémoires ayant des temps d'accès bien inférieurs au quart du temps de cycle machine. Mais la limitation la plus contraignante viendra du temps d'acquisition des données (en lecture) de la part du processeur (temps t_1 sur la figure III.1.2.)

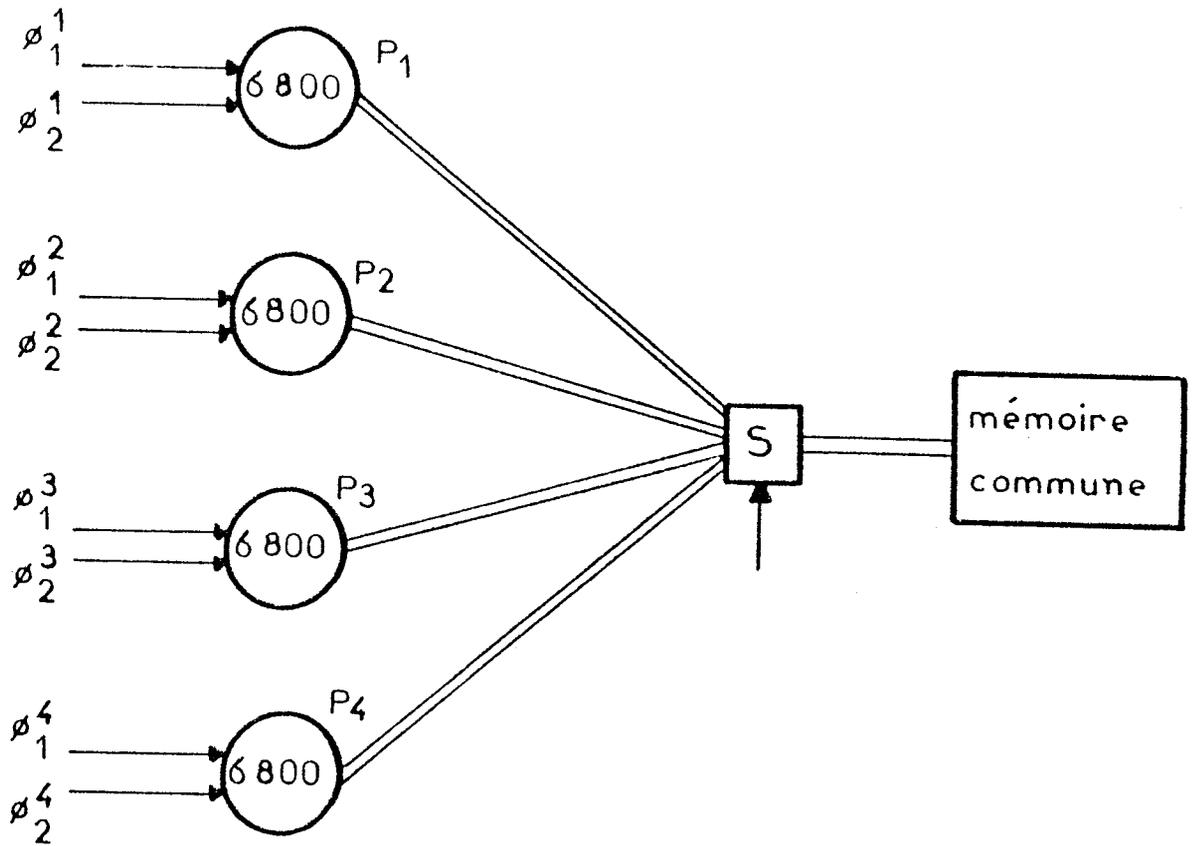


Figure III.1.4.

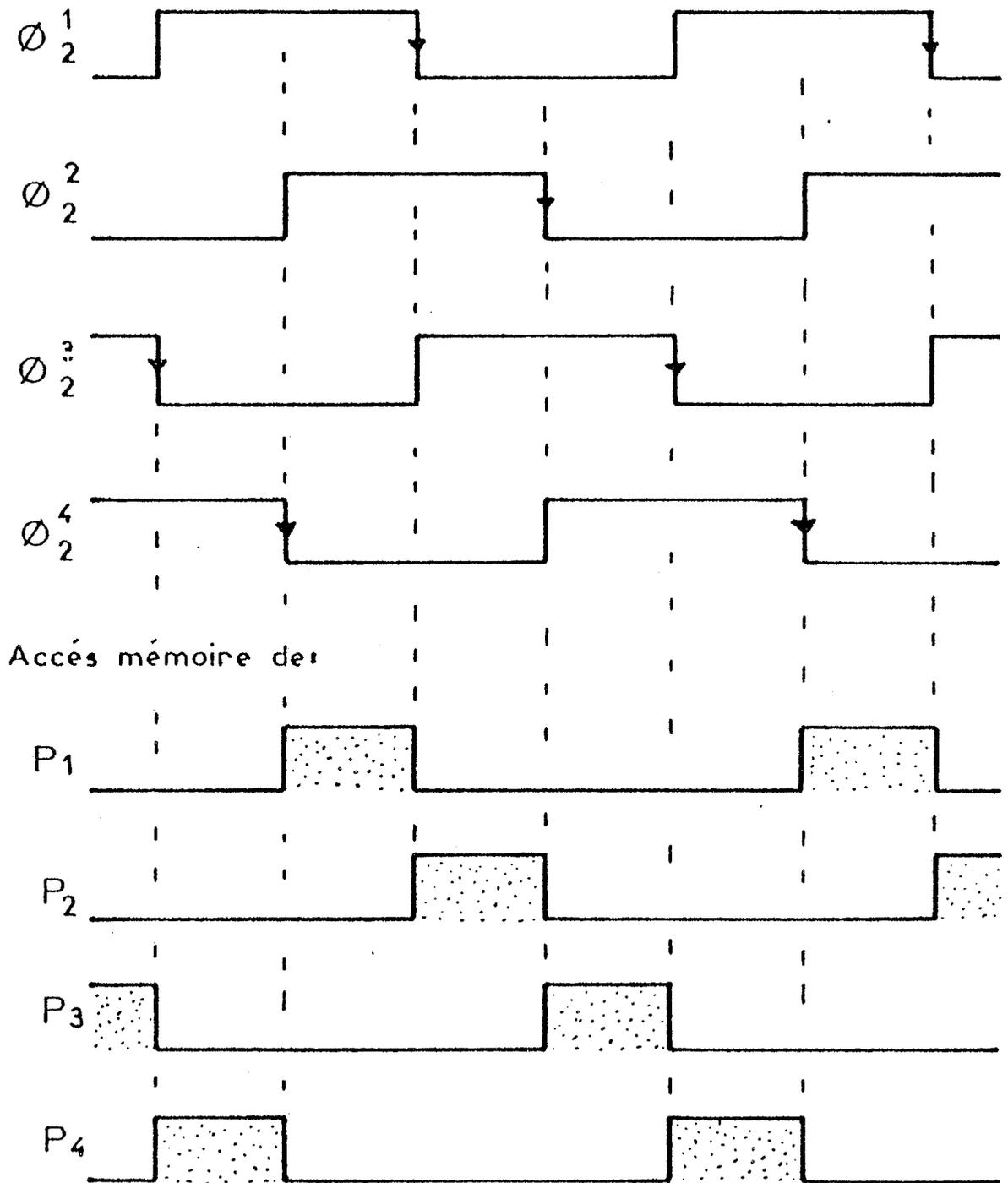


Figure III.1.5.

Quelques techniques peuvent être employées pour surmonter cet inconvénient. Par exemple, on peut rajouter un niveau de mémorisation rapide supplémentaire entre processeurs et multiplexeur (Fig.III.1.6).

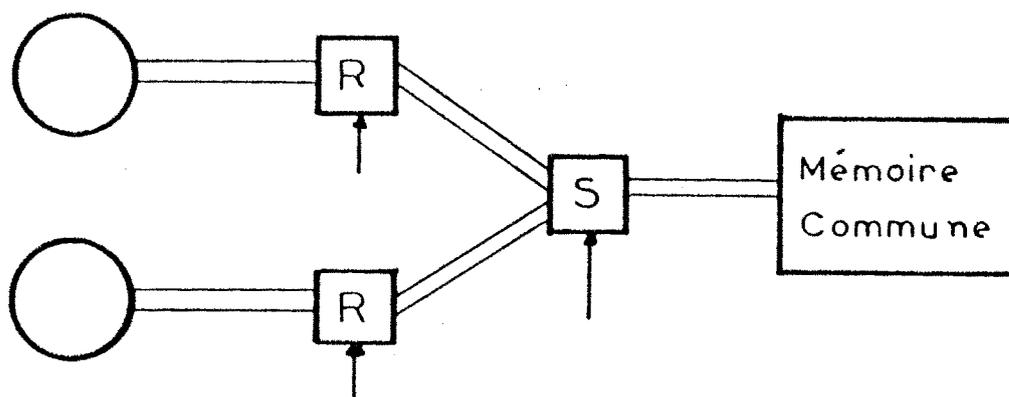


Figure III.1.6.

La réponse la plus simple à ce problème reste l'utilisation de processeurs plus rapides aujourd'hui disponibles. Le temps d'acquisition t_1 est alors abaissé jusqu'à des valeurs comparables à celles de technologies rapides (TTL).

On remarque, également, que dans le cas d'utilisation de mémoires très rapides, le temps de propagation des signaux à travers le multiplexeur S devient prédominant.

III - 1.2.4. Solution mixte

Comme nous l'avons vu, le partage du bus mémoire par multiplexage temporel établit une relation étroite entre le nombre de processeurs et le temps d'accès des circuits mémoires utilisés.

Afin d'utiliser des mémoires de temps d'accès, et par conséquent de coût différents, ainsi que des circuits d'interfaces d'entrée/sortie, nous avons retenu pour "4 M" une structure mixte entre les solutions à deux ou à quatre processeurs.

Nous avons choisi une structure à quatre processeurs MC 6800 que nous avons réparti en deux groupes, (P_A^1, P_A^2) et (P_B^1, P_B^2) .

A l'intérieur de chaque groupe, les processeurs voient leur horloges respectives décalées de un demi-cycle machine. Les deux processeurs d'un même groupe ont donc des signaux horloges analogues à ceux des deux processeurs du système bi-processeur.

D'un groupe à l'autre, les signaux horloges correspondants se déduisent l'un de l'autre par un déphasage d'un quart de cycle machine. L'entrelacement des horloges ainsi obtenu est représenté par le diagramme suivant (horloge Φ_2).

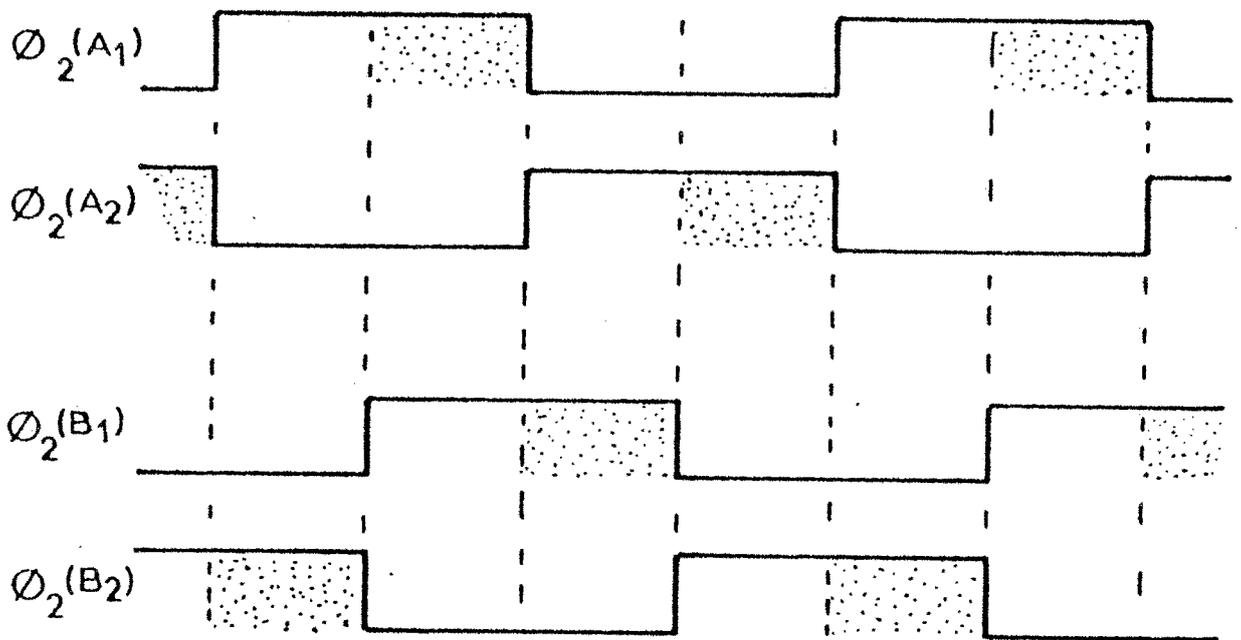


Figure III.1.7.

On remarque donc que l'on retrouve ainsi :

- le diagramme des horloges d'un système biprocesseur pour les deux calculateurs d'un même groupe,
- le diagramme des horloges d'un système quadriprocesseur pour la structure globale.

Nous avons donc défini deux types d'espaces mémoire (physique)

(Fig. III.1.8) :

- un espace mémoire commun à tous les processeurs et donc formé par des circuits mémoires à temps d'accès inférieur au quart de cycle machine,
- un espace mémoire propre à chaque groupe de processeurs et donc constitué de circuit à temps d'accès inférieur au demi-cycle machine

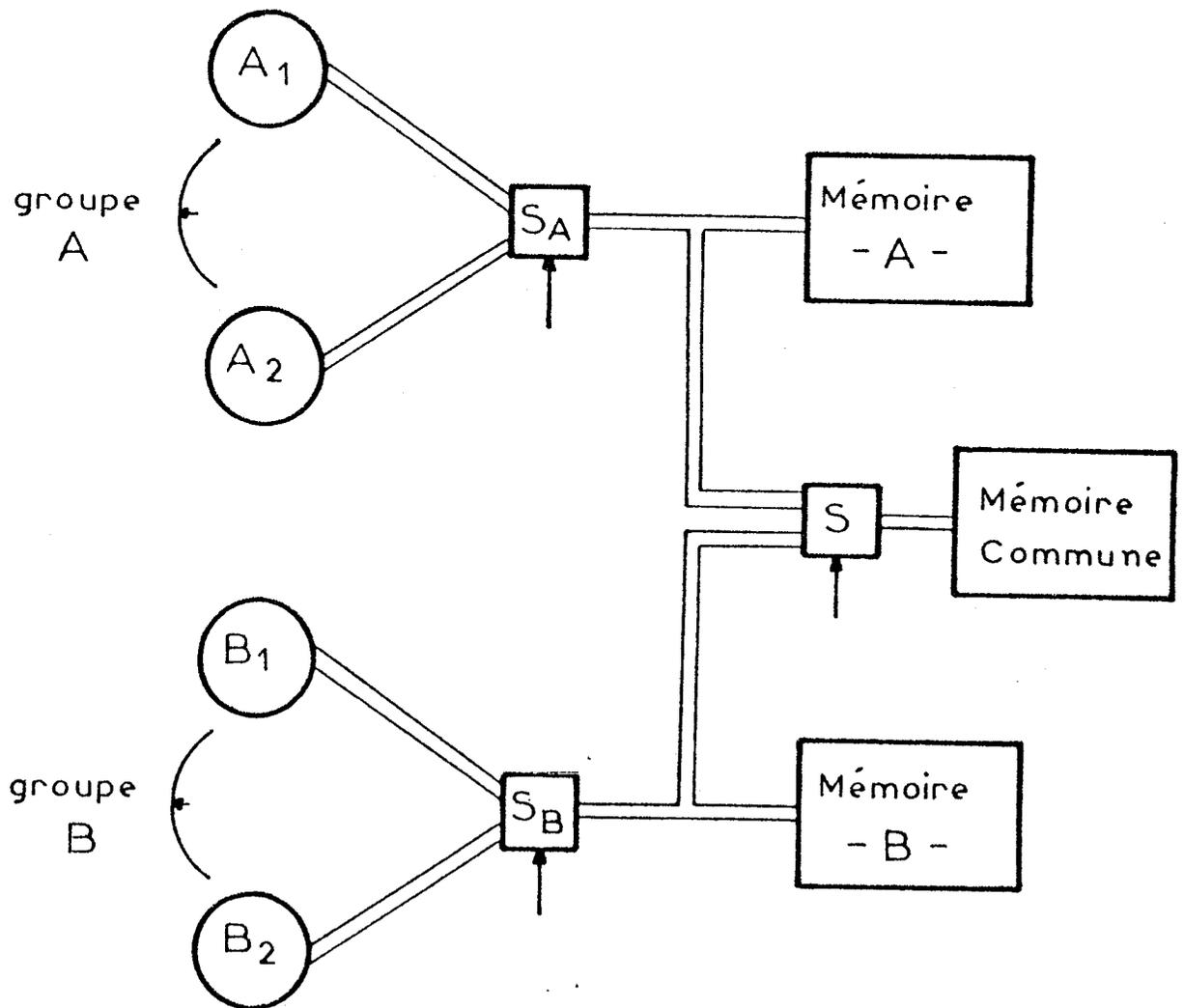


Figure III.1.8.

Cette structure justifiée par la méthode de multiplexage choisie, semble convenir à l'implantation d'un système multi-usage :

- la mémoire commune pourra alors contenir le noyau résident du système d'exploitation (en ROM par exemple) ainsi que des procédures partagées;
- la zone propre à un groupe restant dévolue aux données propres aux activités exécutées sur ce groupe ou aux zones de dialogue avec les organes d'entrée/sortie.

III - 1.2.5. Réalisation du multiplexage temporel : Génération des signaux d'horloge

Sur le chronogramme de la figure III.1.7 on voit facilement que le signal $\phi 2 (A_1)$ (resp. $\phi 2 (B_1)$) se traduit de $\phi 2 (A_2)$ (resp. $\phi 2 (B_2)$) par simple complémentation.

Pour générer les différents signaux horloges il suffit donc de générer deux signaux ϕA , ϕB de période égale au cycle machine, décalés de un quart de cycle machine.

Ces deux signaux sont obtenus à partir d'un signal unique $4 f_0$ de période égale à un quart de cycle machine (Fig. III.1.9).

A l'aide de ces quatre signaux il est facile de piloter les accès mémoire de nos quatre processeurs (Fig. III.1.10).

Du fait de l'utilisation de ces signaux dans le partage mémoire, le chronogramme de la figure III.1.9 doit être absolument suivi. Afin d'éviter toute dérive dans le temps des décalages relatifs des signaux d'horloge, nous employons un circuit de génération d'horloge bouclé. On peut voir sur la figure III.1.11 le schéma synoptique d'un tel générateur. Le signal ϕB est "auto-corrigé" par la boucle de retour.

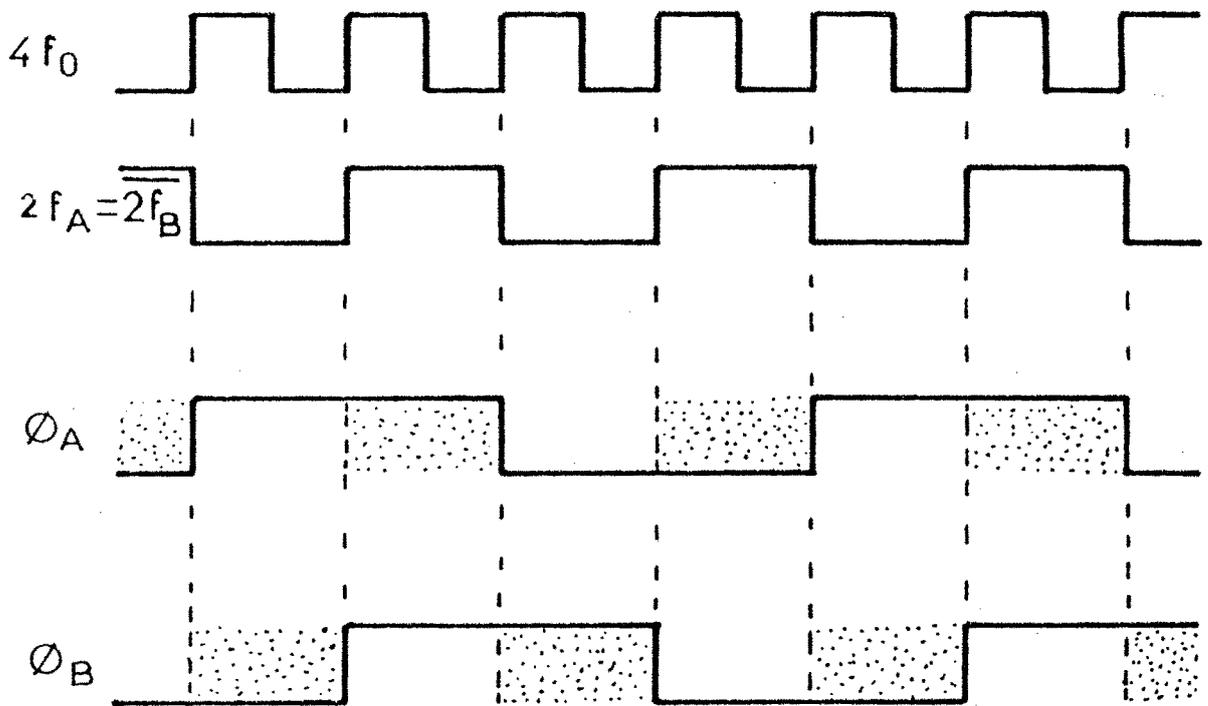


Figure III.1.9.

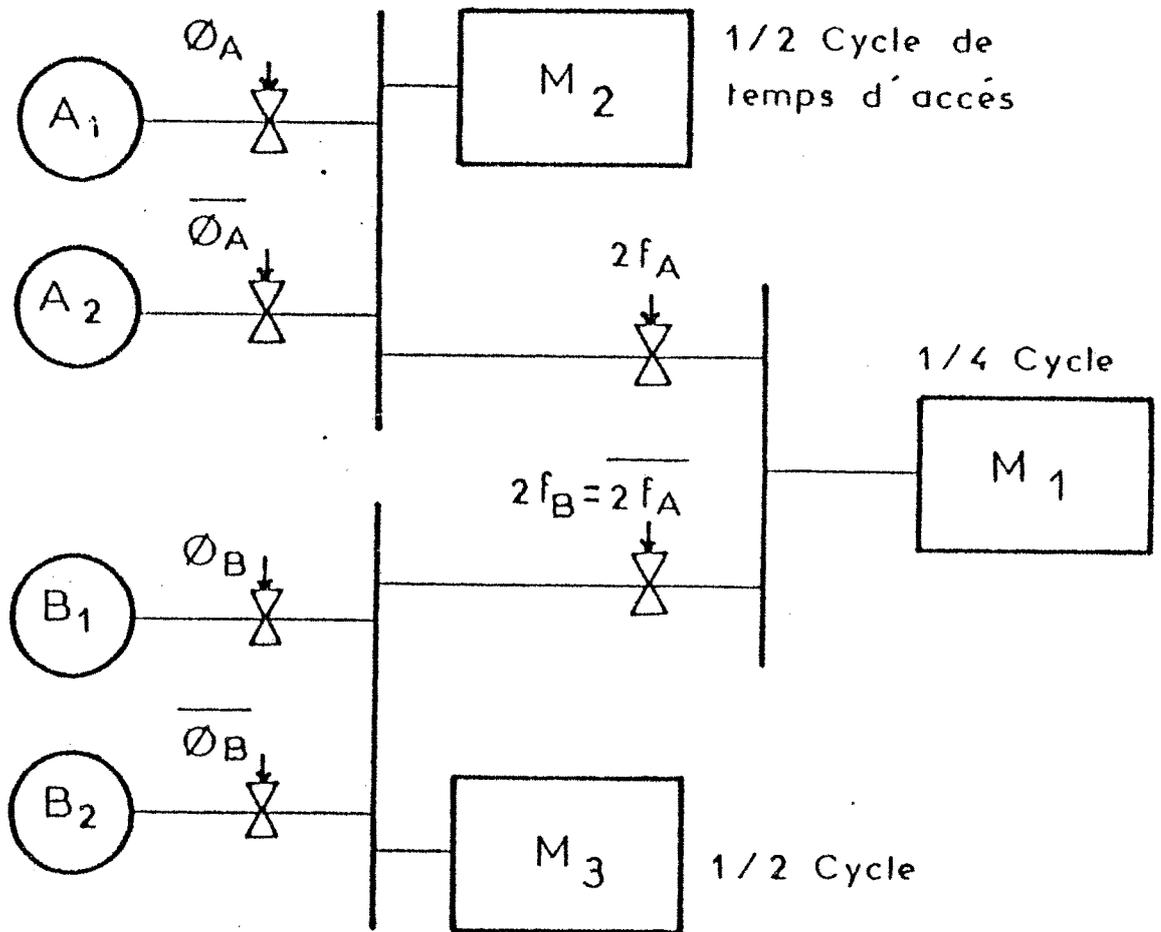


Figure III.1.10.

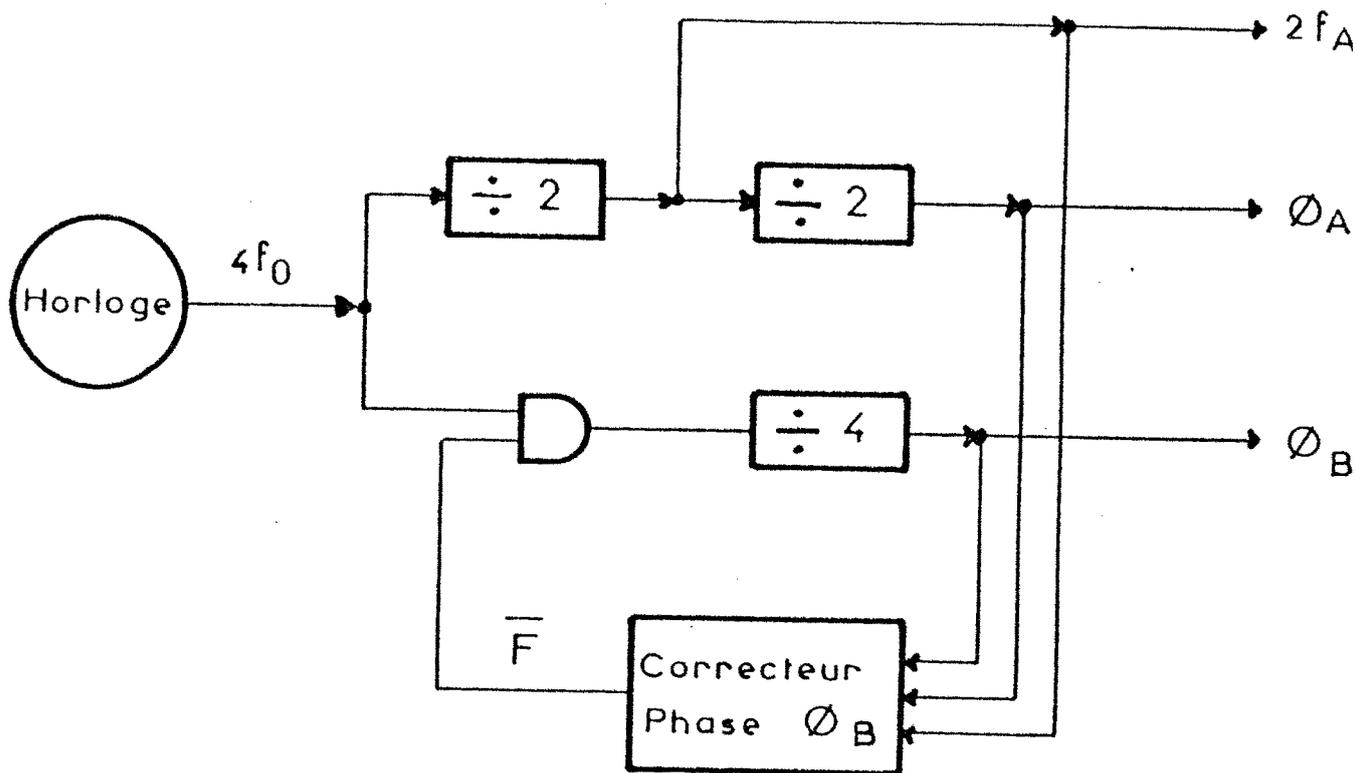


Figure III.1.11.

Une telle conception conduit également à une bonne immunité aux bruits.

La fonction F utilisée en retour est égale à

$$F = \overline{2f_A} \phi_A \phi_B + 2f_A \phi_A \overline{\phi_B} + \overline{2f_A} \overline{\phi_A} \overline{\phi_B} + (2f_A) \overline{\phi_A} \phi_B.$$

Cette fonction reste nulle tant que les relations entre les signaux restent conformes au diagramme de la figure III.1.9.

En cas de retard ou d'avance de ϕ_B un "top" de l'horloge $4f_0$ est enlevé à l'horloge ϕ_B .

III - 2. PROTECTION MEMOIRE ET ALLOCATION

Pour des raisons de sûreté de fonctionnement la zone mémoire commune aux quatre processeurs (32K) a été découpée en deux bancs mémoire de 16 K d'accès indépendants. (M_{11} , M_{12}).

De même, les zones propres à chaque couple se voient partagées en deux bancs de 16 K (M_{21} , M_{22}) et (M_{31} , M_{32}). Ces zones, à temps d'accès long doivent être en partie réservées aux interfaces d'entrée/sortie qui ne peuvent pas, à cause de leur lenteur, s'intégrer dans la zone commune aux quatre microprocesseurs. Elles doivent également contenir des zones de mémoire (morte ou vive). La solution pour l'adressage des mémoires mortes (III.2.1) conduit à interdire l'écriture dans tout bloc comportant des mémoires mortes. Nous avons donc, pour cette zone propre à un couple, deux types d'emplacements :

- des emplacements à lecture seule (ROM)
- des emplacements à lecture/écriture (RAM, E/S)

Nous avons donc adopté la solution du partage de cette zone en deux bancs égaux; chacun de ces deux bancs pouvant être spécialisé dans l'un ou l'autre de ces deux types d'emplacements (Fig. III.2.1).

Une fois ce découpage adopté, plusieurs problèmes sont apparus. Ils ont été progressivement résolus par des modifications et des ajouts successifs à un même mécanisme. Ce mécanisme peut être considéré comme un système élémentaire de protection mémoire.

III - 2.1. Protection de l'écriture en ROM et en RAM

Les zones mémoires de "4M" devaient pouvoir être spécialisées en fonction de l'application visée. On ne pouvait donc utiliser la solution "classique" d'adressage de ROM consistant à utiliser la ligne R/W comme signal de sélection du boîtier.

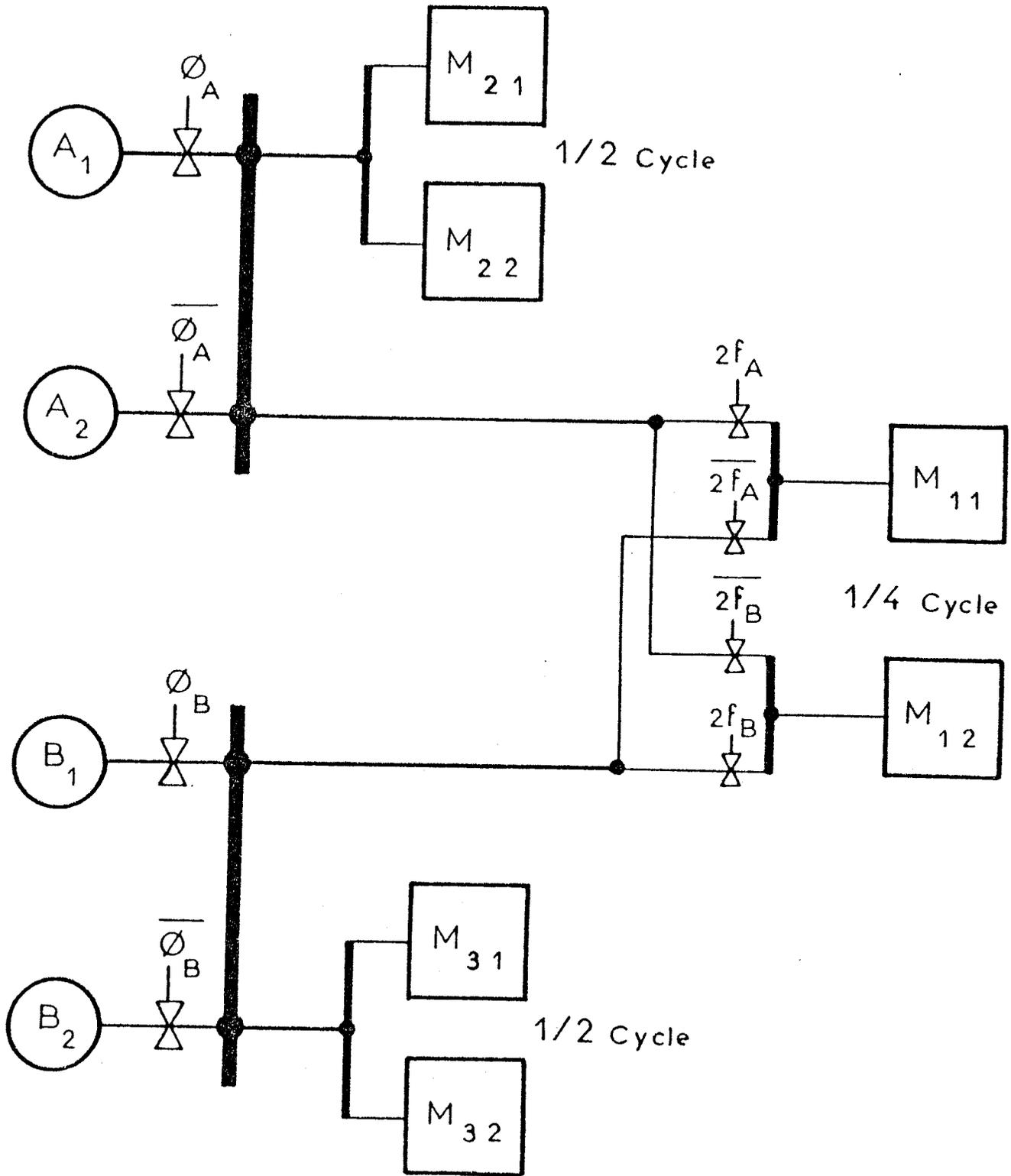


Figure III.2.1.

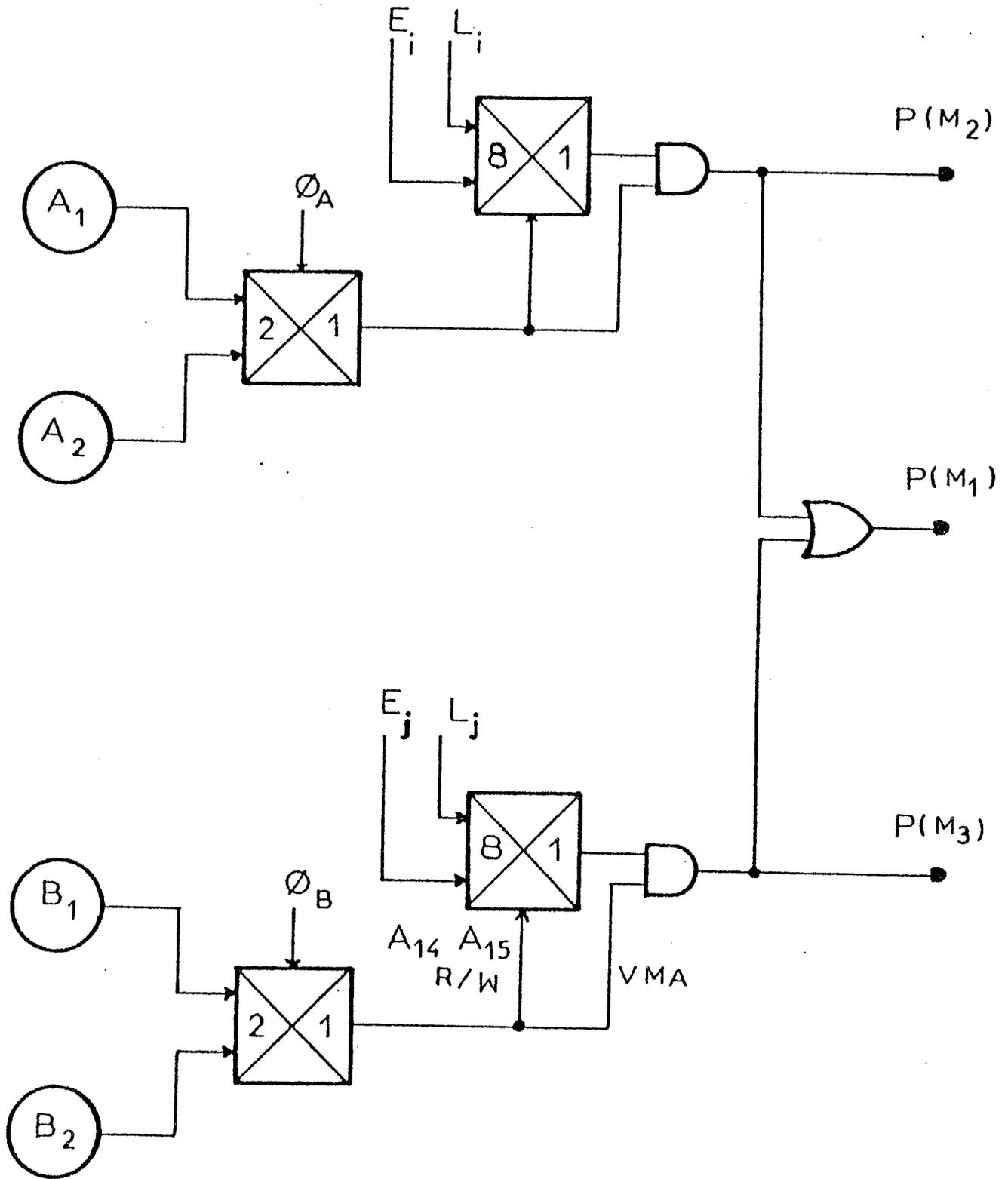


Figure III.2.2.

Il faut donc générer un signal de sélection correspondant au banc accédé. Ce signal de sélection doit pouvoir être inhibé pour les seuls "accès" "écriture". La réalisation choisie qui utilise un multiplexeur 8-1 fournit une protection indépendante pour les quatre bancs accessibles à une paire de processeurs. Ces bancs, possèdent chacun deux valeurs binaires E_i et L_i interdisant ou autorisant, respectivement, l'écriture ou la lecture dans le banc (Fig.III.2.2). On remarque sur cette figure que l'on peut avoir des protections distinctes pour chaque paire de processeurs.

De plus, le signal de sélection $P(M_1) P(M_2) P(M_3)$ peut être calibré et synchronisé par l'intermédiaire des signaux $E_i L_i$.

Cette remarque montre que l'on peut associer droit et fenêtre temporelle d'accès; elle nous a permis de contrôler la calibration des accès mémoire par l'intermédiaire de ce même mécanisme. Notamment, à la désélection des RAM en cycle écriture, il nous a été nécessaire de compenser le retard causé du temps de propagation de l'information dans les multiplexeurs.

Cette maîtrise des moments d'échantillonnage permet donc, comme le décrit le chronogramme (Fig. III.2.3) que l'information valide provenant du microprocesseur enveloppe celle de la mémoire M pour un cycle écriture ECR et vice-versa pour un cycle lecture LEC, améliorant ainsi la sûreté du fonctionnement.

Ce mécanisme de calibration sera employé dans la réalisation finale proposée au paragraphe suivant.

III - 2.2. Protection fine et allocation des pages

Cette protection plus fine est rendue nécessaire par la zone commune aux quatre processeurs, notamment pour des questions d'isolation entre programme parallèle.

Cette zone a donc été découpée, à son tour, en 16 "pages" de 2k octets (8 pages par bancs).

Cette protection est assurée par l'adjonction d'un multiplexeur supplémentaire à chaque processeur (Figure III.2.4).

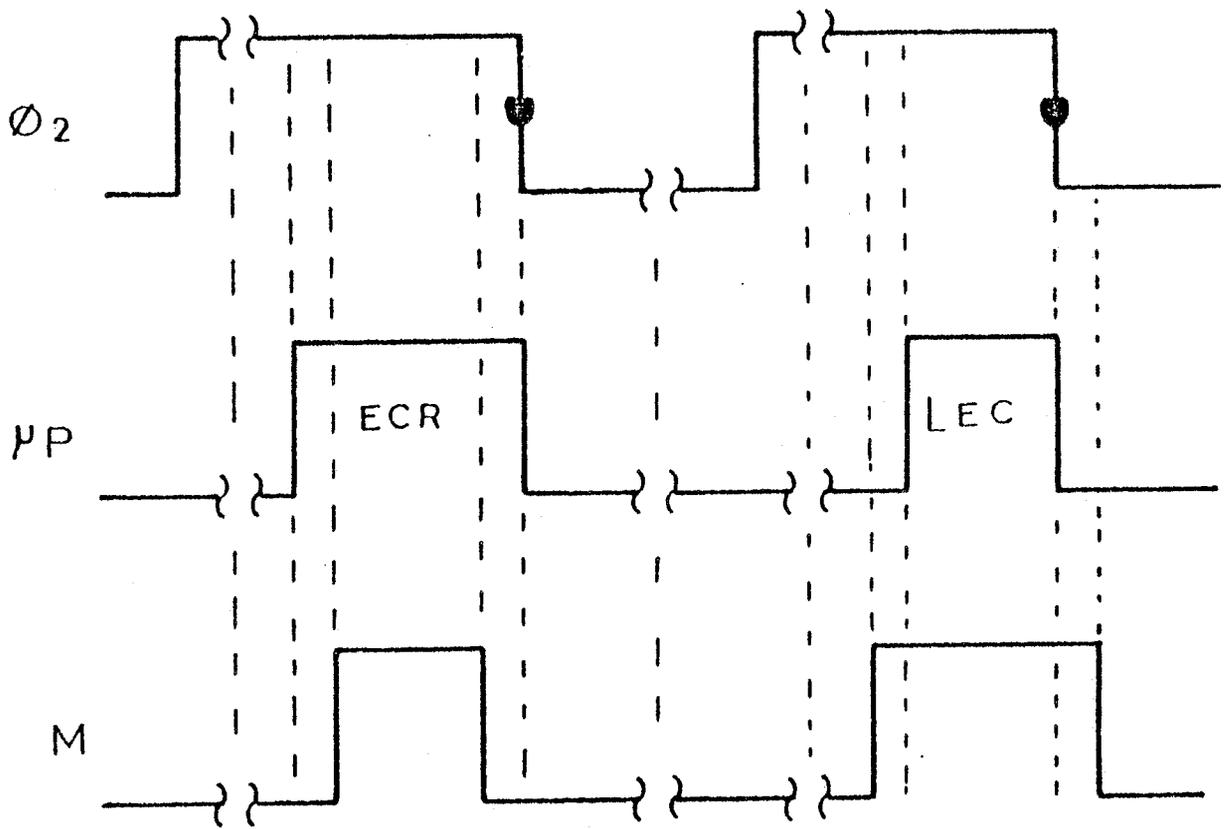


Figure III.2.3.

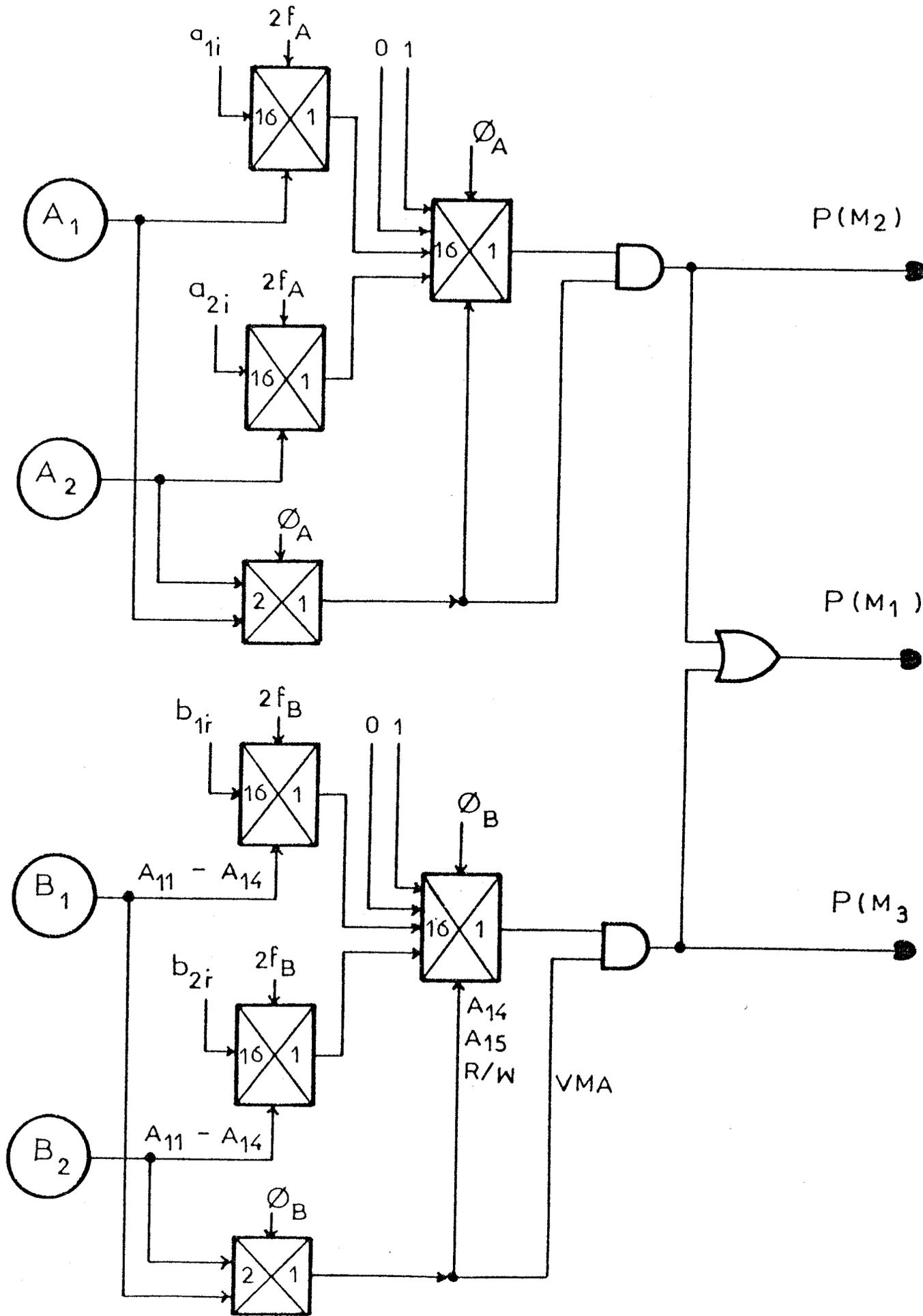


Figure III.2.4.

On remarque sur cette figure le rôle des multiplexeurs dans l'implantation du multiplexage temporel (signaux $2f_A$ et ϕ_A).

Sur cette figure on voit que cette méthode est facilement généralisable à l'espace mémoire tout entier.

Les variables logiques (a_{1i} , a_{2i} , b_{1i} , b_{2i}) permettent une allocation des pages de la mémoire commune, à chacun des processeurs (A_1 , A_2 , B_1 , B_2). Cette allocation des pages peut être statique (câblée une fois pour toute), ou dynamique (programmable au moyen d'un registre adressable).

Les signaux $P(M_1)$ et $P(M_3)$ peuvent être combinés avec les signaux VMA et les phases des processeurs pour générer, en cas d'un accès mémoire illicite, une interruption au processeur correspondant.

III - 3. ARCHITECTURE REALISEE

Cette architecture est présentée sur les figures III.3.1 et III.3.2.

III - 3.1. Liaisons processeurs-mémoire

Ces liaisons sont indiqués par la figure III.3.1. La solution retenue permet à chaque couple d'avoir une liaison avec les deux bancs mémoire communs, indépendante de celle de l'autre couple. Cette solution présente deux avantages :

- sur le plan de la sûreté de fonctionnement comme nous le verrons au chapitre V.
- sur le plan technologique en utilisant des bus de charge capacitive et d'impédance d'entrée raisonnable.

III - 3.2. Implantation et découpage

La machine réalisée a été découpée en sept cartes distinctes :

- une carte unité centrale comportant quatre processeurs
- six cartes banc mémoire.

Ce découpage est indiqué en traits pointillés sur la figure III.3.2.

Ce découpage possède deux caractéristiques intéressantes :

- un interface peu volumineux entre cartes :

au maximum : bus donné (8)

bus adresse (14) + R/W

quatre fils de sélection de banc

- modularité et interchangeabilité rapide des cartes mémoire : "4M" peut fonctionner malgré l'absence d'un ou de plusieurs bancs. Les seuls paramètres fixés pour un banc donné sont :
 - . le temps d'accès maximal
 - . la taille maximale.

III - 3.3. Sélection et organisation du banc

Une portion des bus donnée (BD) n'est pas présente sur la figure III.3.1. En contrepartie cette figure détaille le mécanisme de sélection de banc sur la carte processeur, ainsi que l'organisation interne de chaque banc. Ces derniers sont en effet découpés en 16 pages de 2K octets chacune.

III - 3.4. Système d'interruption

Un système classique d'interruption, proposé sur la carte processeur, est présenté (Fig.III.3.3) pour un groupe de processeurs (X_1 , X_2).

Il consiste à former deux vecteurs d'interruption : (FFC8/FFDF) pour X_2 et (FFE8/FFFF) pour X_1 .

Mises à part les interruptions :

$\overline{\text{RES}}$: qui initialise le système à la mise sous tension,

$\overline{\text{NMI}}$: non masquable,

SWI : programmée.

Il y a par processeur huit niveaux de priorité d'interruption masquable dont chacun correspond à une adresse de départ du programme, qui le traite. Un ou plusieurs de ces niveaux peuvent être inhibés par logiciel.

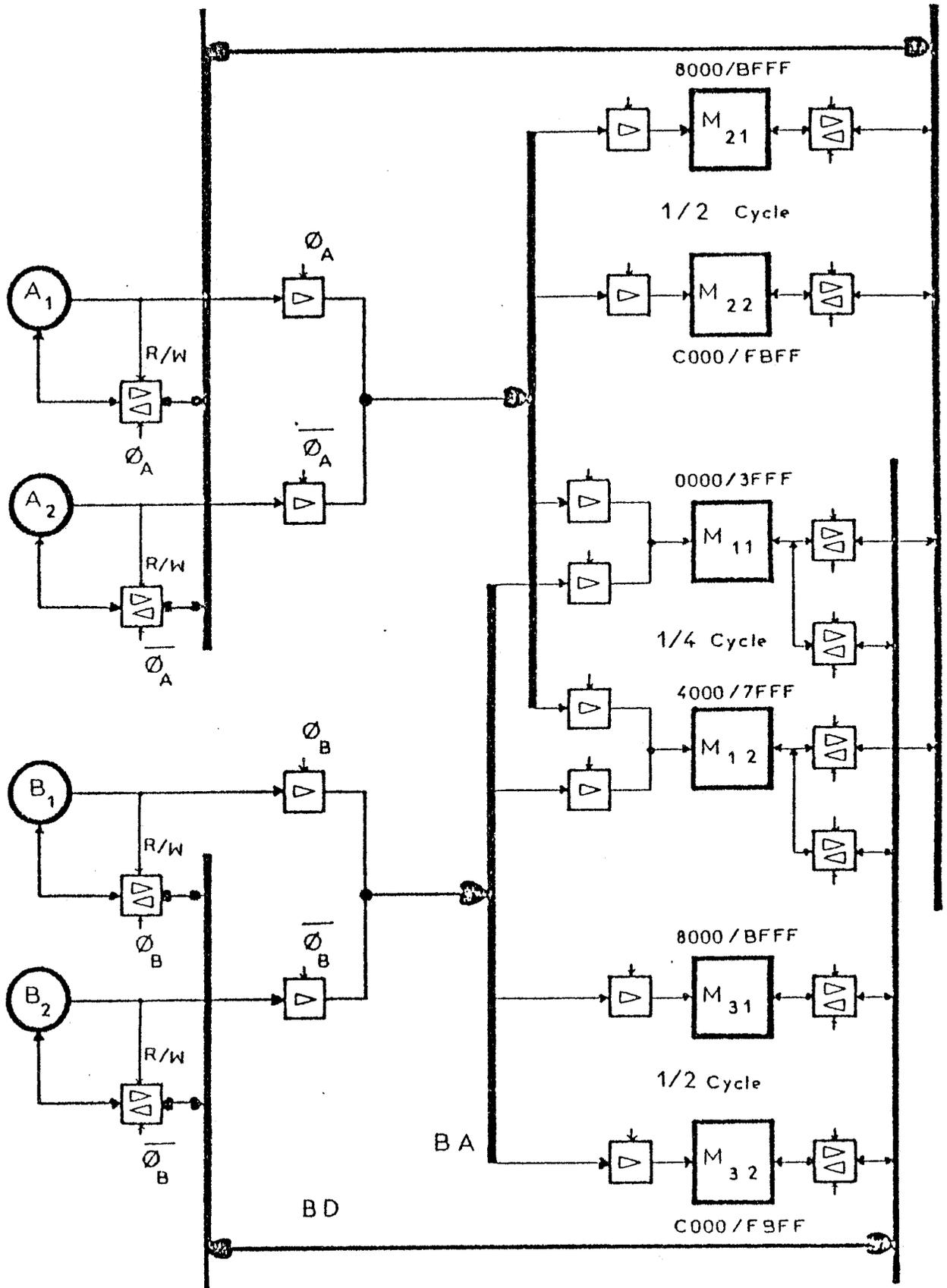


Figure III.3.1. : Liaisons processeurs - mémoires

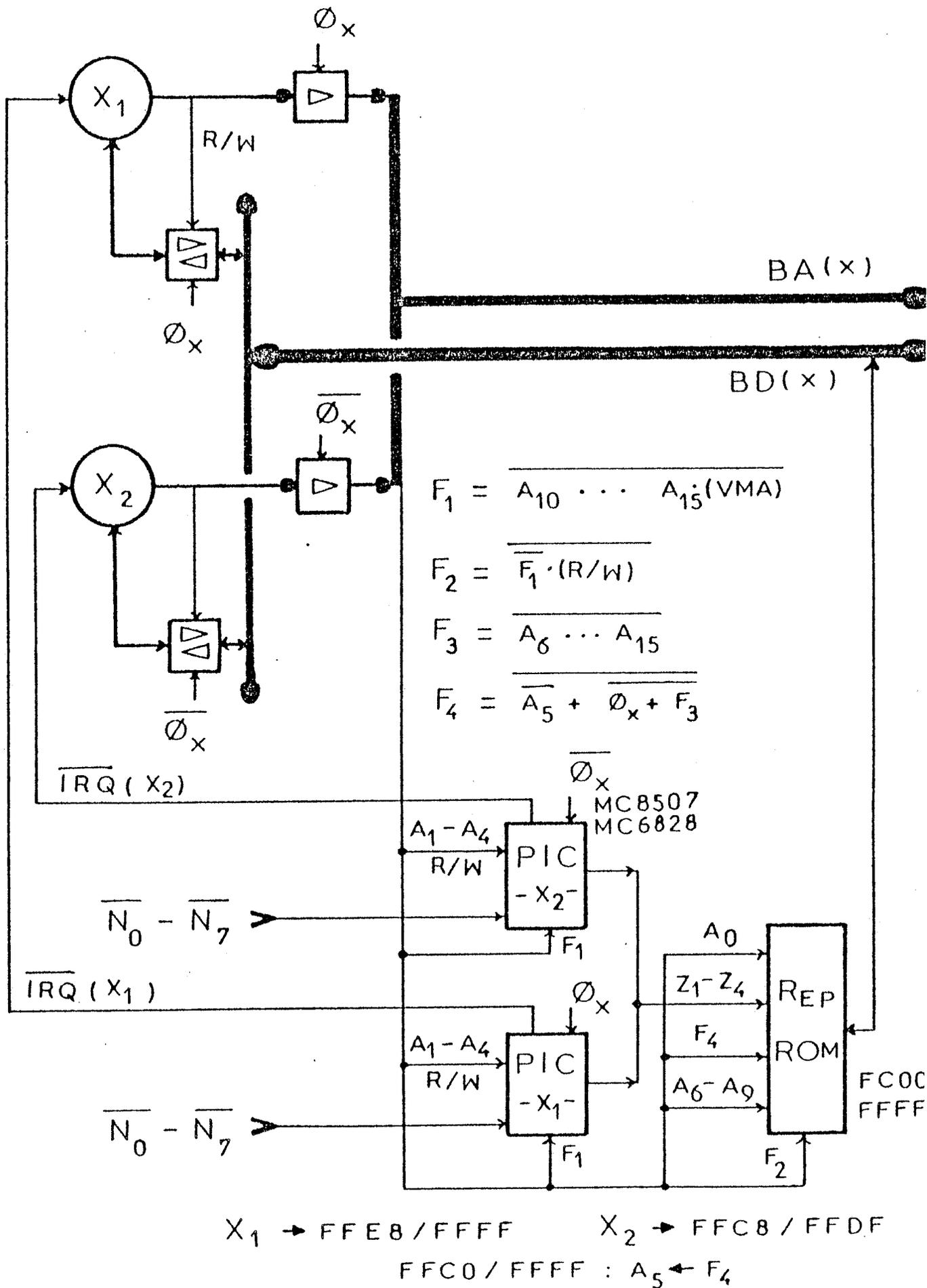


Figure III.3.3. : Système d'interruption vectorisé

CHAPITRE IV
MISE EN OEUVRE DE LOGICIEL
SUR LE SYSTEME "4M"

-0-0-0-

IV - MISE EN OEUVRE DE LOGICIEL SUR LE SYSTEME "4M"

IV - 1. INTERPRETEUR DE RESEAUX DE PETRI

IV - 1.1. Généralité

Ce système peut supporter n'importe quel logiciel, en particulier temps réel où une puissance de parallélisme (degré 3 ou 4) est souhaitée.

Nous avons choisi d'illustrer l'approche "MAS" étudiée en parallèle par l'équipe "conception et sécurité". Il s'agit de décrire les systèmes temps réel sous forme de Réseaux de Petri interprétés. L'étude d'une telle description et la faisabilité sur de nombreux cas pratiques ont été largement étudiés [19].

Les systèmes temps réels ainsi décrits, peuvent être validés par analyse ou simulation; on souhaite que la production du logiciel final respecte les résultats obtenus au cours de ces validations et n'introduisent pas de nouvelles erreurs de conception.

Cette exigence de production sûre du logiciel d'application a amené à l'écriture d'un interpréteur de réseaux de Petri généralisés. Le concepteur a donc à sa disposition une chaîne C.A.O complète pour ses applications temps réels; la description initiale validée éventuellement par simulateur est aussi acceptée comme entrée de l'interpréteur.

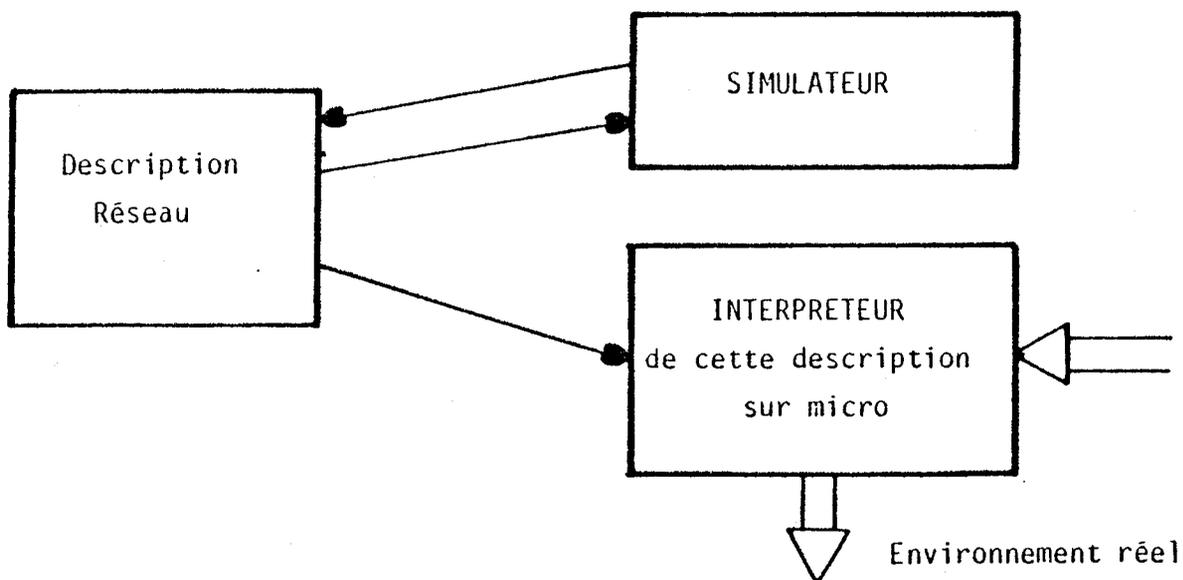


Figure IV.1.

Nous donnerons en IV.1.2.3 les renseignements sur l'implémentation de l'interpréteur sur le système 4M.

En IV.2, nous donnerons un exemple pratique (centrale à béton) modélisé selon les normes de l'interpréteur.

L'interpréteur n'étant pas un produit logiciel achevé, nous donnerons une programmation directe de cet exemple sur 4M dans une annexe.

IV - 1.2. Implémentation d'un interpréteur de Réseau de Petri synchronisé sur 4M

Une application étant décrite par

- . un ensemble de tâches
- . un Réseau de Petri décrivant l'enchaînement des tâches (1,2,3).

Nous proposons une implémentation de cette application de la façon suivante :

- un processeur assurera le calcul de l'évolution du Réseau de Petri et commandera les tâches à exécuter (Partie Contrôle),
- deux processeurs assureront l'exécution des tâches (Partie Opérative)
- pour des raisons de fiabilité 1 processeur sera mis en stand by pour la partie contrôle,
- 4M paraît particulièrement adapté pour cette utilisation et l'interpréteur sera donc implémenté de la façon suivante sur 4M :

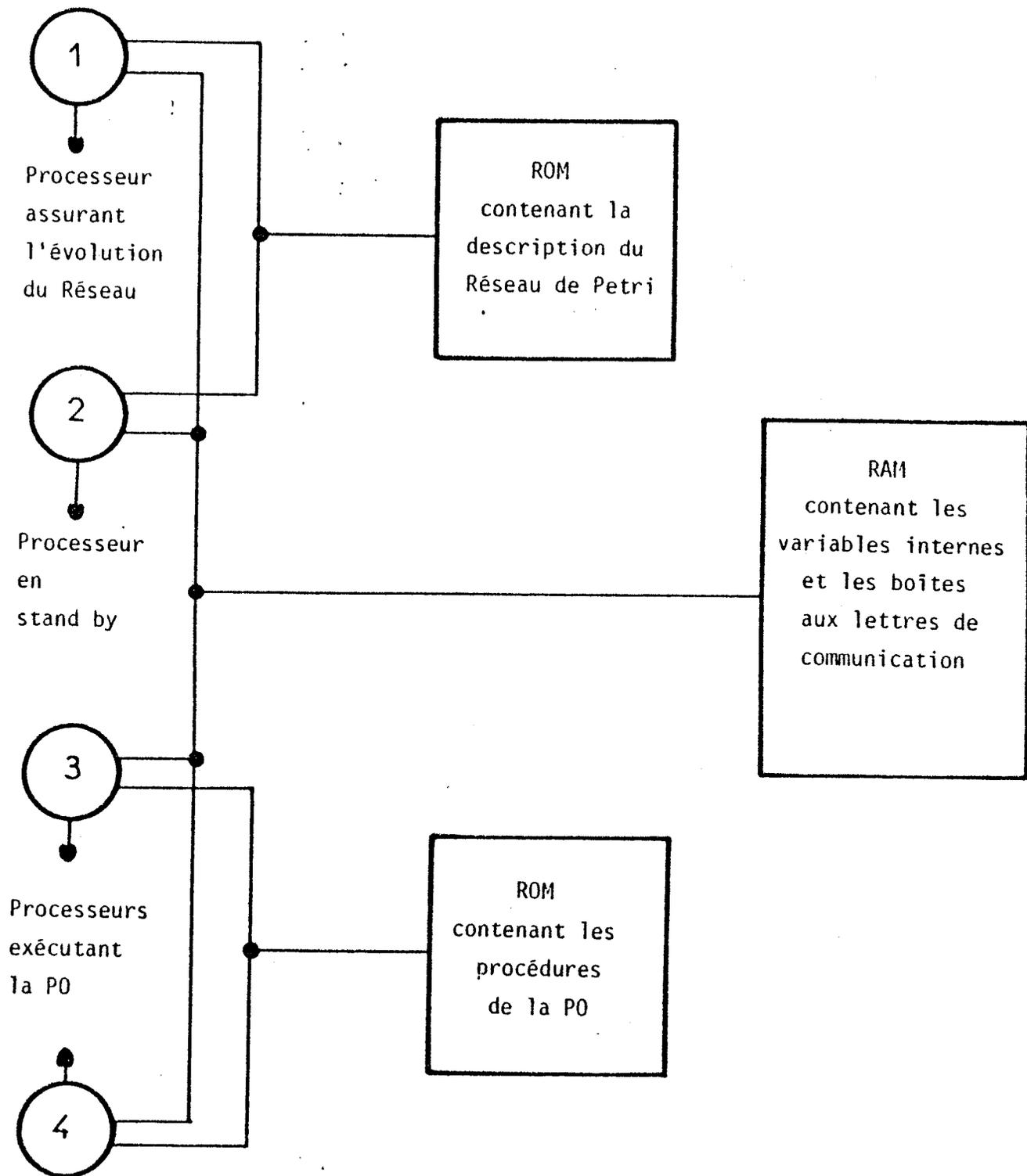


Figure IV.2.

IV - 1.3. Description du logiciel

Le logiciel de l'interpréteur est composé de 4 processus communicant entre eux au moyen de boîte aux lettres. (Fig.IV.3).

Le processus traitement interruption reconnaît les entrées et les met dans la boîte aux lettres événement, il reçoit aussi les signaux processeurs libres, les reconnaît et les met dans la boîte aux lettres processeurs libre. Ce processus est exécuté par le processeur 1.

Le processus évolution du réseau de Petri est directement dérivé du simulateur MAS (4). A partir de la liste des événements il calcule l'évolution du réseau de Petri et donne les tâches à faire. Ce processus est exécuté par le processeur 1.

Le processus allocateur de tâche distribue les tâches aux processeurs de la partie opérative dans une boîte aux lettres (une par processeur). Ce processus peut être plus ou moins complexe (prise en compte de priorité entre tâches, calcul de ces priorités). Pour la première version de l'interpréteur, nous avons adopté une version très simple. Le processus distribue les tâches à faire dans l'ordre d'arrivée de ces tâches. Ce processus est exécuté par le processeur 1.

Le processus exécution de la tâche lit l'adresse de la tâche dans sa boîte aux lettres privée, l'exécute, envoie un signal fin de tâche. Ce processus est exécuté par les processeurs 3 et 4.

Ecriture du logiciel . Seul le processus traitement d'interruption n'existe pas au niveau du simulateur MAS. Le programme fait environ 100 instructions (introduction d'erreur minime). Le reste du logiciel est entièrement dérivé du simulateur MAS. Cela évite l'introduction de nouvelles erreurs de conception.

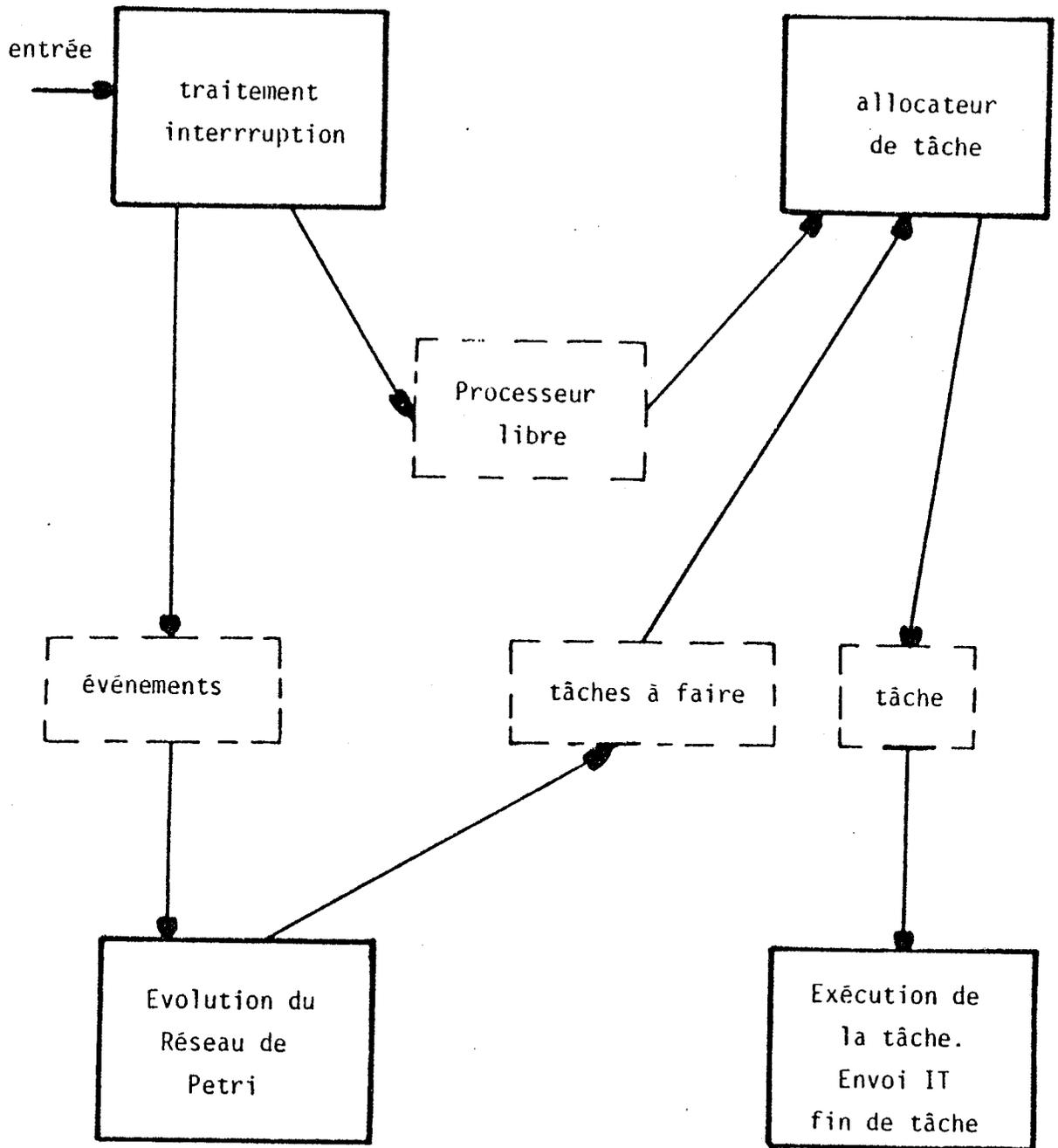


Figure IV.3.

IV - 2. L'EXEMPLE DE L'AUTOMATISATION D'UNE CENTRALE A BETON

IV - 2.1. Description d'une centrale à béton [20]

La fabrication du béton consiste à mélanger des quantités déterminées de différents produits suivant la qualité du béton à obtenir.

A savoir :

- des agrégats : sables, graviers et cailloux de calibres différents
- des ciments
- de l'eau

Les quantités des différents produits solides (agrégats et ciments) sont déterminées par leur poids, la quantité d'eau l'étant par son volume.

La figure IV.4 donne un exemple de réalisation d'une centrale à beton.

Les agrégats stockés dans des compartiments (1 à 6) peuvent s'écouler par des trappes actionnées par des vérins hydrauliques ou pneumatiques. Ces agrégats sont pesés dans une balance (7) mécanique dans les anciennes réalisations et actuellement à jauge de contrainte. La vidange de la balance est réalisée par une trappe. Il faut remarquer que les agrégats ne sont pas pesés séparément mais que leur poids s'ajoutent dans la balance.

Les ciments sont pesés dans une balance semblable à la précédente (10). Ils proviennent de silos (8 et 9), entraînés par des vis sans fin. L'eau mesurée par un compteur volumétrique s'écoule par une vanne d'une réserve permettant un débit important (11). Les produits sont mélangés dans un malaxeur (12) entraîné par un moteur électrique ou hydraulique. En fin de malaxage le béton est disponible par une trappe (13) ou par basculement du malaxeur.

Considération sur l'utilisation et l'automatisation d'une centrale à béton

Nous remarquons tout d'abord la présence de deux balances distinctes pour les agrégats et les ciments. Ceci permet de garantir la quantité de ciment qui peut être petite devant celle des agrégats sans exiger des balances de haute précision. En plus, l'écoulement du ciment étant lent, les deux pesées peuvent être effectuées simultanément.

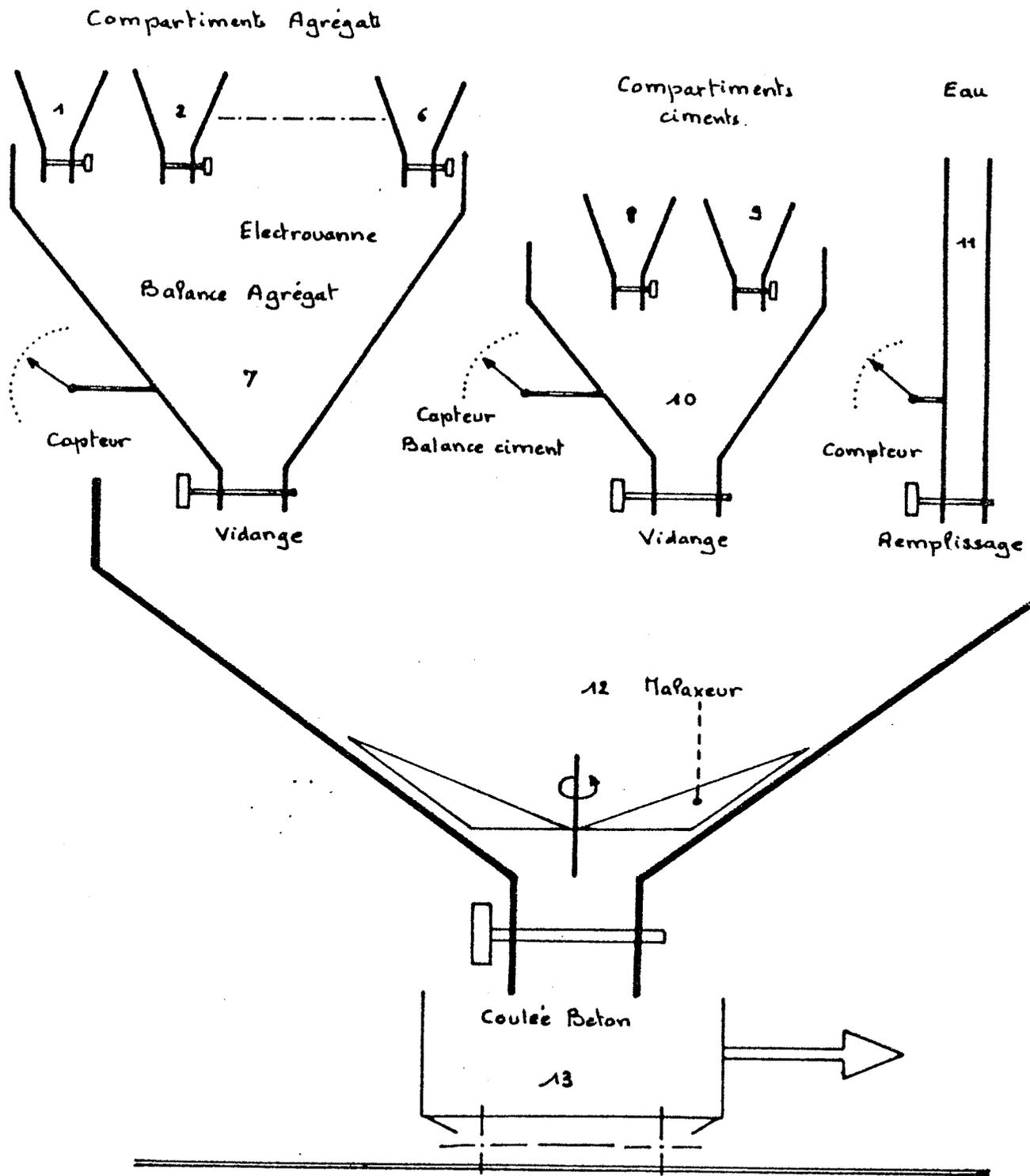


Figure IV.4.

D'autre part, suivant l'utilisation et le mode de transport du béton, la vidange du malaxeur doit pouvoir s'effectuer de façons différentes :

- soit une vidange intermittente commandée manuellement dans le cas d'un transporteur de quantité inférieure à celui du malaxeur,
- soit dans le cas de transporteurs de quantité suffisante, une vidange complète.

Dans ce deuxième cas, il arrive que le transporteur puisse accepter la quantité de plusieurs malaxeurs (exemple : malaxeur : 1 tonne, transporteur : 6 tonnes). Dans le but d'un gain de temps, il est alors possible d'effectuer simultanément des opérations relatives à des cycles différents de fabrication. L'automatisation d'une centrale à béton devient une nécessité.

IV - 2.2. Modélisation sous forme de Réseau de Petri

IV - 2.2.1. Découpage en chaînes d'Asservissement

Sur la figure IV.5 nous avons décomposé l'automatisme en différents éléments qui vont correspondre à la suite de notre étude.

Nous distinguons les chaînes de balance agrégats et ciments. Seule la balance agrégats a été détaillée. Les chaînes sont reliées au "contrôle du processus" par des variables de communication. La chaîne de malaxage et de coulée de béton n'utilise que des variables d'entrées.

IV - 2.2.2. Définition des différentes tâches

On distingue dans le processus quatre tâches pouvant être exécutées en parallèles. Ce sont :

- séquence de pesées et vidange de la balance agrégats
- séquence de pesées et vidange de la balance ciments
- entrées-sortie
- remplissage d'eau, malaxage, coulée et contrôle global de l'ensemble.

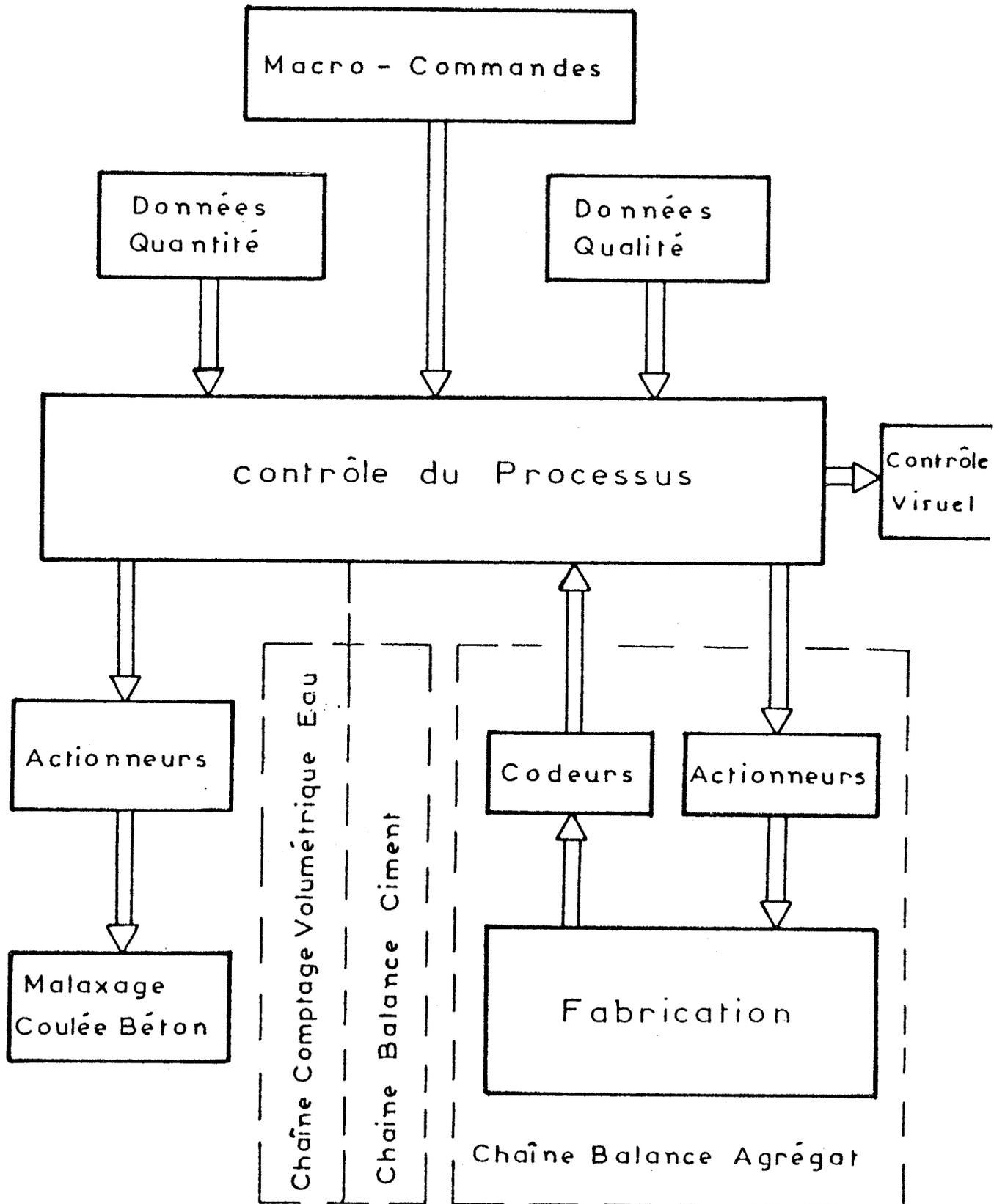


Figure IV.5.

IV - 2.2.3. Modélisation des tâches

Le réseau de Petri général (Fig.IV.6) décrit le déroulement global de ces quatre tâches parallèles dont les processus de contrôle sont appelés :

- séquenceur balance agrégats,
- séquenceur balance ciment,
- séquenceur central,
- entrée/sortie

On attache à chacun des séquenceurs un "compteur d'état". Chacun des états de ce compteur correspond à l'exécution d'une phase de la tâche du séquenceur. Un bit indique si la phase en cours est active ou en attente active.

Ces processus asynchrones ne sont pas indépendants. En effet, le contrôle global du fonctionnement de la centrale de commande, doit vérifier que pour chaque cycle de fabrication ait lieu un, et un seul, cycle de chacune des fonctions suivantes en s'assurant de leur bon déroulement.

- un cycle de pesée pour chacune des deux balances
- une vidange par balance
- un remplissage d'eau
- un malaxage
- une coulée béton

Il ordonne les cycles pesées jusqu'à l'obtention des N fabrications demandées.

En revanche, la coopération de ces processus se limite à échanger très peu de mots de synchronisation. En effet, les séquenceurs balances indiquent simplement les fins de vidange (b_A , b_C) au séquenceur central qui autorise, dans sa boucle de fermeture, d'autres vidanges et indique à son tour le numéro de cycle de fabrication en cours n . Ces processus sont détaillés dans la suite.

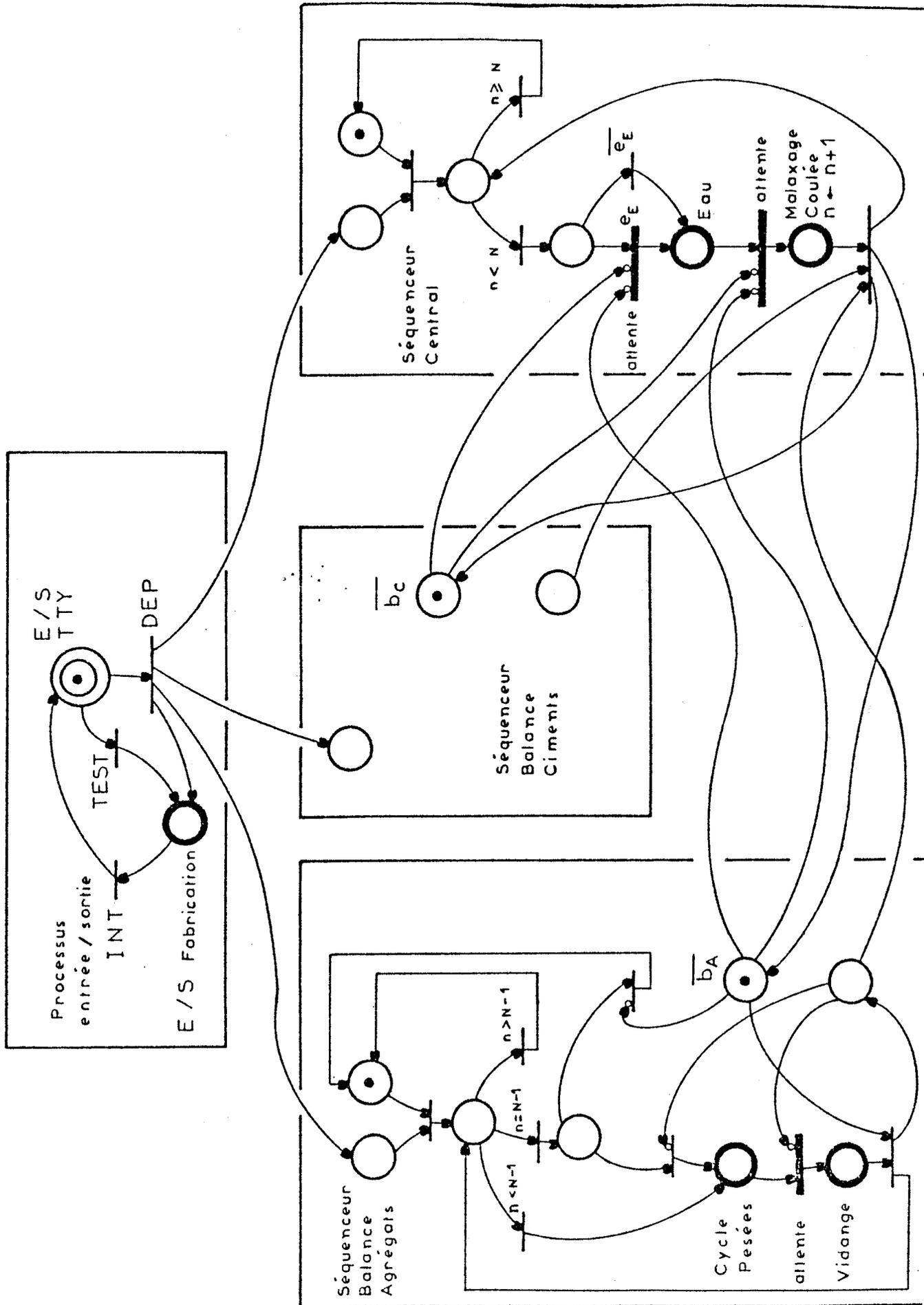


Figure IV.6.

IV - 2.3. Asservissement d'une chaîne balance

Nous rappelons que les balances agrégats et ciment sont identiques, elles ne diffèrent que par leur capacité.

IV - 2.3.1. Principe d'une pesée

Le séquenceur balance doit effectuer nécessairement la pesée des différents produits et la vidange dans le malaxeur.

La figure IV.7 représente le poids indiqué par la balance P_X lors d'un cycle. Ce poids est comparé à une consigne représentative du poids à atteindre.

Pendant le chargement, lorsque la valeur de consigne est atteinte, le séquenceur doit augmenter la consigne d'une valeur égale au poids du produit suivant P_{X_α} et ouvrir la trappe correspondante.

Ainsi, durant la phase de pesée du produit X_α , la consigne est égale à :

$$R.B. = \sum_{\beta=1}^{\alpha} e_{X_\beta} \cdot P_{X_\beta}$$

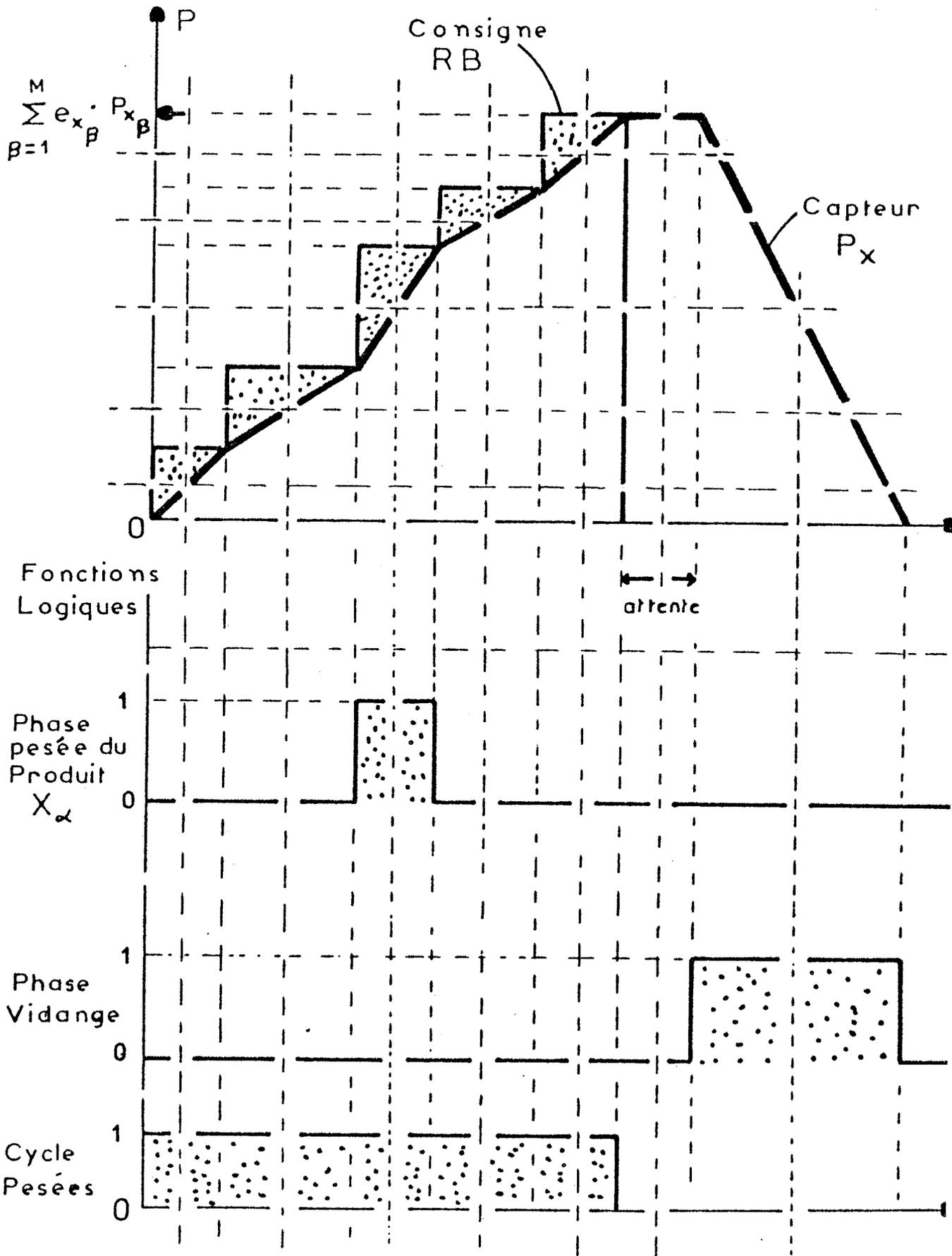
où e_{X_α} représente l'utilisation (si $e_{X_\alpha} = 1$) ou non (si $e_{X_\alpha} = 0$) d'un produit.

La vidange de la balance s'effectue, si elle y est autorisée, après le chargement des M produits. La fin de vidange est assurée lors du passage à zéro de la balance.

IV - 2.3.2. Principe de séquenceur balance (Fig.IV.8)

Les phases α étant successives, on attache chacune d'elles à un état β d'un compteur appelé "compteur d'état" dont les états $\beta = 1$ à M correspondent aux phases pesées $\alpha = \beta$ des produits X_α ; l'état $\beta = 0$ correspond à l'état d'initialisation, l'état $\beta = M+1$ indique la fin d'un cycle de pesées et correspond à la durée de l'attente et la phase de vidange synchronisée par b_X .

Les états β du compteur d'état, correspondants aux produits éliminés X_α par la combinaison d'entrées qualité $(e_{X_1}, \dots, e_{X_M})$, sont sautés. Il en est de même des états non utilisés par le séquenceur.



Principe du Cycle Balance

Figure IV.7.

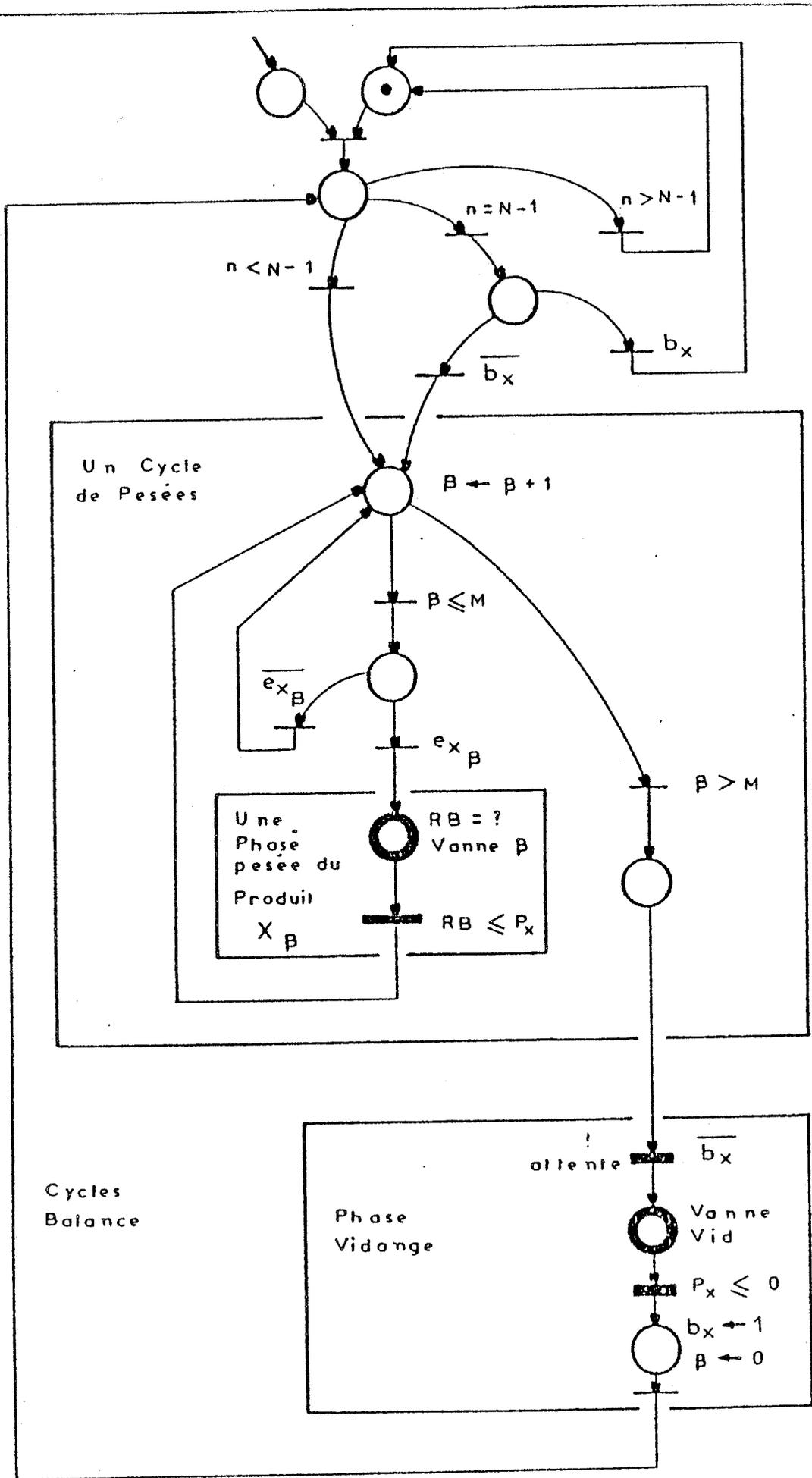


Figure IV.8. Séquenceur Balance

IV -2.4. Principe du séquenceur central (Fig.IV.9)

Les fins de vidange des balances agrégats b_A et ciments b_C sont mémorisées pour interdire de nouvelles vidanges jusqu'à la fin de la coulée de béton du cycle de fabrication précédent.

Cependant, un nouveau cycle de pesées d'une balance peut se réaliser dès la fin de vidange b_X de cette balance, si la comparaison n/N l'autorise.

Les phases remplissage eau, malaxage et coulée étant successives, on attache chacune d'elles à un état γ d'un compteur appelé "compteur d'état" de façon similaire aux séquenceurs balances.

IV - 2.5. Entrée/sortie (Fig.IV.10)

La tâche du processus d'entrée/sortie se divise en deux tâches principales :

- communiquer avec le pupitre de commande (entrée/sortie TTY)
- entrée/sortie de l'asservissement.

a) Entrée/sortie TTY

Elle permet de communiquer avec les macro-commandes qui déclenchent des routines de service telles que :

- initialisation du système : chargement des données et sélection du mode du fonctionnement,
- démarrage, interruption, annulation ou reprise de la fabrication,
- test de l'électronique : mémoire, processeurs et bus (par programmes spécifiques de détection),
- test du logiciel (par simulation visuelle),
- test de l'installation électrique des actionneurs (en effectuant directement des commandes de test aux actionneurs).

b) Entrée/sortie de l'asservissement

Elle consiste à :

- lecture des signaux, en provenance des capteurs pour les communiquer aux séquenceurs correspondants,
- lecture de l'état des séquenceurs (exprimé par leurs compteurs d'état et les mots de synchronisation), pour former le vecteur de commande aux actionneurs,
- sauvegarder les données importantes (états des séquenceurs, etc) dans une mémoire non volatile pour permettre une reprise en cas de panne d'alimentation.

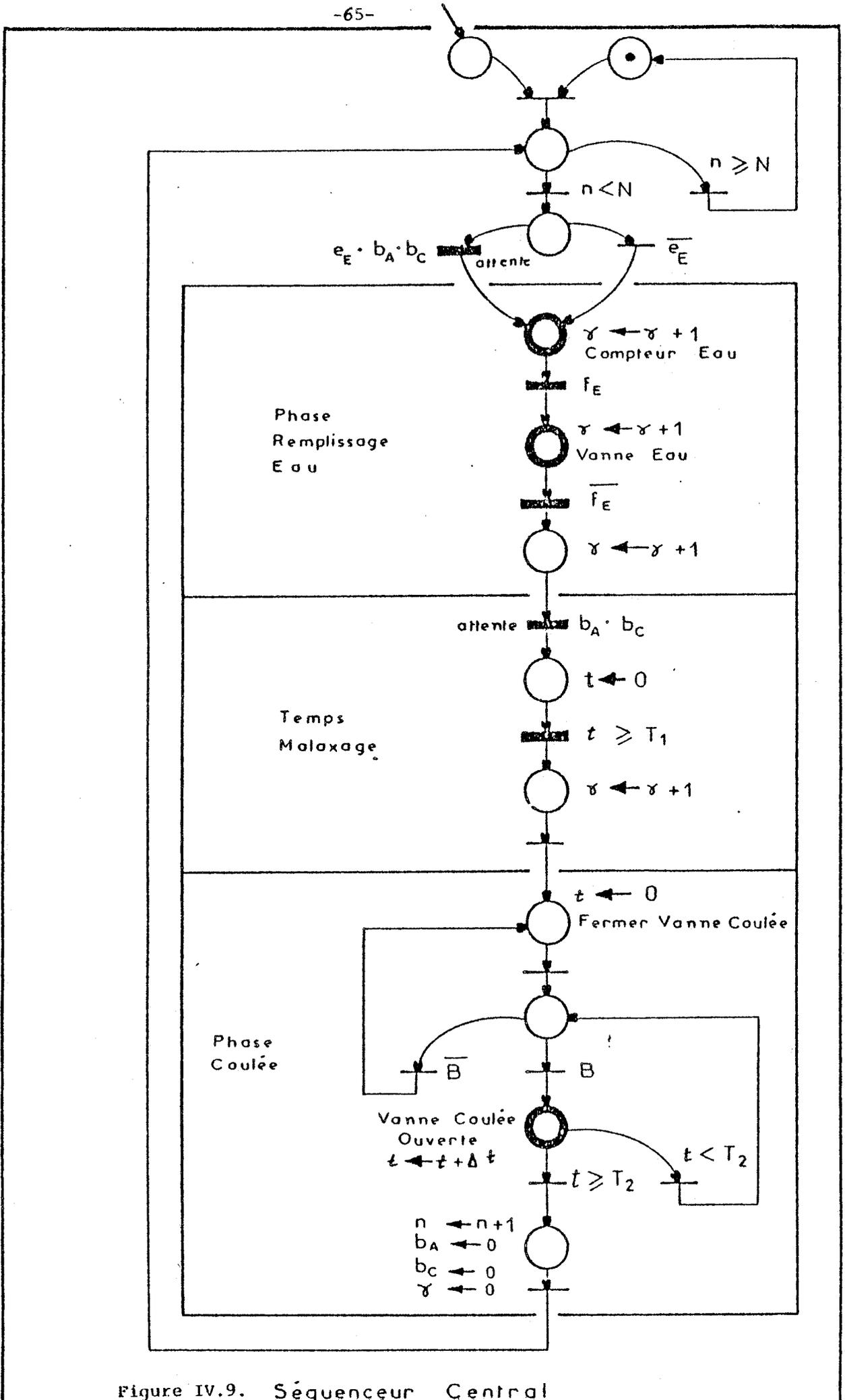


Figure IV.9. Séquenceur Central

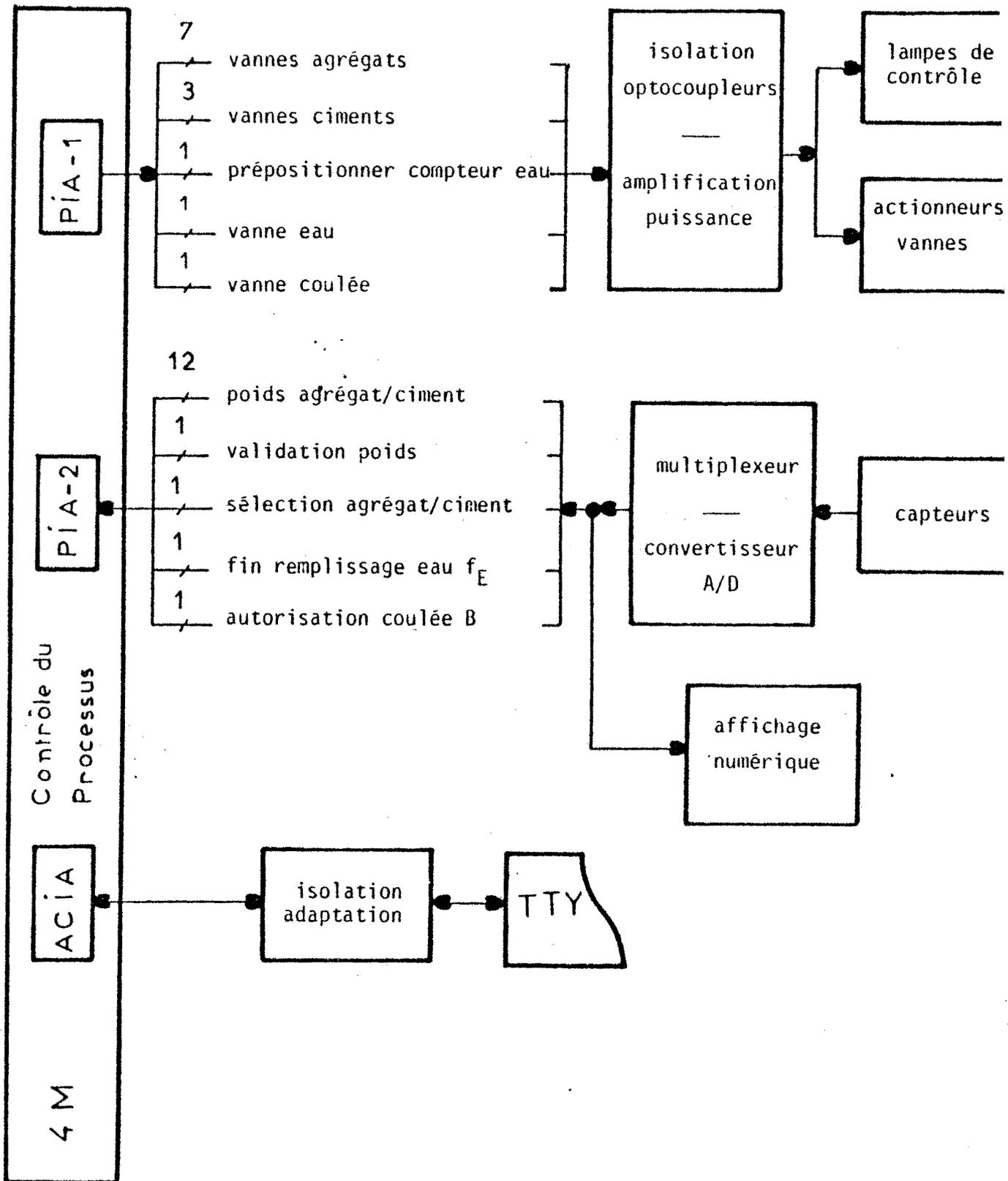


Figure IV.10.

CHAPITRE V
ETUDE DE LA TOLERANCE AUX PANNES

-0-0-

V - 1. INTRODUCTION

Nous étudions dans ce chapitre les propriétés de tolérance aux pannes de l'architecture réalisée qui est décrite au chapitre III.

Nous avons vu au chapitre I que les objectifs et les stratégies de sûreté de fonctionnement pouvaient être multiples : fiabilité, sécurité, disponibilité, dégradation progressive etc... Il n'est pas dans notre objectif de définir les conditions suffisantes que doit réunir une architecture dans chacun de ces cas. En revanche, il est une condition nécessaire que doit satisfaire une architecture, quel que soit l'objectif choisi il convient que les pannes ayant pour effet d'entraîner la perte de l'ensemble des ressources de l'architecture aient un taux aussi faible que possible sinon nul (absence de point dur).

V - 2. EXAMEN DE L'ARCHITECTURE

Cet examen apparaît très évidemment sur les schémas donnés aux figures III.3.1 et III.3.2.

Banc mémoire : une panne de boîtier mémoire ou interface ne fait perdre que la page mémoire correspondante. Il en est de même pour les circuits de sélection et des lignes de communication locales à cette mémoire.

Processeurs : une panne de processeur ne fait disparaître qu'un quart de la puissance de calcul de la machine.

Bus principaux et circuits de sélection : ces circuits étant dédoublés on vérifie qu'une panne conduit à la perte d'un groupe de processeurs ainsi qu'à celle de la mémoire locale à ce groupe.

Horloge et organes de multiplexage temporel : en revanche une panne de cet ensemble mène à la perte de l'ensemble du système. Il s'agit donc d'un point dur, pour lequel des techniques spécifiques de tolérance aux pannes doivent être employées.

V - 3. HORLOGE ET ORGANES DE MULTIPLEXAGE TEMPOREL

Il convient de remarquer, tout d'abord, que ces circuits sont peu nombreux et de complexité faible. Leur taux de panne sera donc peu élevé et cette fiabilité intrinsèque peut être améliorée par l'usage de composants de qualité accrue.

Si néanmoins des fiabilités supérieures sont exigées, ou s'il y a des impératifs de sécurité, deux types de stratégies peuvent être employées :

- la détection localisation, remplacement
- le masquage.

V - 3.1. Détection-localisation-remplacement

Le moyen de détection de pannes proposé consiste à utiliser les invariants suivants qui se définissent du chronogramme de la figure III.1.9.

$$(2f_B) = \overline{2f_A}$$

$$\phi_A(2f_A) \cdot \phi_B = 1$$

$$\phi_A(\overline{2f_A}) \cdot \overline{\phi_B} = 1$$

$$\overline{\phi_A}(2f_A) \cdot \overline{\phi_B} = 1$$

$$\overline{\phi_A}(\overline{2f_A}) \cdot \phi_B = 1$$

On vérifie que ces invariants sont suffisants pour détecter toute panne du système (mise à part l'arrêt de l'horloge).

On peut alors proposer le schéma de la figure V.1 correspondant à une horloge et un distributeur de phases muni d'un autotest fondé sur les invariants, et un système identique en réserve.

La fonction F d'autotest fondée sur la vérification des invariants est définie par le tableau V-2. Un système de temporisation permet de filtrer les pannes transitoires de durée inférieure à τ_2 . En effet, le monostable τ_2 est déclenché en permanence sauf lors d'un arrêt persistant de l'horloge H ou du distributeur de phases.

On peut vérifier sur ce schéma qu'aucune panne simple ne provoque la perte complète du système.

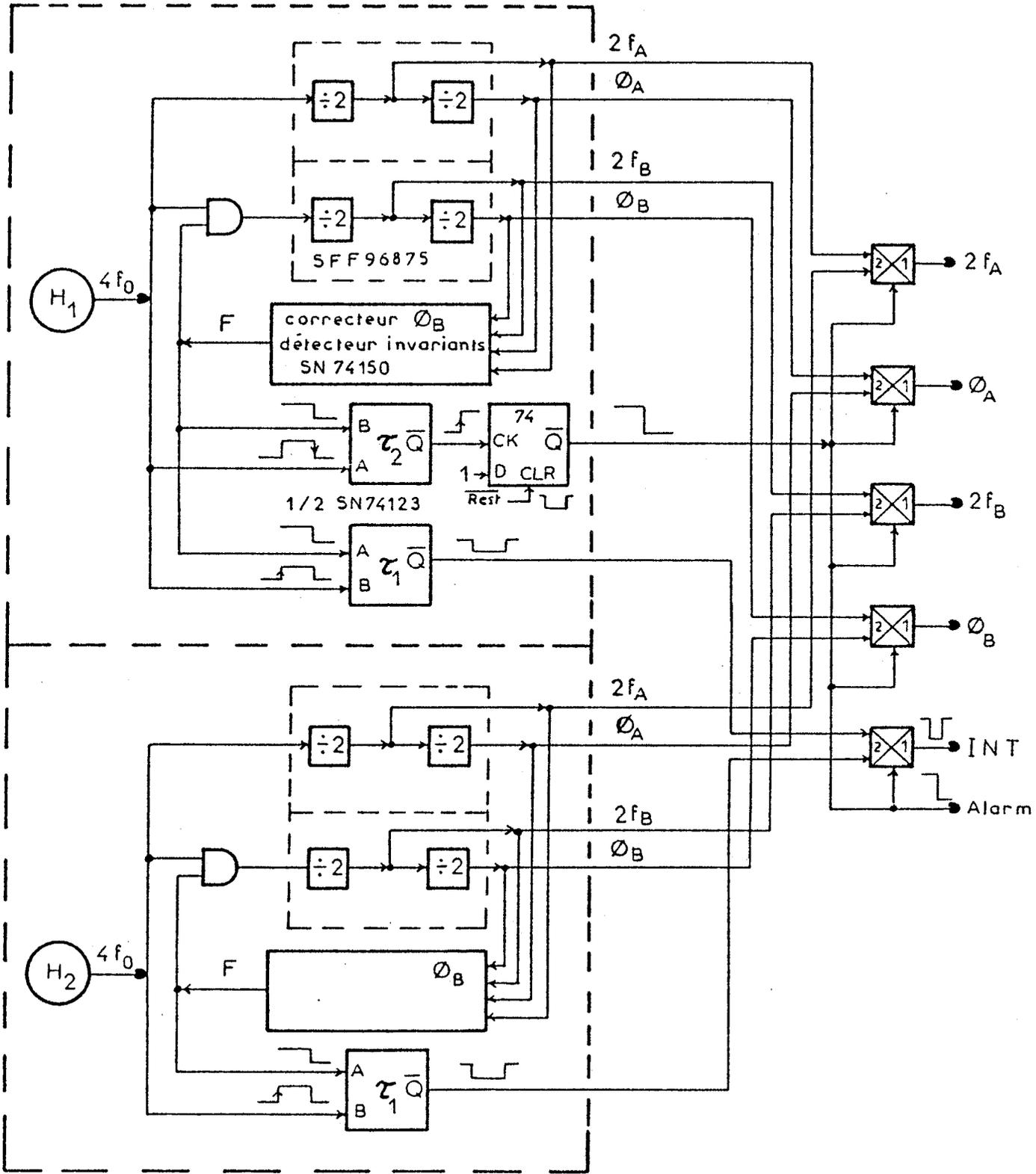


Figure V.1.

\emptyset_B	\emptyset_A	$2f_B$	$2f_A$	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Figure V.2.

En effet, les pannes du vérificateur d'invariants et du système commandant la commutation des multiplexeurs font, soit basculer le système sur l'organe de réserve, qui est alors valide (panne simple), soit rendent le système inapte à basculer, mais cette inaptitude n'aura d'effet que lors d'une panne de l'horloge-distributeur de phases initiales. Enfin, la panne d'un multiplexeur n'entâche qu'un signal de phase ne produisant, au plus, que la perte d'un groupe de processeurs (Fig.III.2.1).

V - 3.2. Masquage

Davies [16] a proposé une horloge synchronisée tripliquée (masquant une panne (Fig. V.3) : une panne simple conduit, au plus, à un signal d'horloge faux parmi les trois de sortie.

A partir de cette horloge on peut construire le système de distributeurs de phase représenté au schéma de la Figure V.4.

Le signal d'horloge 2 permet de générer les signaux $(\overline{2f_A})$ et $\overline{\Phi_A}$. Le signal d'horloge 1 génère les signaux $(2f_A)$ et Φ_A . Le correcteur détecteur d'invariants A resynchronise ces signaux et détecte les discordances (fonction F_1) entre ces deux générateurs. En cas de discordance persistante (supérieure à τ_2), il génère un signal destiné à détruire le groupe de processeurs A.

Le troisième signal d'horloge génère les signaux $2f_B$ et Φ_B qui sont resynchronisés à l'aide des signaux $2f_A$ et Φ_A à travers le correcteur détecteur d'invariants B (fonction F_2) en cas d'erreur persistante (supérieure à τ_2) le groupe de processeurs B est détruit.

Les fonctions F_1 et F_2 sont données à la table V.5.

Il apparaît ainsi que les pannes simples de ces systèmes d'horloge distributeur de phases conduisent, au plus, à la perte d'un groupe de processeurs (ainsi que leur mémoire propre).

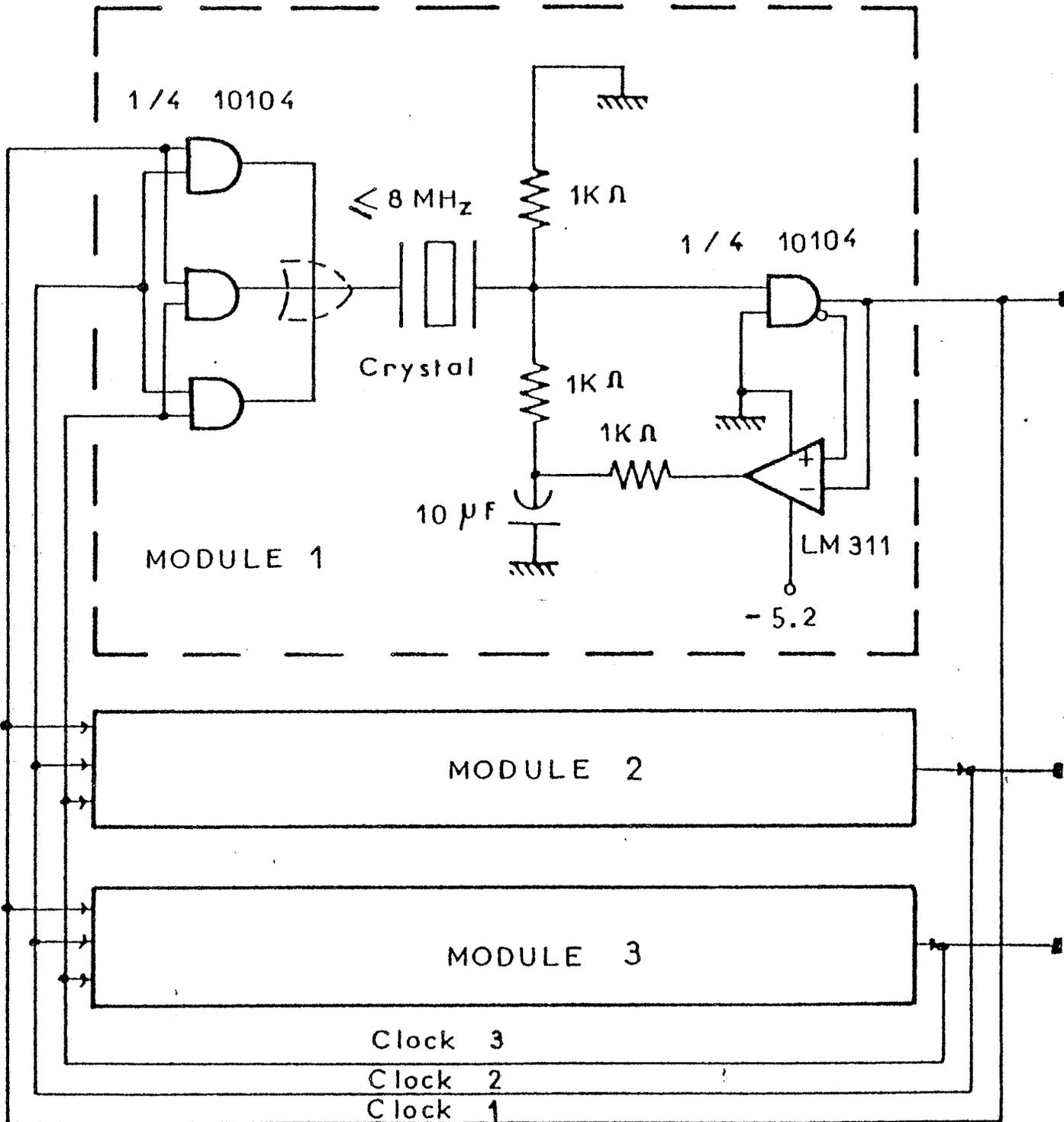


Figure V.3.

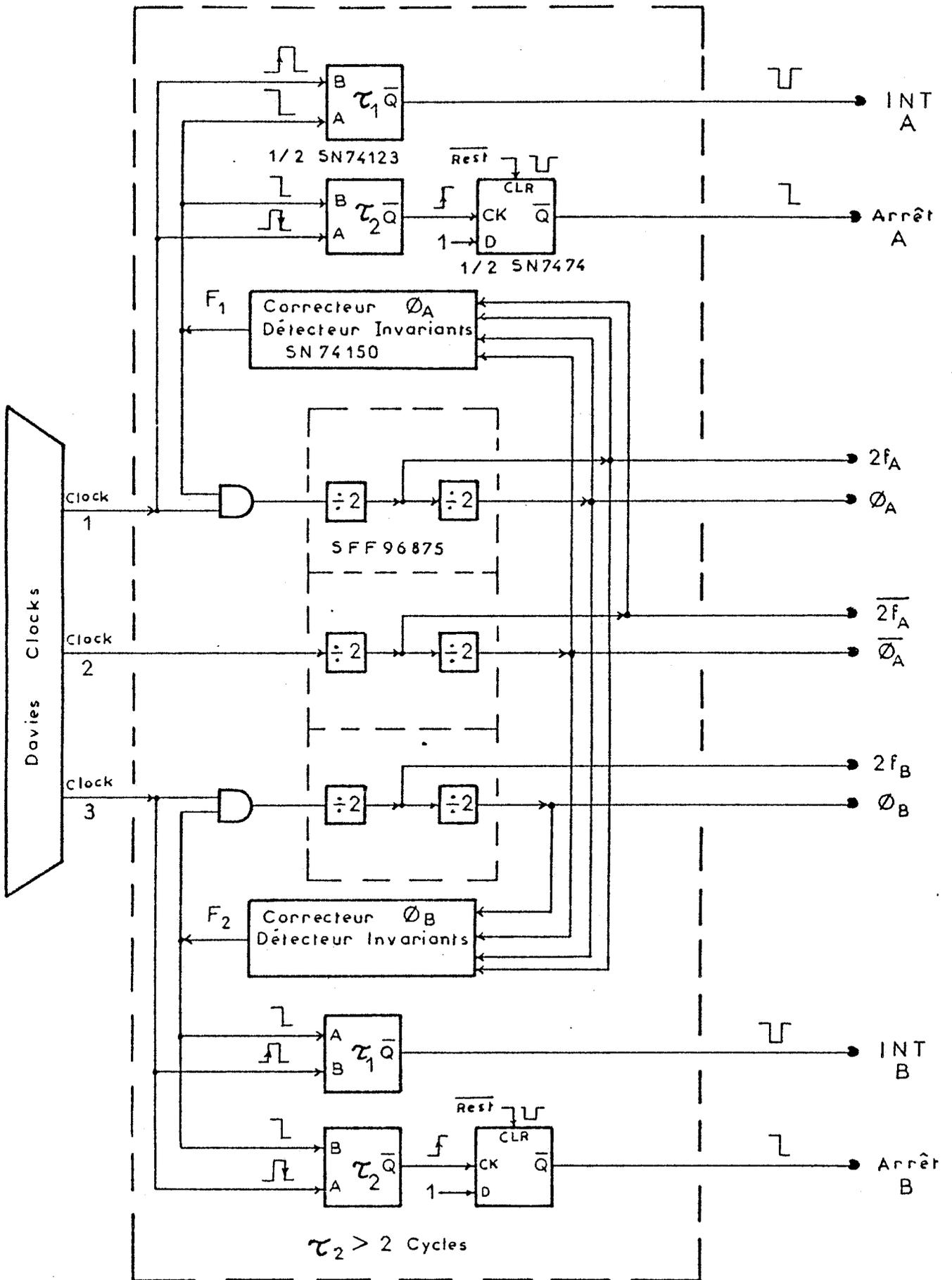


Figure V.4.

$\overline{2f_A}$	$2f_A$	$\overline{\phi_A}$	ϕ_A		F_1
ϕ_B	$2f_A$	$\overline{\phi_A}$	ϕ_A	F_2	
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	1	1	0

Figure V.5.

CHAPITRE VI
EXTENSIBILITE

-0-0-

VI - EXTENSIBILITE

Nous avons vu en I que le principe du multiplexage temporel pouvait poser des problèmes en ce qui concerne son extensibilité. Nous examinerons ici diverses architectures qui permettent de répondre à cette question.

VI - 1. EXTENSIBILITE INTRINSEQUE : Système 8M

1) Horloge et chronogrammes

On dispose actuellement de composants (microprocesseurs 6800, RAM) dont le temps nécessaire pour la lecture du 6800 et celui pour l'accès de la RAM, est inférieur à 1/8 du cycle machine.

On peut donc envisager (III.1) un système à huit microprocesseurs.

Le chronogramme (Fig.VI-1) montre que 8 processeurs travaillant sur des phases décalées de 1/8 cycle peuvent effectuer l'accès à une mémoire dont le temps d'accès est inférieur à 1/8 cycle, sans conflit ni ralentissement.

La figure VI-2 présente un principe d'obtention des signaux ($4f_0$, $2E_A$, $2E_C$, ϕ_A , ϕ_B , ϕ_C , ϕ_D) à partir d'une horloge $8 f_0$, qui seront utilisés pour la synchronisation des accès mémoire.

2) Répartition des mémoires

Afin de diminuer le coût du système, les mémoires rapides coûtant cher, on peut proposer la répartition mémoire suivante (Fig. VI-3) :

- a) Une zone M_1 de taille limitée dont le temps d'accès est inférieur à 1/8 de cycle : cette zone sert aux communications au sein de la machine.
- b) Deux zones M_{21} , M_{22} de temps d'accès inférieur à 1/4 de cycle. Ces zones pourraient être destinées à former la mémoire centrale du système : (mémoire de travail, programmes et données).
- c) 4 zones de temps d'accès inférieur à 1/2 cycle (M_{3A} , M_{3B} , M_{3C} , M_{3D}) spécifiques à chaque couple de processeurs A, B, C, D.

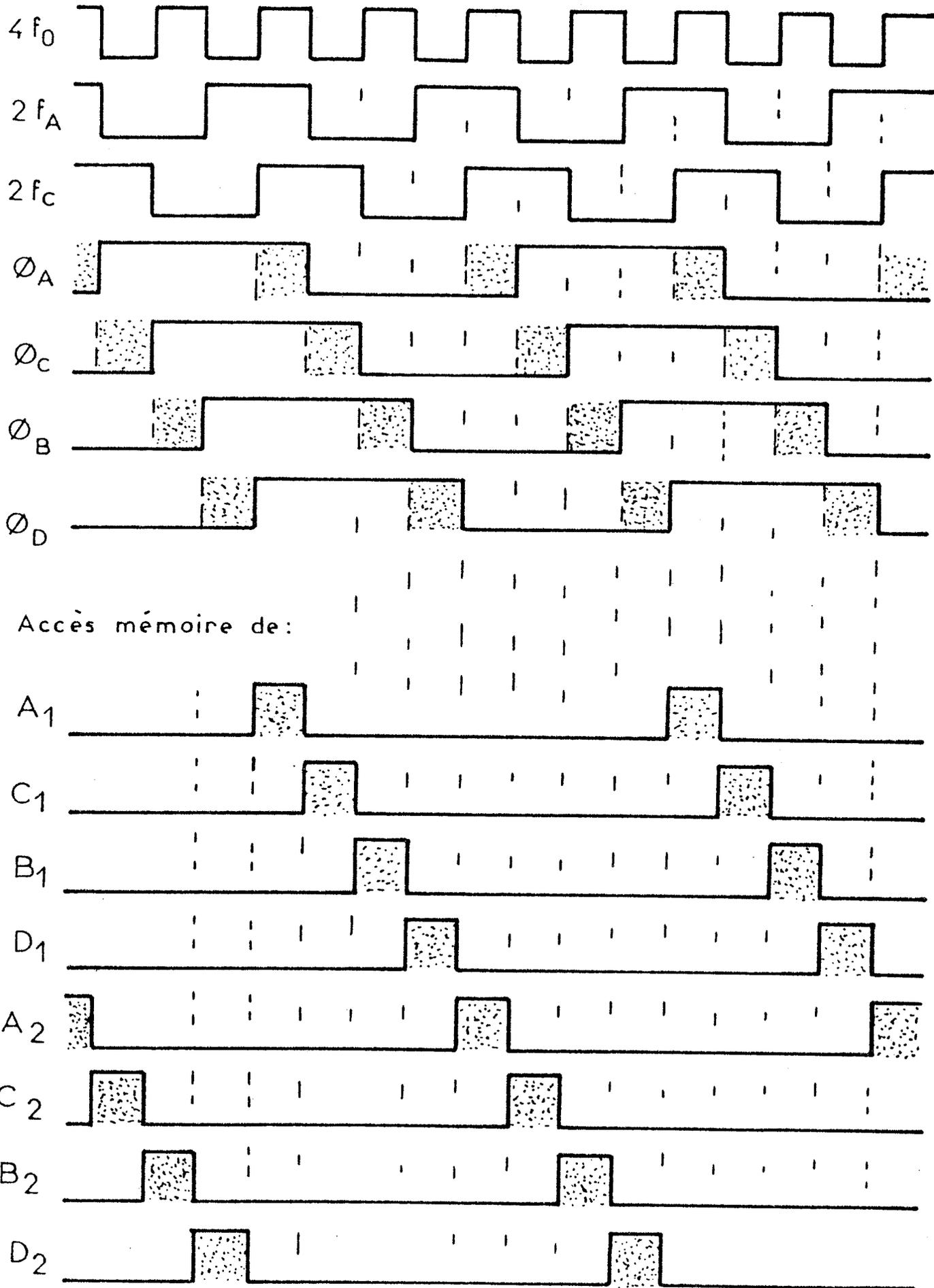


Figure VI.1.

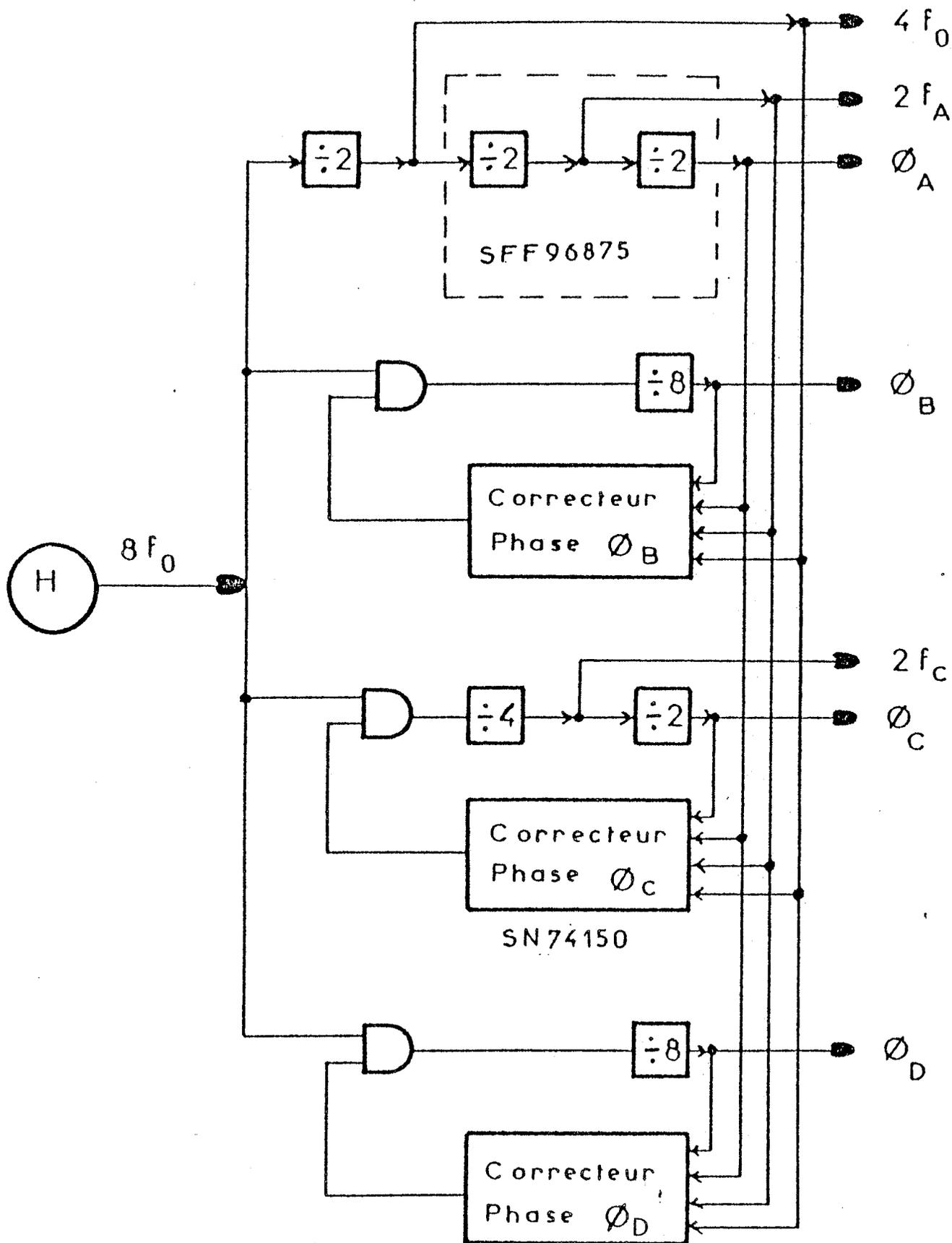


Figure VI.2.

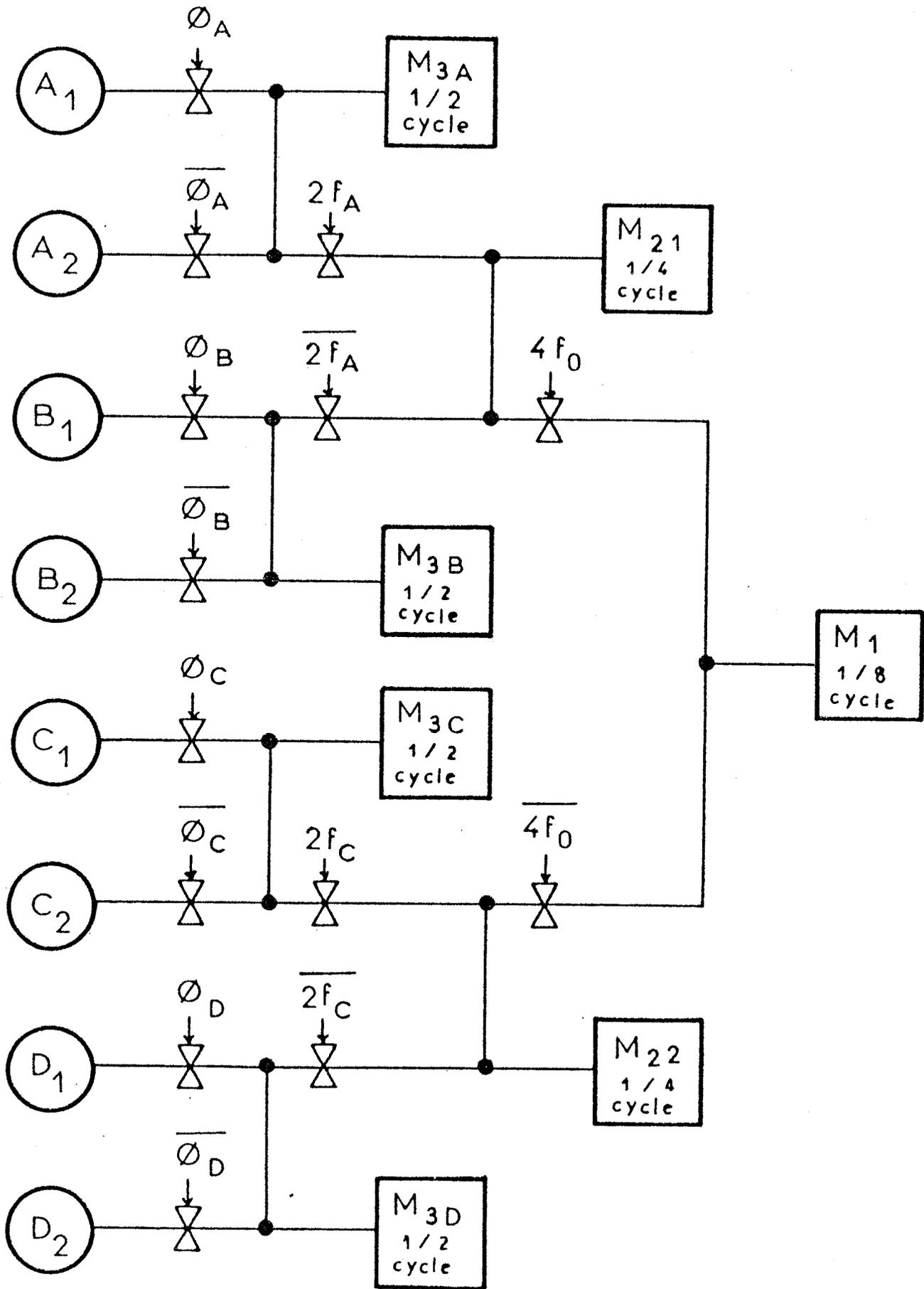


Figure VI.3.

VI - 2. ARCHITECTURES MOLECULAIRES FONDEES SUR LE SYSTEME 4M

Une deuxième possibilité d'extension consiste à assembler entre elles plusieurs molécules 4M se partageant un ou plusieurs processeurs.

VI - 2.1. Proposition 1

Cette première proposition représentée à la figure VI-4 comprend 4 molécules 4M partageant toutes un processeur commun A_{11} qui a accès à toutes les mémoires communes et peut ainsi assurer la communication entre tous les processeurs.

Ce système comprend donc 13 processeurs pouvant travailler en parallèle avec une capacité d'adressage totale comprise entre 260 et 504 K octets suivant la répartition des zones mémoire (Fig. VI-5).

Cette architecture très performante présente cependant une tolérance aux pannes faible, le processeur A_{11} se comportant de ce point de vue là comme un point dur.

VI - 2.2. Proposition 2

On peut obtenir une architecture ayant une plus grande tolérance aux pannes en regroupant 4 molécules 4M avec deux processeurs communs A_{11} et B_{11} appartenant aux 4 molécules. Ainsi la perte d'un de ces processeurs (ou d'une mémoire) ne supprime aucune communication dans cet ensemble de 10 processeurs (Fig. VI-6).

VI - 2.3. Proposition 3: Accès direct mémoire DMA

La figure VI-7 propose l'utilisation du circuit DMA pour des transferts périodiques et assez rapides (débit au moins 1 M octets/sec) de blocs d'information entre quatre molécules "4M" ou ces molécules et l'extérieur par interface d'entrée-sortie.

Le procédé du DMA repose sur l'utilisation des commandes "halt" ou "TSC" du 6800 pour effectuer un "halt bloqué" ou "halt temporaire" ou "vol de cycle".

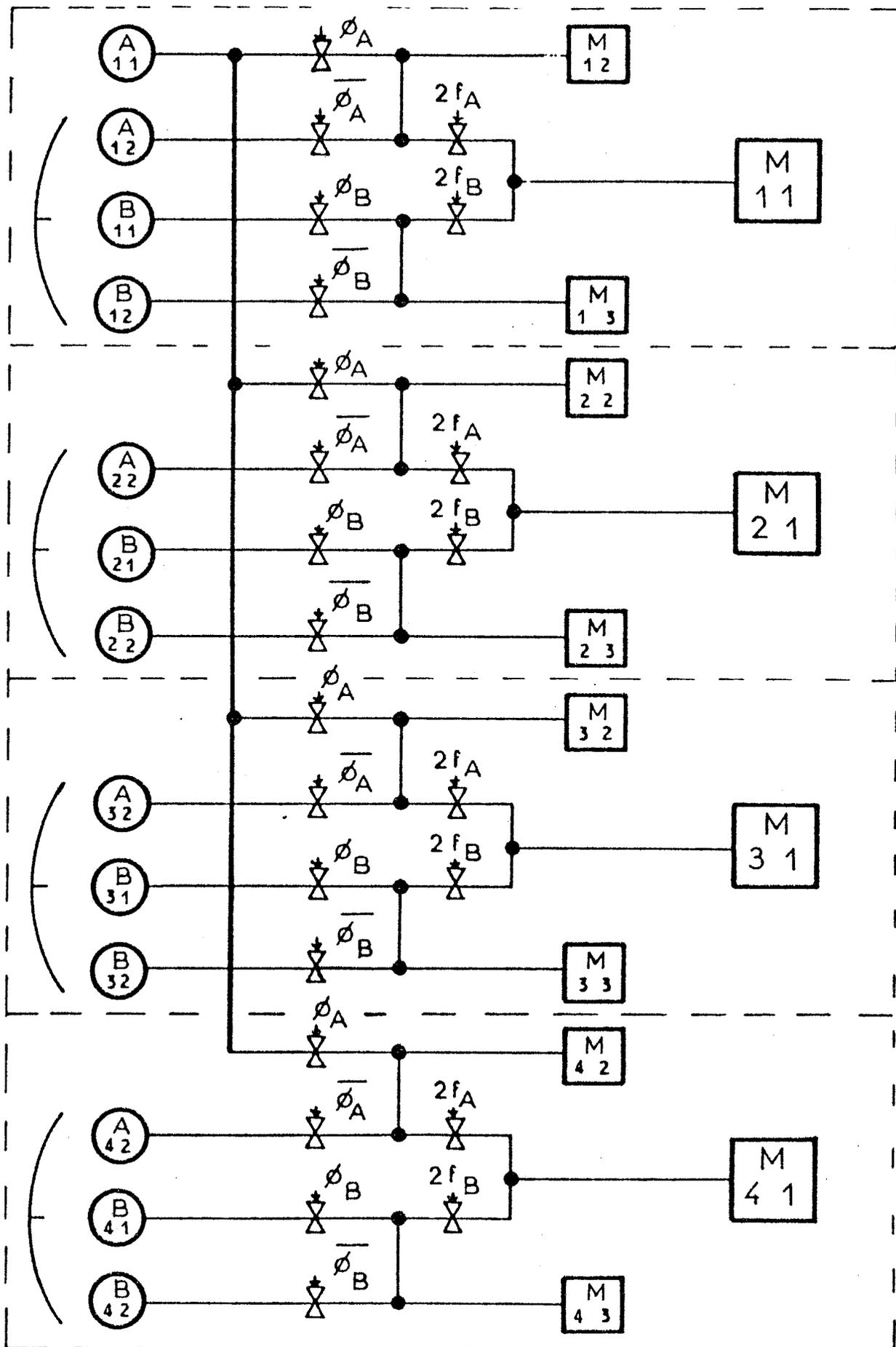


Figure VI.4.

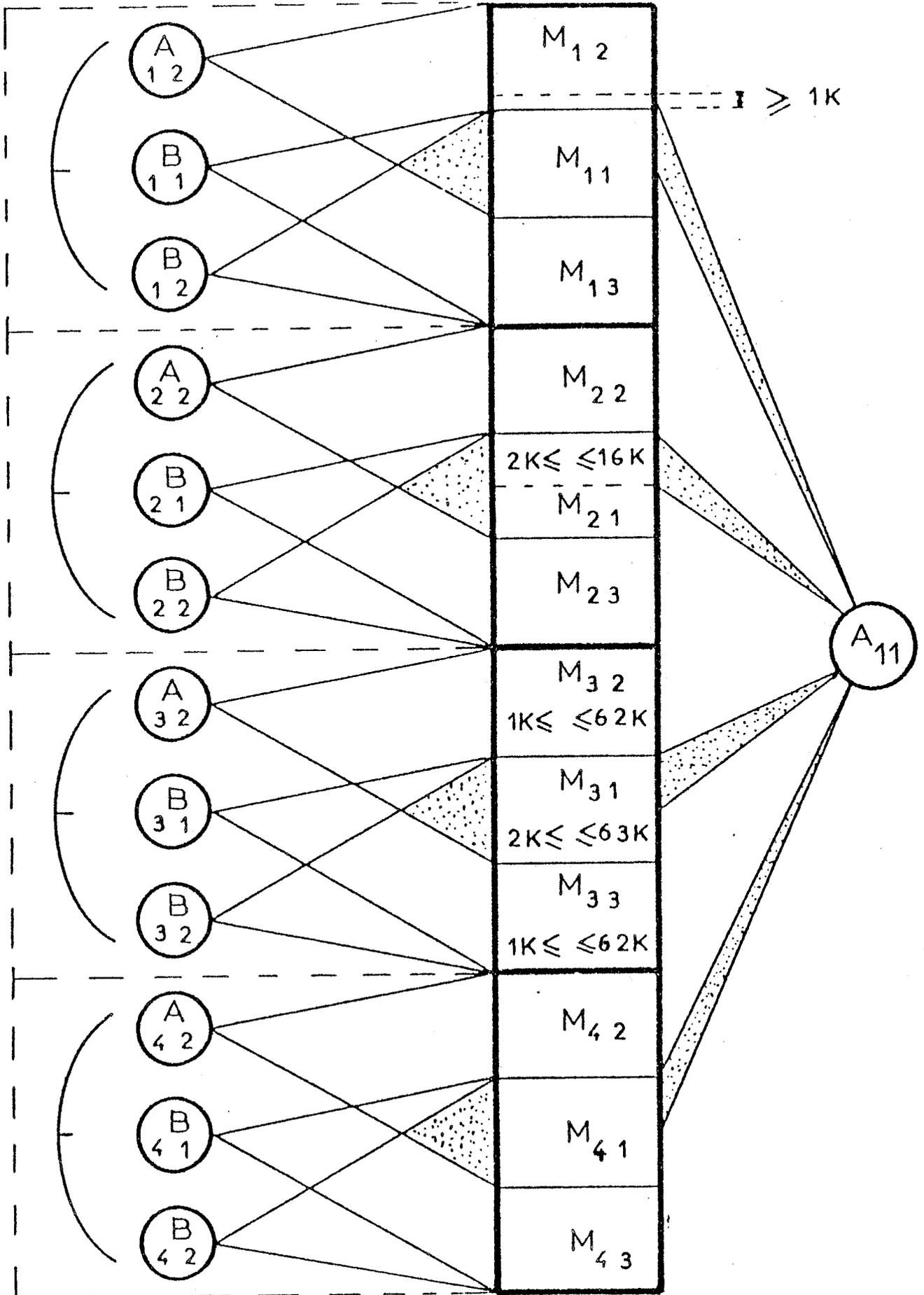


Figure VI.5.

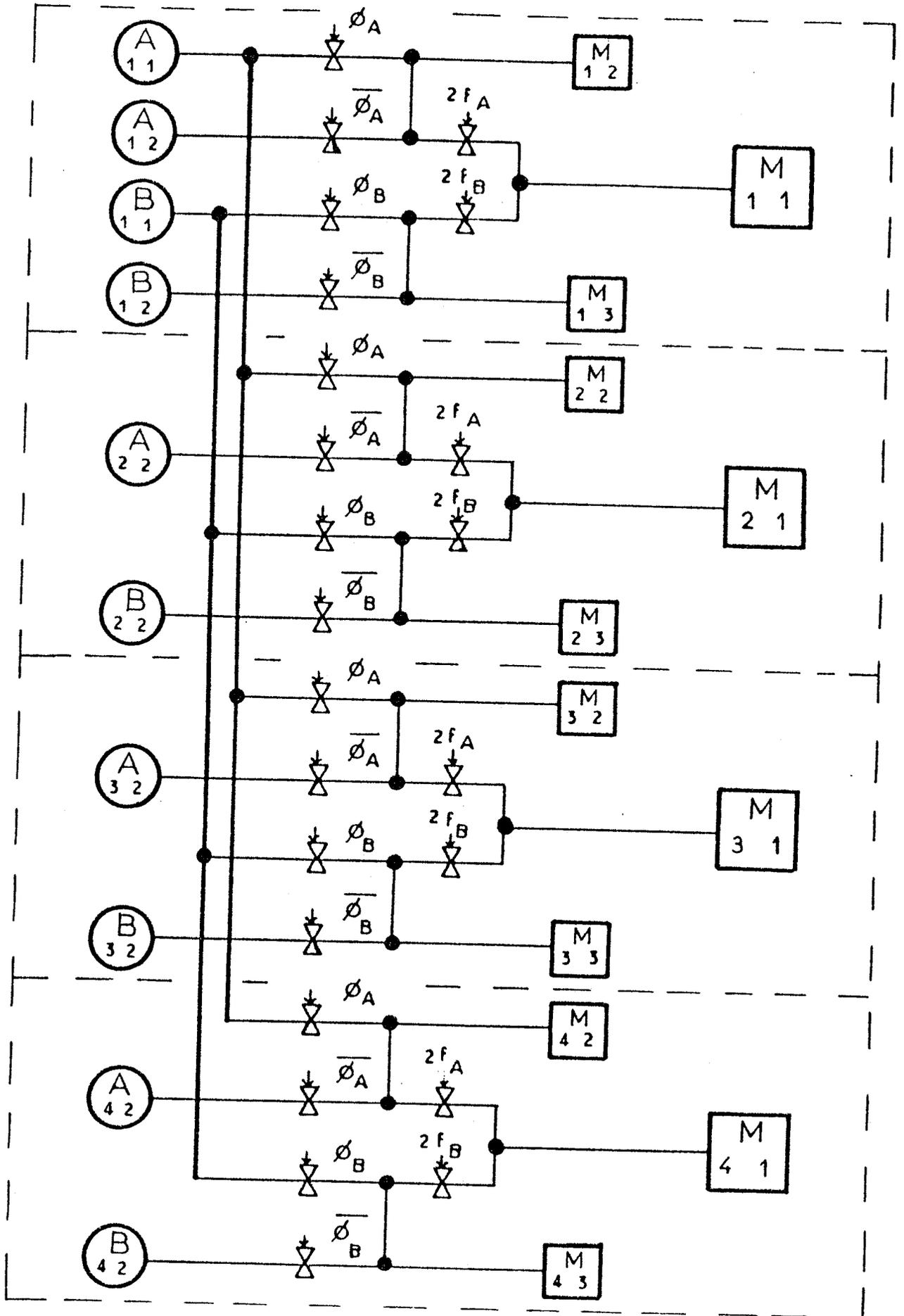


Figure VI.6.

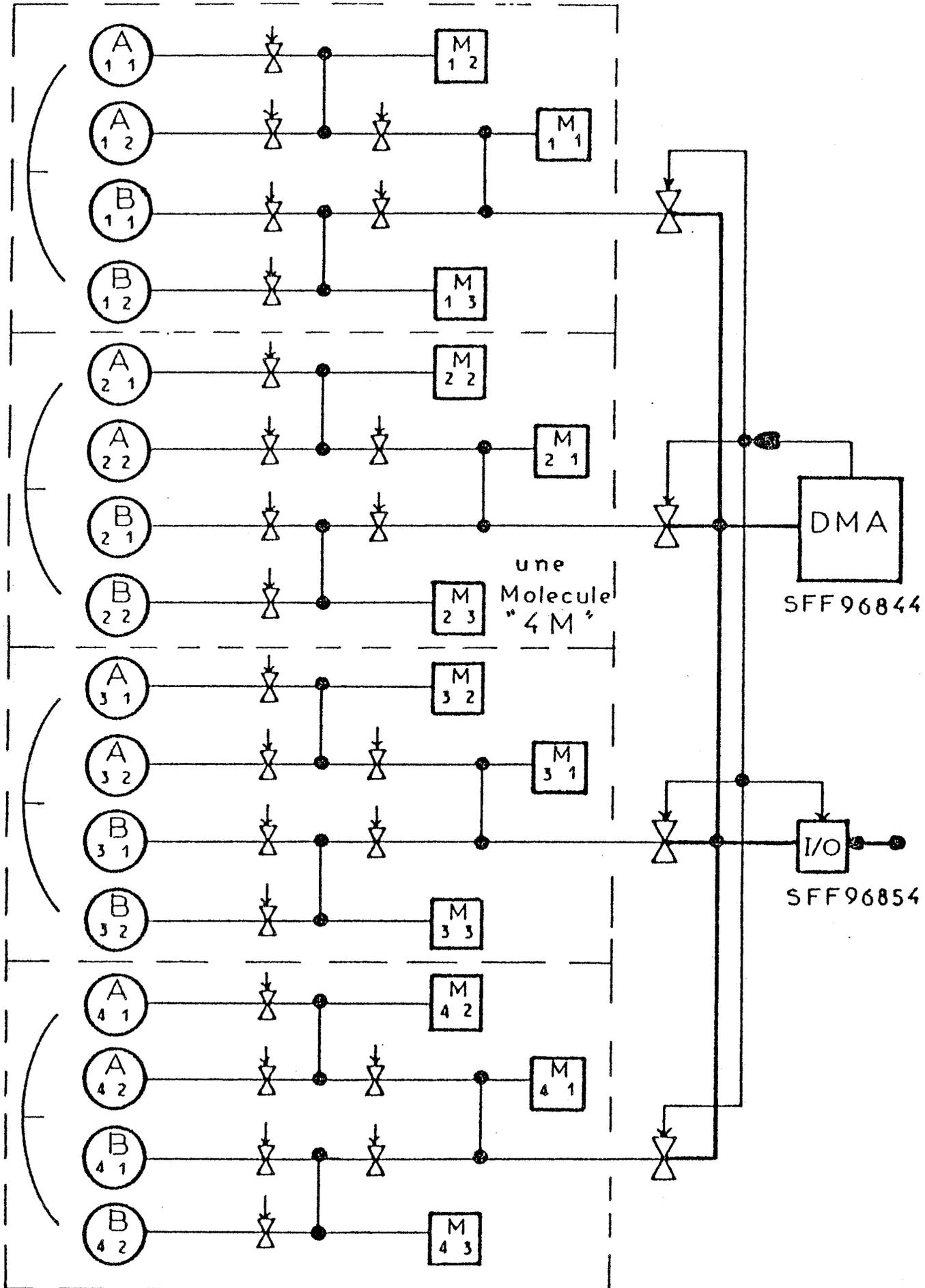


Figure VI.7.

Cette proposition mélange les deux techniques :

- Bus synchrone : multiplexage temporel, pour permettre des communications très fréquentes et rapides, à l'intérieur de chacune des molécules.
- Bus asynchrone : pour permettre le transfert des requêtes de taille importante, entre ces molécules.

VI - 3. GENERALISATION

VI - 3.1. Définition

Nous avons vu au paragraphe VI.2 quelques exemples d'interconnexion de molécules 4M par partage de un ou plusieurs processeurs. On peut généraliser ce principe de la façon suivante :

De façon simplifiée, une molécule "4M" peut se représenter sous la forme d'un graphe biparti non orienté (Fig.VI.8)

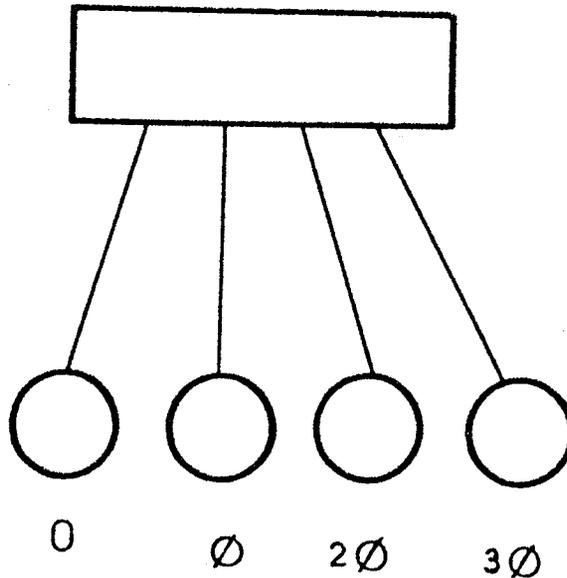


Figure VI.8.

Dans cette figure le noeud "case" représente la mémoire commune et les noeuds "rond" les processeurs. D'après le principe de multiplexage temporel, si la mémoire a un temps d'accès inférieur à $\emptyset = 1/4$ de cycle machine, on doit associer à chaque processeur un déphasage égal à $0, \emptyset, 2\emptyset$ ou $3\emptyset$. Il n'y aura pas de conflit si les 4 processeurs ont des déphasages distincts.

La généralisation de 4M consistera donc en un graphe biparti quelconque avec la restriction suivante :

il faut pouvoir associer à chaque processeur un déphasage valant $0, \emptyset, 2\emptyset$ ou $3\emptyset$ tel que chaque mémoire ne soit pas adjacente à deux processeurs ayant le même déphasage.

Il est évident que dans un tel graphe il n'y aura jamais de conflits d'accès mémoire.

Comme corollaire de cette propriété il est clair qu'une mémoire ne pourra pas être adjacente à plus de quatre processeurs.

VI - 3.2. Exemple

La figure VI.9 présente un exemple de huit molécules "4M" connectées en anneau.

Ce réseau comprend huit processeurs (sur chaque processeur on a indiqué le déphasage par les valeurs $0, 1, 2$ ou 3). Chaque processeur a une capacité d'adressage de 64 K octets et doit adresser 4 mémoires communes. Donc chaque mémoire commune a une capacité inférieure ou égale à 16 K octets.

L'ensemble a donc une capacité mémoire de 128 K octets.

Par le grand nombre de redondances qu'il contient, cet ensemble devrait posséder des propriétés de tolérance aux pannes intéressantes.

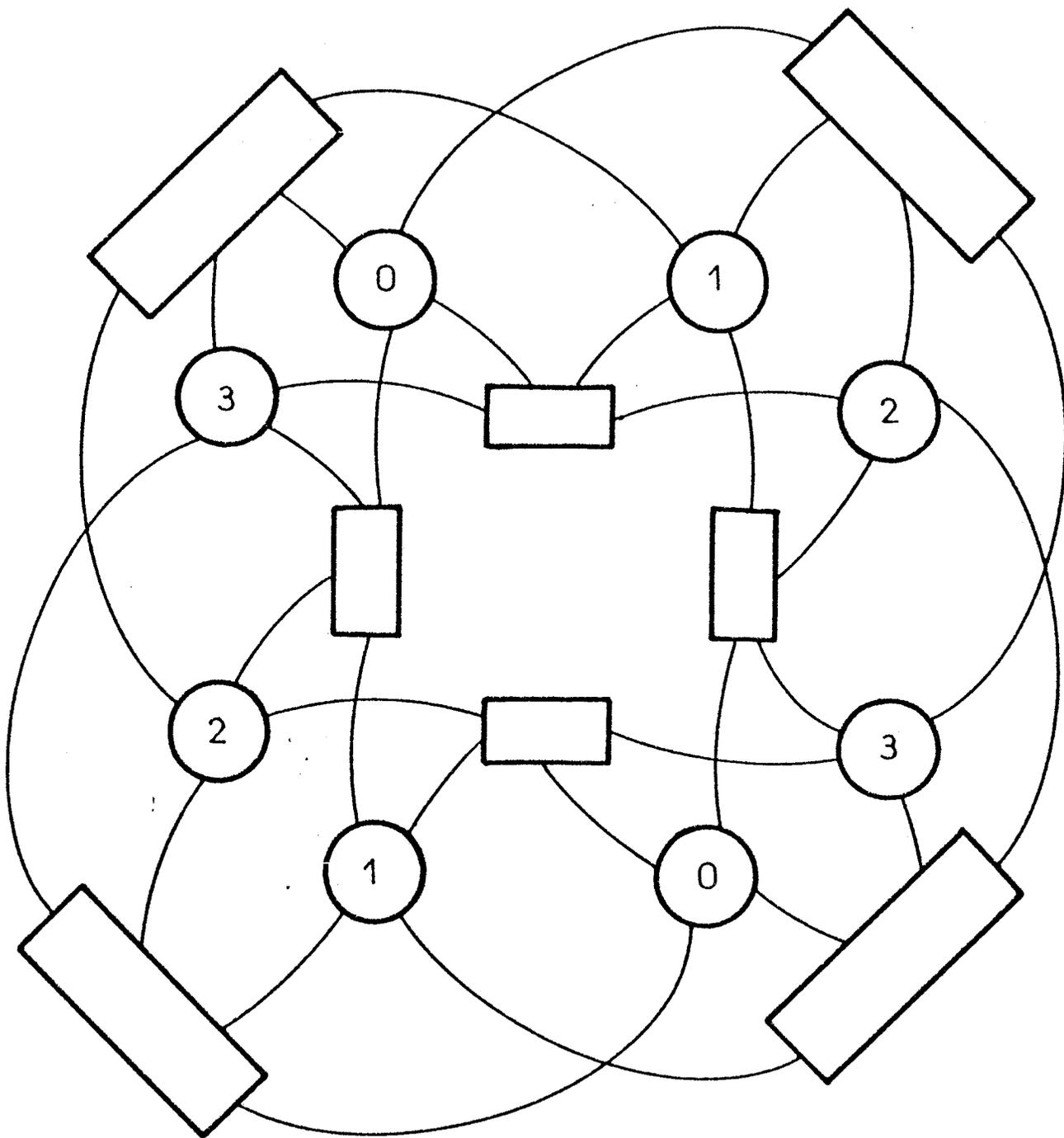


Figure VI.9.

VI - 3.3. Condition nécessaire et suffisante d'existence

On appellera graphe d'interconnexion des processeurs le graphe ordinaire dont les noeuds sont les processeurs et tel qu'il y a une arête entre deux noeuds chaque fois que ces processeurs partagent une mémoire commune dans le graphe biparti initial.

Une condition nécessaire et suffisante pour que ce graphe d'interconnexion soit réalisable en "4M" est alors que ce graphe soit 4-coloriable.

On a alors le corollaire amusant suivant : tout graphe d'interconnexion planaire est réalisable en "4M".

CONCLUSION

CONCLUSION

Nous avons montré dans ce travail qu'une réalisation simple et efficace d'un système multimicroprocesseur est possible; cette solution tirant des "performances" maximales d'un matériel actuellement disponible. Le conflit d'accès mémoire a été résolu de façon précise, simple et transparente à l'utilisateur. Une attention particulière a été apportée aux possibilités d'utilisation de cette architecture pour les systèmes temps réel (étude d'un logiciel spécialisé) et aux problèmes de sûreté de fonctionnement. Cette structure est un compromis coût-réalisation rapide-simplicité-flexibilité-sûreté de fonctionnement. En particulier elle a permis de cerner les points faibles d'une telle structure et de définir, à la lueur de cette expérience, une extension ayant de meilleures caractéristiques de tolérance aux pannes.

ANNEXE A

PROGRAMMATION DIRECTE DE L'EXEMPLE DE
LA CENTRALE A BETON

A.1. INTRODUCTION

On note qu'il est plus facile d'implémenter une application, dont les tâches peuvent s'exécuter parallèlement en temps réel, sur une structure multi-processeurs (à mémoire commune sans conflit d'accès) que sur un seul microprocesseur, spécialement si le nombre des tâches est égale à celui des processeurs.

La figure A.1 présente une allocation fixe des tâches de la centrale à béton (IV-2) sur les processeurs de "4M". Ainsi que la tâche de simulation de l'environnement sur un microprocesseur M6800 externe. Ce dernier teste l'état des sorties de commandes pour simuler les capteurs (Fig.IV.7).

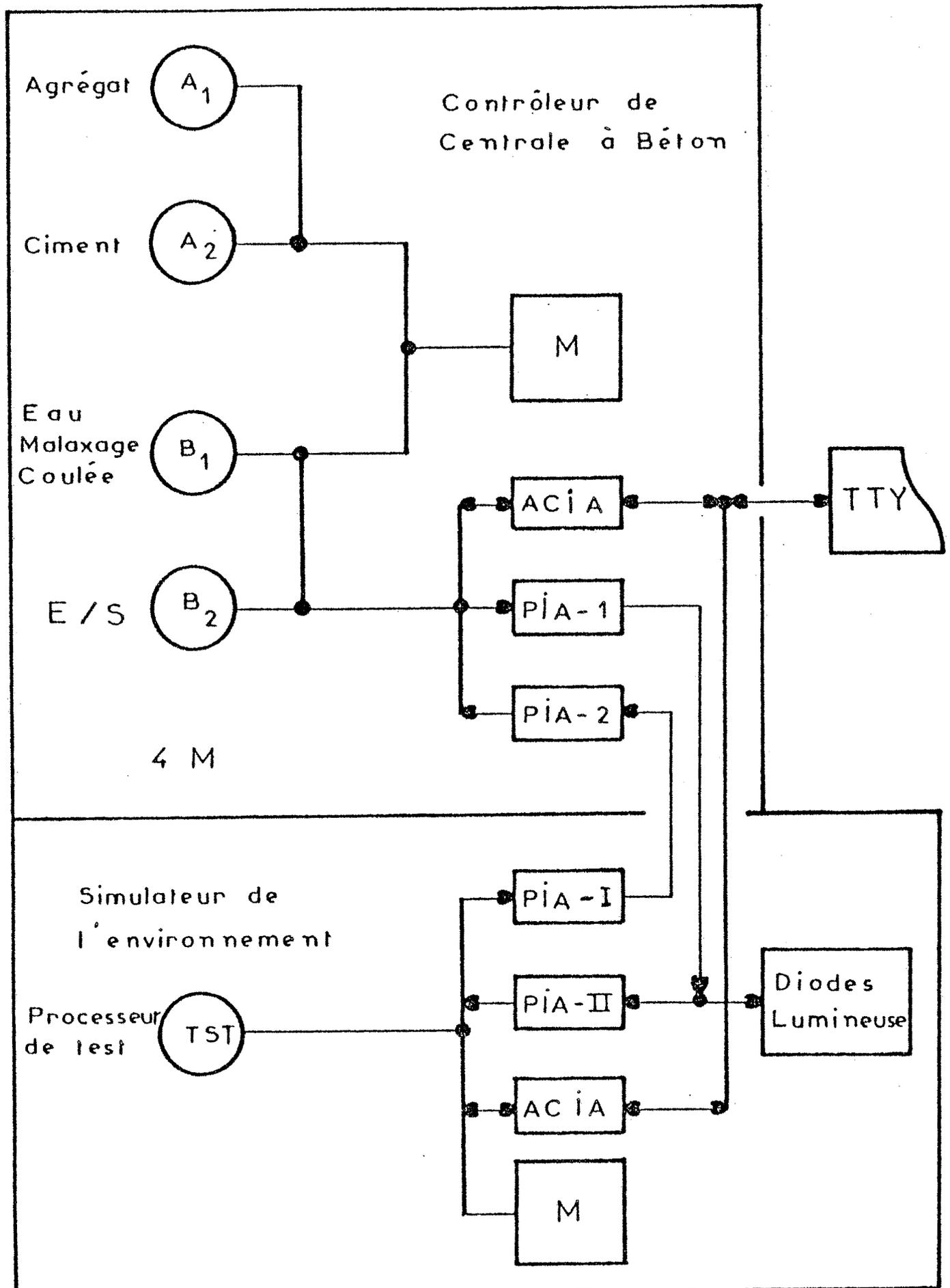


Figure A.1.

A.2. ALGORITHMES

Agrégats/ciments

Itérer jusqu'à (fin cycle fabrication)

Initialisation

Itérer

 cycle pesées

jusqu'à i = nb . var.

 Fin itérer

 attente vidange

 phase vidange

 mot synchronisation ← vrai

Fin itérer

Eau/Malaxage

Itérer jusqu'à fin cycle fabrication

 contrôle eau

 contrôle malaxage

 phase coulée béton

 nb cycles fabrication ← nb c. +1

 mots synchr. ← faux

Fin itérer

Contrôle eau

SI mot synchr. eau = 0

Alors si contrôle eau = 1

alors attendre synchronisation

 Fin

 Phase remplissage eau

 Mot synchrone. eau ← 1

Fin

Fin contrôle eau

Contrôle malaxage

Attendre synchronisation

Comptage temps malaxage

Fin contrôle malaxage

Attendre synchronisation

Itérer jusqu'à mot A = 1

Itérer jusqu'à mot C = 1

Fin attendre synchronisation

Simulation d'un capteur

Itérer jusqu'à fin simulation

Lecture PIA 1

Suivant pos. bit activé faire

calcul fonction poids

écriture PIA2

Fin itérer

Entrée/Sortie

Itérer

Lecture mémoire commune

Ecriture PIA 1

Lecture PIA 2

Ecriture mémoire commune

Fin itérer

A.3. LISTINGS

PARC 001 WORLD

```

*
* PARTIE ADRESSES
*
00000
00001
00002
00003
00004
00005
00006
00007 2700 A N COU 12700 NOMBRE DE CYCLES FABRIC. A
00008 2701 A N1 COU 12701
00009 2704 A NOTA COU 12704 ELEMENT DE SYNCHRON.
00010
00011 0000
00012
00013 0000 0000 A PAGR RMD 12 POIDS DE L'AGREGAT
00014 0000 0002 A ADRA RMD 2
00015 0000 0002 A ADRA RMD 2
00016 0010 0002 A CAPTA RMD 2 CAPTEUR D'AGREGAT
00017 0012 0002 A AGR RMD 6 TABLEAU SELECTION D'AGREG.
00018 0010 0001 A DA RMD 1
00019 0010 0002 A REFA RMD 2 REFCR POIDS AGREGAT
00020
00021 0000
00022 0000 P AGREG COU *
00023
00024
00025
* INITIALISATIONS
*
00026 0000 7F 0010 D INITA CLR REFA
00027 0000 7F 001A D CLR REFA+1 REFA = 0
00028 0000 CE 0012 D LDX #ADR
00029 0000 FF 0001 D STX ADRA
00030 0000 CE 0000 D LDX #PAGR
00031 0000 FF 000C D STX ADRPA
00032 0012 CE 01 A LDAA #1
00033 0014 D7 0010 D STAA DA
00034
00035
* ITERER
*
00036 0017 00 2700 A TSTA1 LDAA N
00037 001A 4A DLEA
00038 0010 01 2701 A CMPA #1 N-1 > N1 ??
00039 001C 22 10 0000 DHI ITA1

```

PARC 002 AGREG

```

00040 0000 4C
00041 0001 01 2701 A CMPA #1 N > N1 ??
00042 0004 22 00 0000 DHI TSTA2
00043 0005 7E 0004 P FFA JMP FINA
00044 0009 00 2704 A TSTA2 LDAA NOTA
00045 000C 01 01 A CMPA #1
00046 000E 27 FC 000C DLEA FIA
00047 0010 FC 000E D ITA1 LDX ADRA
00048 0013 00 00 A LDAA 0.X
00049 0015 01 00 A CMPA #0 AGREG(0) = 0 ??
00050 0017 27 10 0004 DLEA JUSDA
00051 0019 FC 000C D LDX ADRPA
00052 001C AC 01 A LDAA 1.X
00053 001E 00 001A D ADDA REFA+1
00054 001F 07 001A D STAA REFA+1
00055 0021 AC 00 A LDAA 0.X
00056 0023 00 0019 D ADCA REFA
00057 0025 D7 0019 D STAA REFA REFA (--- REFA+PAGR(0))
00058
00059
* PESEE AGREGATS
*
00060 0040 FC 0010 D LDX REFA
00061 0041 00 0010 D ITA2 CPX CAPTA REFA > CAPTA ??
00062 0042 22 FC 004F DHI ITA2
00063
00064
* FIN PESEE
*
00065 0004 7C 0010 D JUSDA INC BA
00066 0007 FC 0010 D LDAB BA
00067 000A 01 00 A CNPD #C DA > C ??
00068 000C 22 11 000F DHI ITA3
00069 000E FC 000E D LDX ADRA
00070 000F 00 INX
00071 0010 FC 000C D STX ADRA ADRA (--- ADRA+1)
00072 0011 FC 000C D LDX ADRPA
00073 0012 00 INX
00074 0013 00 INX
00075 0015 FC 000C D STX ADRPA ADRPA (--- ADRPA+2)
00076 001D 20 01 0000 DRA ITA1
00077

```

PAGE 000 ACRLC

```

00001          * FIN ITERER
00002          *
00003          *
00004          * ATTENDRE SYNCHRON
00005          *
00006 00000 0000 00 2704 A 1TAS LDAA  #0TA
00007 00000 0000 01 00 A      CNPA  #0
00008 00000 0000 20 09 0000      CHC   1TAS
00009          *
00010          * VIDANGER
00011          *
00012 00000 0000 00 0000 D 1TAS LDX   CAPTA
00013 00000 0000 00 0000 A      CPX   #0
00014 00000 0000 22 10 0000      DMI   1TAS
00015          *
00016 00000 0000 70 2704 A      INC   #0TA  #0TA 1 1
00017 00000 0000 70 0000 P      JMP   INITA
00018 00000 0000 01          FINA  RUP
00019          *
00020          *
00021 00000          0000 P      END   AGNEC
00022          *
TOTAL ERRORS 00000
    
```

PAGE 001 C1RLN

```

00001          *
00002          * PARTIE CEMENTS
00003          *
00004          *
00005          *
00006          *
00007          *
00008          *
00009          *
00010          *
00011          *
00012          *
00013          *
00014          *
00015          *
00016          *
00017          *
00018          *
00019          *
00020          *
00021          *
00022          *
00023          *
00024          *
00025          *
00026          *
00027          *
00028          *
00029          *
00030          *
00031          *
00032          *
00033          *
00034          *
00035          *
00036          *
00037          *
00038          *
00039          *
00040          *
00041          *
00042          *
00043          *
00044          *
00045          *
00046          *
00047          *
00048          *
00049          *
00050          *
00051          *
00052          *
00053          *
00054          *
00055          *
00056          *
00057          *
00058          *
00059          *
00060          *
00061          *
00062          *
00063          *
00064          *
00065          *
00066          *
00067          *
00068          *
00069          *
00070          *
00071          *
00072          *
00073          *
00074          *
00075          *
00076          *
00077          *
00078          *
00079          *
00080          *
00081          *
00082          *
00083          *
00084          *
00085          *
00086          *
00087          *
00088          *
00089          *
00090          *
00091          *
00092          *
00093          *
00094          *
00095          *
00096          *
00097          *
00098          *
00099          *
00100          *
00101          *
00102          *
00103          *
00104          *
00105          *
00106          *
00107          *
00108          *
00109          *
00110          *
00111          *
00112          *
00113          *
00114          *
00115          *
00116          *
00117          *
00118          *
00119          *
00120          *
00121          *
00122          *
00123          *
00124          *
00125          *
00126          *
00127          *
00128          *
00129          *
00130          *
00131          *
00132          *
00133          *
00134          *
00135          *
00136          *
00137          *
00138          *
00139          *
00140          *
00141          *
00142          *
00143          *
00144          *
00145          *
00146          *
00147          *
00148          *
00149          *
00150          *
00151          *
00152          *
00153          *
00154          *
00155          *
00156          *
00157          *
00158          *
00159          *
00160          *
00161          *
00162          *
00163          *
00164          *
00165          *
00166          *
00167          *
00168          *
00169          *
00170          *
00171          *
00172          *
00173          *
00174          *
00175          *
00176          *
00177          *
00178          *
00179          *
00180          *
00181          *
00182          *
00183          *
00184          *
00185          *
00186          *
00187          *
00188          *
00189          *
00190          *
00191          *
00192          *
00193          *
00194          *
00195          *
00196          *
00197          *
00198          *
00199          *
00200          *
00201          *
00202          *
00203          *
00204          *
00205          *
00206          *
00207          *
00208          *
00209          *
00210          *
00211          *
00212          *
00213          *
00214          *
00215          *
00216          *
00217          *
00218          *
00219          *
00220          *
00221          *
00222          *
00223          *
00224          *
00225          *
00226          *
00227          *
00228          *
00229          *
00230          *
00231          *
00232          *
00233          *
00234          *
00235          *
00236          *
00237          *
00238          *
00239          *
00240          *
00241          *
00242          *
00243          *
00244          *
00245          *
00246          *
00247          *
00248          *
00249          *
00250          *
00251          *
00252          *
00253          *
00254          *
00255          *
00256          *
00257          *
00258          *
00259          *
00260          *
00261          *
00262          *
00263          *
00264          *
00265          *
00266          *
00267          *
00268          *
00269          *
00270          *
00271          *
00272          *
00273          *
00274          *
00275          *
00276          *
00277          *
00278          *
00279          *
00280          *
00281          *
00282          *
00283          *
00284          *
00285          *
00286          *
00287          *
00288          *
00289          *
00290          *
00291          *
00292          *
00293          *
00294          *
00295          *
00296          *
00297          *
00298          *
00299          *
00300          *
00301          *
00302          *
00303          *
00304          *
00305          *
00306          *
00307          *
00308          *
00309          *
00310          *
00311          *
00312          *
00313          *
00314          *
00315          *
00316          *
00317          *
00318          *
00319          *
00320          *
00321          *
00322          *
00323          *
00324          *
00325          *
00326          *
00327          *
00328          *
00329          *
00330          *
00331          *
00332          *
00333          *
00334          *
00335          *
00336          *
00337          *
00338          *
00339          *
00340          *
00341          *
00342          *
00343          *
00344          *
00345          *
00346          *
00347          *
00348          *
00349          *
00350          *
00351          *
00352          *
00353          *
00354          *
00355          *
00356          *
00357          *
00358          *
00359          *
00360          *
00361          *
00362          *
00363          *
00364          *
00365          *
00366          *
00367          *
00368          *
00369          *
00370          *
00371          *
00372          *
00373          *
00374          *
00375          *
00376          *
00377          *
00378          *
00379          *
00380          *
00381          *
00382          *
00383          *
00384          *
00385          *
00386          *
00387          *
00388          *
00389          *
00390          *
00391          *
00392          *
00393          *
00394          *
00395          *
00396          *
00397          *
00398          *
00399          *
00400          *
00401          *
00402          *
00403          *
00404          *
00405          *
00406          *
00407          *
00408          *
00409          *
00410          *
00411          *
00412          *
00413          *
00414          *
00415          *
00416          *
00417          *
00418          *
00419          *
00420          *
00421          *
00422          *
00423          *
00424          *
00425          *
00426          *
00427          *
00428          *
00429          *
00430          *
00431          *
00432          *
00433          *
00434          *
00435          *
00436          *
00437          *
00438          *
00439          *
00440          *
00441          *
00442          *
00443          *
00444          *
00445          *
00446          *
00447          *
00448          *
00449          *
00450          *
00451          *
00452          *
00453          *
00454          *
00455          *
00456          *
00457          *
00458          *
00459          *
00460          *
00461          *
00462          *
00463          *
00464          *
00465          *
00466          *
00467          *
00468          *
00469          *
00470          *
00471          *
00472          *
00473          *
00474          *
00475          *
00476          *
00477          *
00478          *
00479          *
00480          *
00481          *
00482          *
00483          *
00484          *
00485          *
00486          *
00487          *
00488          *
00489          *
00490          *
00491          *
00492          *
00493          *
00494          *
00495          *
00496          *
00497          *
00498          *
00499          *
00500          *
    
```

PAGE 002 CIMEN

```

00341P 001C 22 10 0030      BHI      ITC1
00342P 0020 4C              INCA
00343P 0021 01 2701 A      CNPA      N1
00344P 0024 22 03 0029      BHI      TSTC2
00345P 0026 7E 0004 P      FFC1     JMP      FINE
00346P 0029 06 2703 A      TSTC2    LDAA    NOTE
00347P 002C 01 01 A      CNPA     #1
00348P 002E 27 7C 002C      DEC     FFC1
00349P 0030 7C 0000 D      ITC1    LDX     ADRC
00350P 0033 AC 00 A      LDAA    0.X
00351P 0035 01 00 A      CNPA     #0
00352P 0037 27 10 0034      DEC     JUSQC
00353P 0039 7C 0000 D      LDX     ADRC
00354P 003C AC 01 A      LDAA    1.X
00355P 003E 00 0000 D      ADDB    REFC+1
00356P 0041 07 0000 D      STAA   REFC+1
00357P 0044 AC 00 A      LDAA    0.X
00358P 0046 09 0007 D      ADCA   REFC
00359P 0049 07 0007 D      STAA   REFC
00360
00361
00362
00363P 004C 7C 0007 D      LDX     REFC
00364P 004F 0C 0009 D      ITC2    CPX     CAPTC
00365P 0052 22 70 004F      BHI     ITC2
00366
00367
00368
00369P 0054 7C 0000 D      JUSQC   INC     DC
00370P 0057 7C 0000 D      LDAB    DC
00371P 005A C1 02 A      CMPC    #2
00372P 005C 22 11 0061      BHI     ITC3
00373P 005E 7C 0000 D      LDX     ADRC
00374P 0061 00              INX
00375P 0062 1F 0000 D      STX     ADRC
00376P 0065 7C 0000 D      LDX     ADRC
00377P 0068 00              INX
00378P 0069 00              INX
00379P 006A 7F 0000 D      STX     ADRC
00380P 006D 20 C1 0030      BRA     ITC1

```

N > N1 77

CIM(0) = 0 77

REFC (--- REFC+PCIM(0))

* PESEE CIMEN

REFC > CAPTC 77

* FIN PESEE

ADRC (--- ADRC+1

ADRC + 2

PAGE 003 CIMLN

```

00801
00802
00803
00804
00805
00806P 006F 06 2703 A      ITC3    LDAA    NOTE
00807P 0072 01 00 A      CNPA     #0
00808P 0074 2C 79 006F      DNL     ITC3
00809
00810
00811
00812P 007C 7C 0009 D      ITC4    LDX     CAPTC
00813P 0079 0C 0000 A      CPX     #0
00814P 007E 22 70 007C      BHI     ITC4
00815
00816
00817
00818P 007E 7C 2703 A      INC     NOTE
00819P 0001 7C 0000 P      JMP     INITC
00100P 0004 01          FINE    NOP
00101
00102          0000 P      END     CIMLN
TOTAL ERRORS 00000

```

PAGE 001 EAU

```

00001 *
00002 *   PARTIC VOL. D'EAU/MALAXAGE
00003 *
00004 *   NBR   EAU
00005 *   OPT   REL.PAGE=46
00006 *
00007 2700 A N   EQU  $2700   NB CYCLES FABR. A REALISER
00008 2701 A N1  EQU  $2701
00009 2702 A NOTE EQU  $2702   ELEM SYNCHRON EAU
00010 2703 A NOTE EQU  $2703   ELEM SYNCHRON CIMENT
00011 2704 A N01A EQU  $2704   ELEM SYNCHRON AGREGAT
00012 *
00013 0000 *   DSCT
00014 *
00015 0000 0001 A ATE  RUC  1   ACTIONNEUR TRAPE EAU
00016 0001 0001 A PCC  RUC  1   POSITIONNEUR COMPTEUR EAU
00017 0002 0001 A ACD  RUC  1   ACTION COULEE DETON
00018 0003 0001 A FRE  RUC  1   FIN REMPLISSAGE D'EAU
00019 0004 0001 A ITH  RUC  1   INDICATEUR TEMPS MALAX.
00020 0005 0001 A TH  RUC  1   TEMPS MALAXAGE
00021 0006 0001 A THJ  RUC  1
00022 0007 0001 A TCD  RUC  1   TEMPS COULEE DETON
00023 0008 0001 A TSP  RUC  1   INDICATEUR SYSTEME PRET
00024 0009 0002 A RETE RUC  2
00025 0000 0001 A DC1  RUC  1   AUTORISATION COULEE DETON
00026 0000 0001 A CON  RUC  1   COULEE EAU
00027 *
00028 0000 *   PSET
00029 *
00030 0000 P EAU  EQU  *
00031 *
00032 0000 00 01  A INTE  LDAA  N1
00033 *
00034 *
00035 *
00036 *   CONTROLE NB CYCLES DE FABRICATION
00037 *
00038 0000 0002 00 2700 A   LDAA  N
00039 0005 01 2701 A   CHFA  N1   N 1 N1 77
00040 0000 22 14 001E   DRT  EAU1
    
```

PAGE 002 EAU

```

00041 0000 7E 0001 P   JMP  FINE
00042 *
00043 *
00044 *
00045 *   CONTROLE EAU
00046 *
00047 0000 7C 2702 A   LDAD  NOTE
00048 0010 01 00  A   CHPD  N0   NOTE = 0 77
00049 0012 20 10 0001 A   DRE  ITC1   SI NOTE =1 ALORS CONTROLE
00050 0014 7C 0000 D   LDAD  CON
00051 0017 01 01  A   CHPC  N1   CON = 1 77
00052 0019 20 07 001E A   DRE  EAU1
00053 0010 00 007E P   DCR  ITC2
00054 *
00055 *
00056 *   PHASE REMPLISSAGE D'EAU
00057 *
00058 0010 7E 0001 D EAU1 CLR  PCC   PCC 1-- 0
00059 0021 7E 0000 D CLR  ATC   ATC 1-- 0
00060 *
00061 *   ITEREN JUSQU'A FIN REMPLISSAGE D'EAU = 1
00062 *
00063 0024 7C 0003 D ITC0 LDAD  FRE
00064 0027 01 01  A   CHPD  N1
00065 0029 20 09 0024 A   DRE  ITC0
00066 0020 07 0000 D   STAA  ATC   ATC 1-- 1
00067 0020 07 2702 A   STAA  NOTE   NOTE 1-- 1
00068 *
    
```


PAGE 000 INDU

```

00007P 0000 04 04 A ANDA #104
00008P 0001 27 05 0070 DEQ CASE
00009P 0071 FF 0007 D STX CAPTA CAS ADREC
00010P 0074 20 03 0070 BRA TSTIG
00011P 0076 FF 0009 D CASC STX CAPIC CAS CIMENT
00012P 0079 DC 0000 D TSTIG LDAA PIA2+1
00013P 007C 04 02 A ANDA #102
00014P 007E 27 05 0005 BLW TSTI7
00015P 0000 7C 000F D INC FRL
00016P 0003 20 03 0000 BRA TSTI0
00017P 0005 7F 000F D TSTI7 CLR FRL
00018P 0000 0C 000C D TSTI0 LDAA PIA2+1
00019P 0000 04 01 A ANDA #101
00020P 0000 0C 0010 D LDAA BLT
00101
00102 0000 P END INDU
TOTAL ERRORS 00000

```

PAGE 001 SIMUL

```

*****
*
* SIMULATION TEMPS WILL BE CAPTURED
*
NAM      SIMUL
OPT      REL.PAGE-40
*
00007    2705 A PIA1 EQU 12705
00008    2707 A PIA1 EQU 12707
00010    0000          DSCI
00012    0000 000A A A1  RND 10
00013    0000 0002 A D1  RND 2
00014    0000 0002 A Y1  RND 2
00015    0000 0001 A X1  RND 1
00016    0001 0002 A PIA1 RND 2
00017    0011 0002 A PIA2 RND 2
00018    0013 0001 A I   RND 1
00019    0014 0002 A Y11  RND 2
00020    0016 0002 A SAVX  RND 2
00021    0010 0001 A SELEC RND 1
00022    0019 0002 A D1A  RND 2
00023    0010 0002 A D1C  RND 2
00024    0000          PSET
00025    0000 P SIMUL EQU *
00026P 0000 7F 0019 D CLR D1A
00027P 0003 7F 0010 D CLR D1C
00028P 000C 7F 0013 D SIMUL1 CLR I
00029
*
* LECTURE PIA
*
00030P 0000 FF 2705 A LDX PIA1
00031P 0000 FF 000F D STX PIA1
00032P 000F FC 2707 A LDX PIA1+1
00033P 0012 FF 0011 D STX PIA2
00034
*
00035P 0015 CC 0000 D LDX #A1
00036P 0010 FF 0010 D STX SAVX SAVX (--- ADR A1)
00037P 0010 00 SCC
00038P 0010 79 0010 D ROL PIA1+1
00039P 001F 00 00A1 P JSR SUBP
00040P 0022 DC 0013 D ITEC1 LDAA I A (--- I)

```


PAGE 004 SIMUL

```

00100 *
00101 * RUT1 : CALCUL ET AFFICHAGE DE POIDS
00102 *
00103 *
00104 * CALCUL DELTA T = 100 US
00105 *
00106P 0003 00 20 A RUT1 LDAA #32
00107P 0005 4A DECA
00108P 000C 26 FC 0003 DNL RUT1
00109 *
00110 * CALCUL Y - AX + D
00111 *
00112P 000C FC 0010 D LDX SAVX
00113P 000B AC 00 A LDAA 0,X
00114P 0000 00 0000 D ADCA 01+1
00115P 0000 07 0000 D STAA Y1+1
00116P 0003 09 000A D ADCA 01
00117P 000C 07 000C D STAA Y1
00118 *
00119 * DEVALIDER POIDS
00120 *
00121P 0009 00 0012 D LDAA PIA2+1
00122P 000C 04 F7 A ANDA #4F7
00123P 000E 07 0012 D STAA PIA2+1
00124 *
00125 * AFFICHAGE POIDS
00126 *
00127P 0001 FC 0000 D LDX Y1 X ( -- Y1
00128P 0004 FF 0014 D STX Y11
00129P 0007 00 04 A LDAA #4
00130P 0009 70 0015 D RUT12 ACL Y11+1
00131P 000E 79 0014 D RDL Y11
00132P 000C 4A DECA
00133P 0000 20 FC 0003 DNL RUT12
00134P 000C 00 0014 D LDAA Y11
00135P 0005 07 0011 D STAA PIA2
00136P 000B 00 0012 D LDAA PIA2+1
00137P 000B 04 0F A ANDA #40F
00138P 0000 0A 0015 D ORAA Y11+1
00139P 0000 07 0012 D STAA PIA2+1
00140 *
00141 * SELECTION (AGREGAT / CIMENT)
00142 *

```

PAGE 005 SIMUL

```

00143P 0003 00 0012 D LDAA PIA2+1
00144P 000C FC 0010 D LDAB 1
00145P 0009 01 07 A CNPB #7
00146P 0000 02 0A 0107 DHI RUT13
00147P 0000 0A 04 A ORAA #004 -- CAS AGRÉGAT : SELECTION
00148P 0007 07 0012 D STAA PIA2+1
00149P 0100 FF 0010 D STX 01A
00150P 0105 20 00 010F DRA RUT14
00151P 0107 04 FC A RUT13 ANDA #4FC -- CAS CIMENT : SELECTION
00152P 0109 07 0012 D STAA PIA2+1
00153P 010C FF 0010 D STX 01C
00154 *
00155 * VALIDATION A CONTIENT PIA2+1
00156 *
00157P 0101 0A 00 A RUT14 ORAA #100
00158P 0111 07 0012 D STAA PIA2+1
00159 *
00160 * FIN RUT1
00161 *
00162P 0114 00 RIS
00163 *
00164P 0110 01 FSIMUL NOP
00165 0000 P END SIMUL
TOTAL ERRORS 00000

```

Annexe - B -

Rep	Nb	Désignation	Caractéristiques
MC 6800	5	C P U	1 μ S - 1 M Hz
INTEL 2114 - 2	8K Octets	R A M	200 ns - 1K X 4 bits
INTEL 2708	8K Octets	Rep R O M	450 ns - 1K Octets
MC 6850	4	A C I A	450 ns
MC 6821	6	P I A	450 ns
SFF 96875	2	\emptyset	\leq 8 M Hz
SN74LS244	32		
SN74LS245	12		
SN74150	4	16 - 1	
SN74151	4	8 - 1	
SN74154	6	4 - 16	
SN74S138	2	3 - 8	

Nomenclature des Principaux Circuits Intégrés Utilisés

BIBLIOGRAPHIE

-0-0-

- [1] G. MAZARE : "Structures multi-microprocesseurs, Problème de parallélisme, Définition et évaluation d'un système particulier", Thèse d'Etat, Grenoble, juin 1978.
- [2] H. BELLM, A. SAUER : "Methods of data exchange between computers", Proceeding of Euromicro 77 Symposium, North-Holland Publishing company, pp.16-22.
- [3] FTC6 : Proceedings, Paris, juin 1976.
- [4] FTC7 : Proceedings, Los Angeles, juin 1977.
- [5] FTC8 : Proceedings, Toulouse, juin 1978.
- [6] J.C. LAPRIE : "Prévision de la sûreté de fonctionnement et architecture de structures numériques temps réel réparables", Thèse d'Etat, Toulouse, juin 1975.
- [7] C. BELLON : "Etude de la dégradation Progressive dans les Systèmes Répartis", Thèse 3e cycle, INPG, Grenoble, Sept. 1977.
- [8] Z. AVIZIENIS, B. PARHAMI : "A fault tolerant parallel computer system for signal processing", Proc. of fault tolerant computing symposium, Urbana, pp.2-9, 1974.
- [9] G. GERMAIN, F. BROWAEYS, C. MERAUD : "COPRA, un multiprocesseur à haute sûreté de fonctionnement", Doc. interne EMD-SAGEM.
- [10] J.S. MILLER : "Design features of an aerospace multiprocessor", Int. Workshop on computers Architecture, Grenoble, 1973.
- [11] M.A. FISCHLER, O. FIRSCHEIM : "A fault tolerant multiprocessor architecture for real-time control applications". Proc. of 1st Annual Symp. on Computer architecture, pp.151-160, Floride 1973.
- [12] W.A. WULF, C.G. BELL : "Cmmp- a multi-mini-Processor", Proc. AFIPS, FJCC, Vol.41, pp.765-777, California 1972.
- [13] WENSLEY : "SIFT Software Implemented Fault-Tolerance", AFIPS, FJCC, Vol.41, pp.243-253, California 1972.
- [14] J.D. NICLOUD : "MUBUS - Standard", Microscope , Vol.1, n°2, pp.7-8, March 1976.

- [15] M. FAIMAN : "MUMS - A Reconfigurable microprocessor architecture",
Computer, Vol.10, n°1, pp.11-16, Jan. 1977.
- [16] D. DAVIES : "Reliable Synchronisation of Redundant Systems", Digital
Systems Laboratory, Stanford, California, October 1977.
- [17] L.C. WIDDOES : "The MINERVA multi-microprocessor", Proceedings of
the 3rd Annual Symposium on computer architecture,
Miami, Jan. 1976.
- [18] H. ROTH LISBERGER : "A standard bus for multiprocessor architecture",
Euromicro Symposium, Amsterdam, pp.23-33, October 1977.
- [19] M. MOALLA, J. SIFAKIS, N. ZACHARIADES : "MAS : Un outil d'aide à la
description et à la conception des automatismes logiques",
ENSIMAG, Grenoble 1977.
- [20] H.M. HABANNAKEH : "Conception et Réalisation de l'Équipement Auto-
matique d'une centrale à béton", thèse de 3e cycle,
Montpellier, juin 1977.
- [21] A. OSBORNE : "An Introduction to Microcomputers", Sybex 1976.
- [22] J.B. PEATMAN : "Microcomputer-based design", Mc.Graw-Hill, 1977.
- [23] J. CLAVIER, M. NIQUIL, G. COFFINET, F. BEHR : "Théorie et Technique
de la transmission des données; Systèmes télé-informatiques",
Masson, 1977.
- [24] CROCUS : "Système d'exploitation des ordinateurs", DUNOD, 1976.
- [25] H. LILEN : "Introduction à la Micro-Informatique du Microprocesseur
au Micro-Ordinateur", Radio, 1977.
- [26] MOTOROLA : "M6800 Microprocessor Applications Manuel", 1975.
- [27] SESCOSEM : "Microprocesseur M6800 et circuits associés", Thomson, 1978.
- [28] TEXAS INSTRUMENTS : "Data Book", 1977.
- [29] INTEL : "Data Book", 1979.

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU le rapport de présentation de :

- Madame G. SAUCIER, Maître de Conférences à l'Institut
National Polytechnique de GRENOBLE

Monsieur Hussein HABANNAKEH-MIDANI

est autorisé à présenter une thèse en soutenance pour l'obtention
du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Génie Informatique"

Grenoble, le 11 Mai 1979

Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique

