Génération automatique de partie(s) contrôle(s) de microprocesseurs sous forme de PLA spécialisés

Henry Derantonian

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR–INGENIEUR
«Informatique»

par

DERANTONIAN Henry

GENERATION AUTOMATIQUE DE PARTIE(S) CONTROLE(S) DE MICROPROCESSEURS SOUS FORME DE PLA SPECIALISES.

Thèse soutenue le 6 juillet 1984 devant la commission d'examen.

F. ANCEAU	Président
L. BOLLIET	Examinateurs
B. COURTOIS	
J.L. LARDY	
J.P. MOREAU	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Je tiens à remercier:

Monsieur le Professeur F.ANCEAU, responsable de l'équipe de recherche en architecture des ordinateurs, pour l'honneur qu'il me fait d'accepter de présider le jury et pour m'avoir aidé à mener à bien le compilateur de PLA,

Monsieur L.BOLLIET, Professeur à l'universiré de grenoble III, qui m'a honoré en acceptant de participer au jury,

Monsieur B.COURTOIS, responsable de l'équipe de test des circuits VLSI, pour avoir voulu siéger au jury et m'avoir permis d'approfondir ma connaissance sur le microprocesseur MC68000,

Monsieur L.LARDY, responsable de l'équipe de conception du CNET/Meylan, qui a bien voulu juger mon travail et faire partie du jury,

Monsieur J.P.MOREAU, délégué auprès du directeur technique d'EFCIS, pour l'attention toute particulière qu'il a accordée à ce travail et qui a accepté d'en être le rapporteur,

Tous les membres de l'équipe de recherche en architecture d'ordinateurs et en particulier, Monsieur J.P.SCHOELLKOPF, pour les nombreuses discussions sur l'aspect logiciel du compilateur et Mademoiselle M.OBREBSKA, pour ses conseils efficaces

Tous mes collègues, Pierre, Daniel, pour leur aide généreuse qui a contribué à parfaire la rédaction de cette thèse,

Les membres du service de tirage de l'IMAG, pour l'excellente qualité de leur travail.

à mes parents,

à mon épouse,

et à mes enfants

CHAPITRE I

Introduction

page

CHAPITRE II

DEFINITION DE L'ARCHITECTURE INTERNE

DES MICRO-PROCESSEURS

CHAPITRE III

PRSENTATION DES METHODES DE CONCEPTION

D'UNE PARTIE CONTROLE A

L'AIDE DE PLA

page

CHAPITRE IV

DISCRIPTION DE LA LANGAGE INTERMEDIAIRE

CHAPITRE V

STRUCTURE DE DONNEE DU COMPILATEUR

page

CHAPITRE VI

REALISATION MATERIELLE DES ELEMENTS DU LANGAGE INTERMEDIAIRE

page

CAHPITRE VII

APPLICATION DU COMPILATEUR DES PLA AU CAS DU MICROPROCESSEUR MC6800

CHAPITRE I

INTRODUCTION

Introduction

Le developpement rapide des microprocesseurs réalisant les fonctions avec une compléxité accrue et une vitesse de calcul croissante nécessite l'utilisation d'une C.A.O. (conception assistée par ordinateur) adaptée; en particulier pour la conception et l'implantation des structures, relatives aux parties contrôles.

plusieurs contraintes nous ont amené à concevoir un outil de génération de parites contrôles, ces dernières interviennent:

-Lorsqu'on a besoin d'un circuit dont le coût de conception est trés faible, et qui nécessite une conception rapide.

-Pour l'augmentation de la sécurité de conception, c'est à dire la diminution des erreurs de conception.

-La testabilité qui devient jour après jour difficile à atteindre quand la complexité augmente.

-Les contraintes geographiques dans l'organisation globale d'un processeur. Celles-ci proviennent du coût des lignes d'interconnexions sur la pastille. Une telle contrainte nécessite une organisation décentralisée de l'unité de contrôle.

Le premier but du formalisme PAMELA est la génération automatique d'une parite contrôle sous forme de PLA (Program Logic Array); à partir de la structure algorithmique de son automate. Plusieurs réalisations pour une même partie contrôle à l'aide de deux PLA et plusieurs types de compteurs sont envisagées.

Un tel outil permet le passage automatique des spécifications (l'algorithme) à la réalisation physique du circuit. Une réalisation qui permet de conçevoir une partie contrôle de plus en plus réduite et régulière.

Ce programme génére deux ou plusieurs PLA sous forme de matrices :

-Un PLA d'enchainement associé à la mise en action d'un

compteur de micro-programme auto-incrémentable, ce qui assure l'enchainement des differents états.

-Un ou plusieurs PLA de génération des commandes. Ces PLA activent les divers organes de la parite opérative. Les matrices des décodeurs seront générées en cas de paramétrisation des actions.

La description algorithmique permet au concepteur de réaliser les deux types d'automates: de Moore et de Mealy.

Les matrices, après une optimisation topologique par le systeme "PAOLA" serviront à générer les masques des PLA. A titre d'exemple la partie contôle du microprocesseur MC6800 est générée par le systéme PAMELA, prouvant la souplesse de cet outil.

Ce systéme est écrit en PASCAL U.C.S.D sur un micr-ordinateur de type PASCALINE.

Dans la partie contrôle, faite essentiellement de PLA il existe une partie faite de circuit en logique aléatoire. Le contrôle des signaux: reset, halt, codes invalides et les interruptions dans le cas du microprocesseur MC68000 est approfondie dans les annexes 4 et 5 montrant une telle logique aléatoire.

CHAPITRE 2

DEFINITION DE L'ARCHITECTURE INTERNE

DES MICRO-PROCESSEURS

1-Conception structurée des machines d'états finis

Une méthodologie, pour la conception des circuits intégrés digitaux, nécessite une démarche descendante et structurée.

Une approche pour réduire le temps et le coût de conception est l'utilisation des structures programmables: les PLA (Program Logic Arrays) et les ROM (Read Only Memories). Ces dérnières présentent plusieurs avantages, tels que:

-régularité de la structure.
-possibilité d'extension pour la plupart des structures programmables.
-structures autotestables.

Un domaine important de l'utilisation des structures programmables en conception des systèmes digitaux, est l'implantation des machines d'états finis en PLA.

2-Les niveaux de conception des machines d'états finis

Dans la conception materielle des systèmes intégrés digitaux, nous distingons plusieurs niveaux méthodologiques. Chacun de ces niveaux a besoin d'une méthode spécifique de modélisation et eventuellement de ses propres notations graphiques et symboliques. A Partir de la séquence des niveaux de conception, nous distinguons les niveaux suivants:

-Implantation I
-Bloc B
-Transfer de registre RT (ou RTL)
-Logique L
-Circuit C
-Géométrie G

Chaque niveau a ses primitives specifiques. Les primitives d'un niveau particulier peuvent être implémentées par l'utilisation des primitives du niveau inferieur. Les primitives du niveau B sont les manipulation de données, les mémoires, les contrôleurs,...etc. Les

représentations typiques de niveau B, sous forme graphique sont les blocs diagrammes en utilisant des boites. Les primitives du niveau RT sont: les registres, fonctions, multiplexeurs, démultiplexeurs, les chemins de données,....etc. Par exemple un manipulateur de donnée comme l'unité arithmétique et logique d'un ordinateur peut être implémenté par l'utilisation des primitives d'un niveau RT. La representation symbolique du RT, utilise les langages de description comme langage CASSANDRE, IRENE, DDL, ISP,...etc.

Les primitives du niveau logique sont les portes et flip-flops. Par conséquent les primitives du niveau B sont: les transistors, resistances, capacités. Les "sticks" diagrammes sont les representations du niveau T. Le niveau C permet une representation des connexions. Enfin le niveau géométrie, illustre la géométrie et le placement de differentes matérielles et leurs surfaces sur la pastille.

3-Le niveau de transfer de registre

Le niveau RT est un niveau trés utile entre les niveaux B et L. Un circuit intégré, représenté comme une boite noire au niveau B, peut etre constitué de milliers de portes. Par conséquent la description de ce niveau peut être très imprécise. D'autre part la description du niveau L peut etre trés complexe, pour qu'il soit comprehensible. Donc un niveau intermédiaire entre B et L avec ses propres notations pour un niveau de conception intermédiaire est indispensable.

4-Langage de RTL, IRENE

Ce langage a été conçu pour servir de support à une méthodologie de conception de circuits intégrés CAPRI presenté par F.Anceau ref.[ANC-83]. Des formes de descriptions comportementale, structurelle et mixte sont définies dans réf.[MAR-83].

Le projet de compilateur de silicium CAPRI utilise le langage IRENE. Il génère le comportement des deux parties d'un processeur

la partie opérative et la partie contrôle.

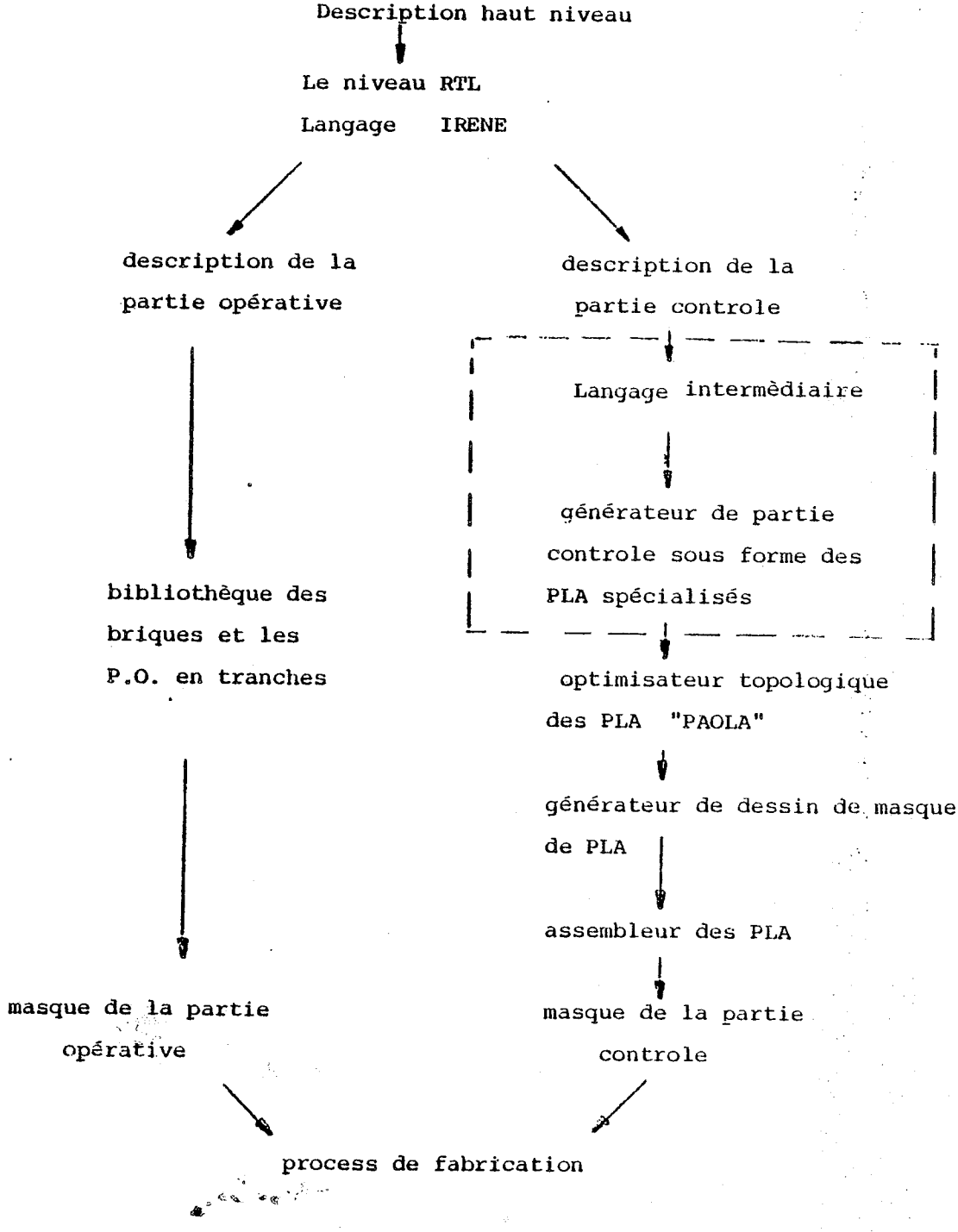


Figure 2-1 Décomposition du projet de compilateur de silicun en differentes couches de conception automatique

****

5-La notion de partie contrôle et partie opérative

Une machine séquentielle peut être décomposée en deux parties:
- Une partie opérative qui effectue les manipulations de données et les operations.
- Une partie contôle qui commande la partie opérative pour une séquence d'operation. L'unité d'horlogerie du système synchronise ces séquences.

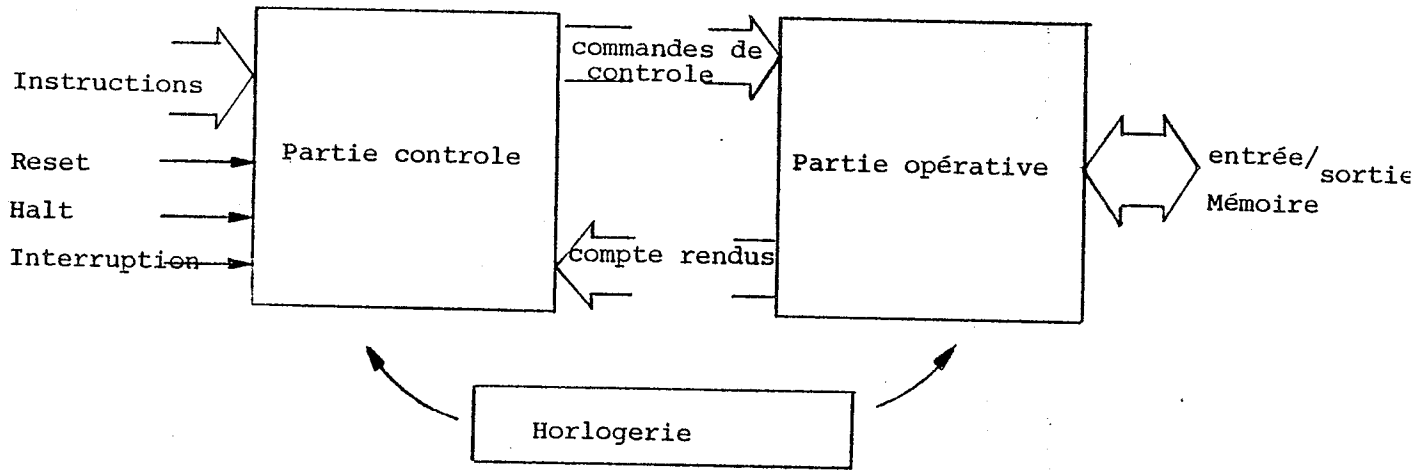


Figure 2-2 Décomposition d'une machine d'état fini

****

La partie contôle commande la partie opérative par l'intermédiaire des lignes de contrôle. La partie opérative envoie son état par l'ensemble des lignes de compte rendus. Une telle décomposition peut être exraite de la specification algorithmique. Les définitions suivantes peuvent être extraites:
- La machine virtuelle, définie à partir de l'ensemble d'instructions
- Le contenu de la sous partie contrôle de la machine physique (Exemple les PLA, ROM's,..etc.) sont déduits à partir des niveaux intermédiaires.
- Le materiel (les deux parties) est défini à partir du dernier

niveau d'interpretation.

6-Spécification algoriithmique

Le comportement d'un ordinateur comme toutes les machines sequencielles peut etre decrit par un algorithme. Les primitives du langages algorithmique utilisé pour décrire ces specifications peuvent faciliter son intérprétation en matériel. La facilité d'une telle interprétation dépend de l'adaptation des descriptions comportementales à la réalisation materielle du système.

7-Les niveaux d'interprétation

La transformation d'une spécification algorithmique en machine physique peut être réalisée en plusieurs étapes. Chacune de ces étapes est considéré comme un niveau d'interpretation où l'execution de chaque instruction du niveau précédent en plusieurs étapes intermédiaires dépend de l'exécution des instructions de ce niveau. Donc la description comportementale d'un processeur peut etre vue à travers les niveaux d'interprétation. La figure 2-3 presente un tel classement des niveaux.

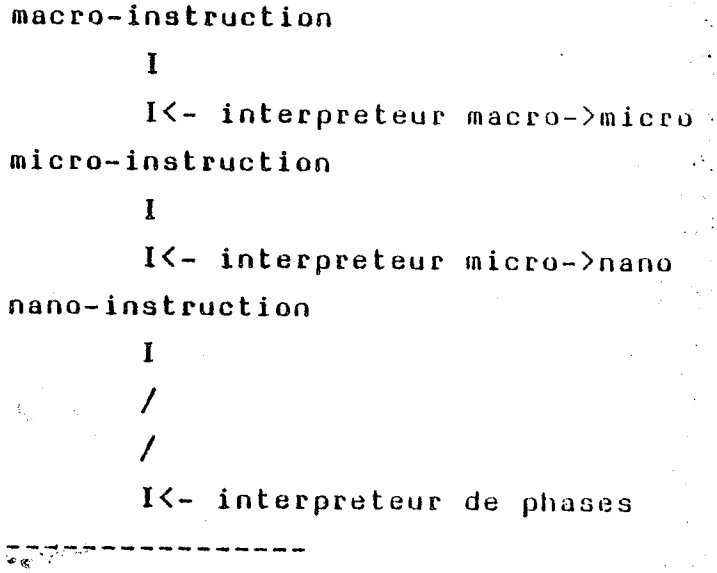


Figure 2-3- Niveaux d'interprétations

****

8-L'algorithme d'interpretation des instructions

Les chemins logiques exités dans la partie contrôle pour interpréter les macro-instructions sont presentés par le schéma de la figure 2-4. Les états de l'algorithme sont représentés par des rectangles. Les tansitions d'un état à l'autre correspondent aux flèches. La condition de transition est définie par la structure de chaque état, représentée par la partie basse de chaque réctangle. L'identificateur de chaque état et sa condition de transition sont marqués dans chaque réctangle. Les actions réalisées par chaque état correspondent aux flèches vers l'exterieur.

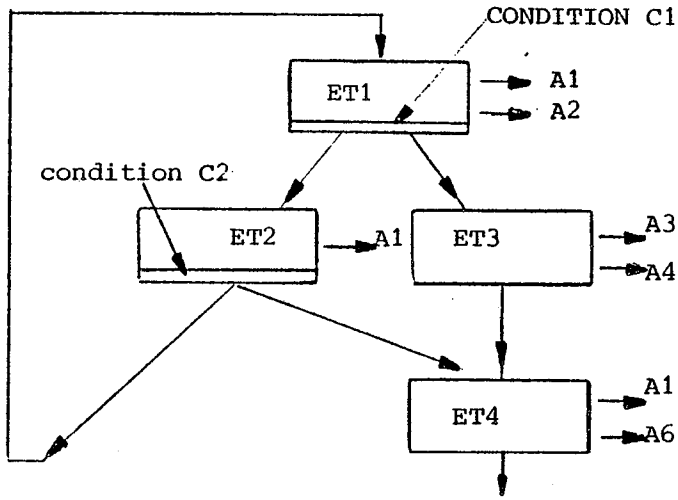


Figure 2-4 Une partie d'un algorithme d'interpretation

****

Les états sans condition de transition sont présentés par les états ET3 et ET4 sur la figure 2-3.

Les descriptions détaillées des algorithmes, répondant aux éxigences des réalisations matérielles, nécessitent les désignations suivantes:

1- définition des actions
2- définition des conditions
3- définition des états

Les éléments sont définis par les définitions primitives d'un langage intérmédiaire (presenté dans le 6-ème chapitre).

Le chapitre suivant decrit l'architecture de parties contrôles générées automatiquement par le système PAMELA.

CHAPITRE 3 .

PRESENTATION DES METHODES DE CONCEPTION

D'UNE PARTIE CONTROLE A

L'AIDE DE PLA

1-Introduction:

Plusieurs types d'architectures existent pour réaliser une partie contrôle. Les solutions les plus optimisées utilisent des PLA spécialisés. Ce sont celle qui sont visées par le système PAMELA. Les paragraphes qui suivent, détaillent ces architectures. Pour mieux aborder les structures des PLA utilisés, la solution à PLA unique est tout d'abord presentée, ensuite sont presentées d'autres approches.

2-L'unité de contrôle réalisée avec un PLA

Une telle unité de contrôle est considérée comme un automate implémenté à l'aide d'un PLA. La matrice ET décode l'instruction à executer, l'état de la partie opérative et l'état de l'algorithme de contrôle. Ce dernièr provient du registre d'état de l'automate. La matrice OU génère l'état futur qui sera chargé dans ce registre d'état et les commandes pour la partie operative. Cette technique facilite l'implantation de la partie contrôle mais la surface du PLA devient trés importante, quand l'algorithme d'interprétation des états devient assez complexe. L'éxemple d'un PLA exécutant un algorithme hypothétique est presenté pour clarifier les notions utilisées dans le langage intermédiaire.

3-Exemple d'un séquenceur mono-PLA

Figure 3-1 presente un tel PLA. Les commandes C1,C2 sont les conditions. Celles-ci sont activées par l'exterieur. Les valeurs véhiculées par ces bits, correspondent aux conditions autorisant les transitions des états dans l'algorithme d'interprétation. Deux entrées; directes et complémentées pour chaque bit de ces conditions, favorisent la détection de tous les produits de ces bits. Le mode de détection des valeurs est basé sur l'implantation des tansistors entre les lignes de conditions et les colonnes du

PLA-ET.

Les entrées E1 et E2 correspondent aux sorties S1 et S2 une periode d'horloge aprés. Cette intervalle est échantilloné entre deux instants T1 et T2. Les valeurs de ces bits sont memorisées dans le registre d'état. Les valeurs des bits E1 et E2, comme les conditions, sont détectées par les monômes qui remplacent les états de l'algorithme d'interprétation. Il est évident que le ET logique des valeurs des conditions et des états dans le PLA ET active un monôme qui représente l'état courant de l'algorithme. Par conséquet l'état futur est généré par les deux sorties S1 et S2 du PLA-OU.

Les actions de l'algorithme d'interprétation sont prises sur les sorties du PLA OU. Les actions An [n=1:6], de figure 3-1 sont les sorties envoyées vers la partie opérative.

Les tables de verité de la figure 3-2, présentent les profils des conditions et des états qui provoquent l'activation des monômes du PLA de la figure 3-1. Les monômes sont presentés par des lignes verticales. Les cases correspondantes qui activent un monôme [MA->MF] sont reliées indiquant le ET logique entre les conditions et les états.

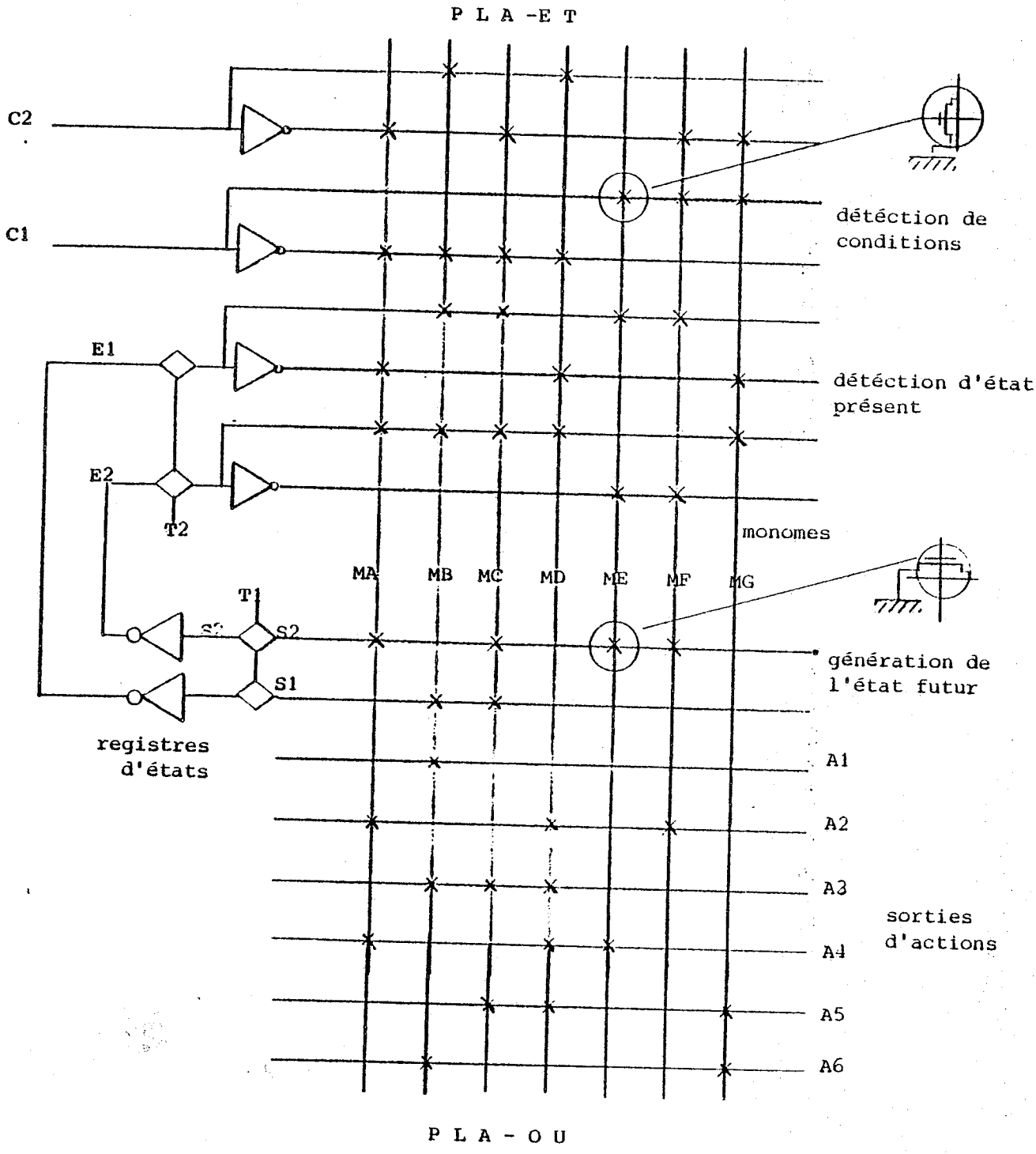


Figure 3-1-structure d'un PLA de séquenceur

****

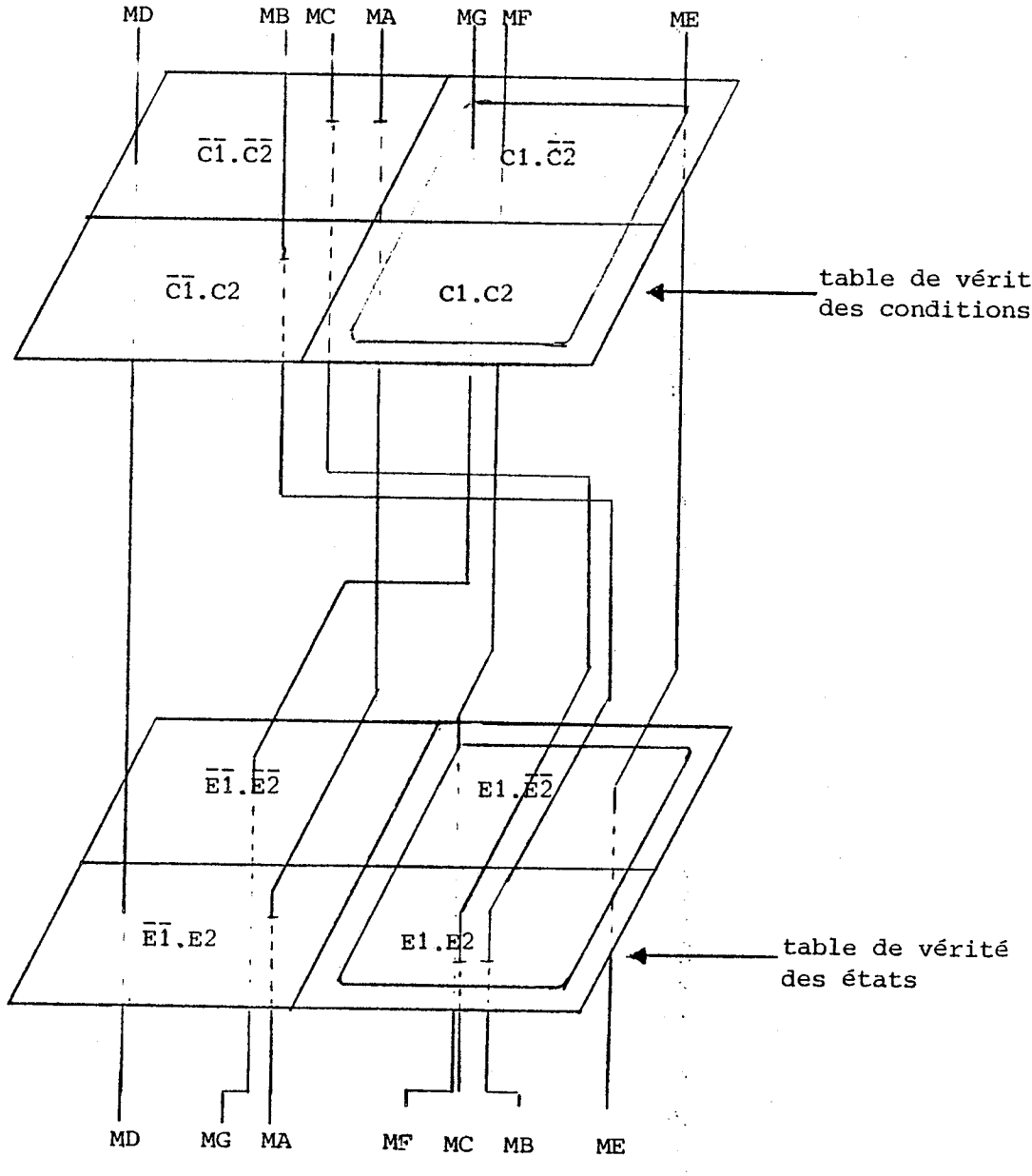


Figure 3-2-Présentation de l'activation des monômes par les profils d'entrées du PLA-ET

****

4-PLA-OU

L'activation de chaque monôme par le PLA-ET active les sorties du PLA-OU. Deux types de sorties peuvent être distinguées, les sorties Sn rebouclées dans le PLA-ET et les sorties An.

5-Etats futurs

L'état suivant est déterminé par l'implantation des transistors au croisement des lignes S1 et S2 et d'un monôme. Pour n-sorties 2^n profils peuvent être obtenue pour chaque monôme. La table de vérité pour deux sorties S1 et S2 du PLA de figure 3-1 est presentée par la figure 3-3.

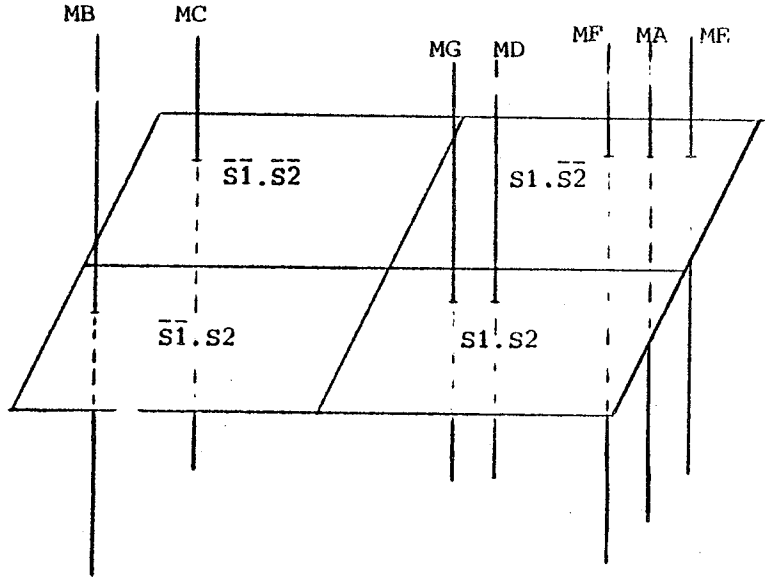


Figure 3-3- Les états futurs

****

6-Optimisation de surface d'un PLA unique

La réalisation d'une parite contrôle uni-PLA est facile mais quand la complexité de l'algorithme augmente la surface du PLA devient trop grande. On peut réduire la surface d'un PLA de séquencement par son découpage pour les tâches de séquencement et de génération des commandes. Par la suite nous allons étudier les optimisation à considèrer pour ce découpage.

7-PLA de séquencement

Le séquencement est assuré par le PLA ET et une partie du PLA OU qui génère l'état futur. Le mécanisme de séqunCement est simple, l'état présent est détecté par le PLA-ET. Un des monomes correspondant à cet état est activé. L'activation est conditionée par le profil des conditions appliquées au PLA.

L'état futur est généré par les lignes du PLA-OU. Celui-ci est stocké dans le registre d'état pour fournir l'état futur au PLA-ET. Différents successeurs peuvent être choisis selon les profils des conditions qui correspondent aux branchements dans l'algorithme d'interprétation.

8-PLA d'enchainement

On peut réduire la surface du PLA de séquencement en utilisant un compteur d'états non-conditionés de l'algorithme. Le PLA d'enchainement force le compteur avec le code du premier état d'une séquence. Dans ce cas là le compteur dénombre les états des séquences. En fin de séquence le code d'un état d'éclatement est généré, le PLA d'enchainement calcule son successeur. Le compteur est chargé de nouveau par ce code et effectue l'énumération de la nouvelle séquence. On peut donc noter que le séquencement est assuré par le couple compteur-PLA d'enchainement. Une telle architecture diminue la surface du PLA d'enchainement. Le taux de réduction dépend du nombre d'états simples de l'algorithme d'interprétation.

9-Les commandes de contrôle du compteur

Le compteur d'état fonctionne dans deux modes: chargement et incrémentation. En mode chargement il est forcé par le nouveau code d'état. En mode d'incrementation il énumère les états des séquences. Deux signaux correspondants doivent être généré par le

PLA d'enchainement. Il suffit d'avoir un monôme activé pour générer le signal "chargement". Par contre si l'état en cours n'est pas connu par ce PLA (aucun monôme activé) le compteur est mis en mode d'incrémentation. La figure 3-4 présente une telle configuration.

10-Types des compteurs utilisés

Le compteur binaire est le premier type considéré par le système. L'utilisation d'un tel compteur nécessite le codage binaire et successif des états des séquences. Cette contrainte est automatiquement considérée par le système de compilation et le code des états correspond au comportement des compteurs binaires. Chaque bit de ce compteur est constitué d'une cellule Maitre-Esclave, occupant la surface de deux bascules. Une optimisation le plus poussé necessite l'élimination de la commande d'incrémentation du compteur et l'utilisation de registres à décalages.

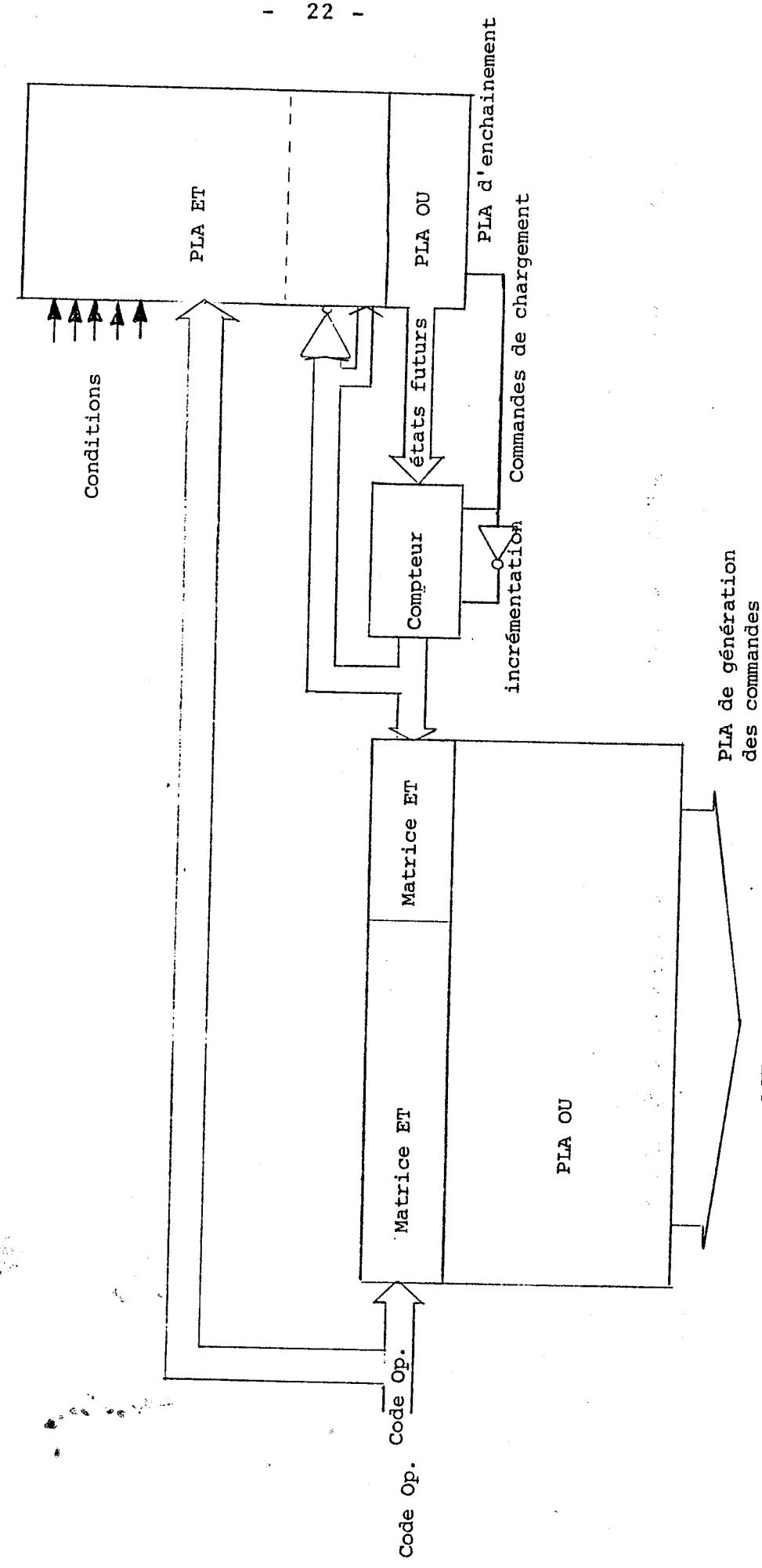


Figure 3-4 Architecture deux PLA d'un séquenceur

L'utilisation de compteurs polynomiaux répond à ce probléme et occupe la moitié de la surface d'un compteur binaire. Chaque fois qu'un compteur polynomyal est chargé avec un code il s'incremente à chaque coup d'horloge. Le choix de cette architecture necessite un codage particulier des états, qui répond au comportement des compteurs polyomiaux. Le systéme suppose une procédure de regroupement des séquences pour adapter les codages des états au type du compteur utilisé.

II-Structure d'un compteur polynomyal

Un tel compteur est constitué d'un registre à décalages avec au moins deux prises reliées à un ou exclusif. La sortie de cette porte "S" est envoyé au premier bit du compteur. Ces compteurs énumèrent les séquences non primitives si les prises sont mal choisies. En éffet chaque compteur de n bits pêut compter une séquence de:

$2^n - 1$ états.

L'un des états qui corresponds à 0000....0 ou 111....1 est un état piège selon que la porte S est un et exclusif ou un ou exclusif. La figure 3-5 presente les compteurs polynomiaux constitués de plusieurs bits.

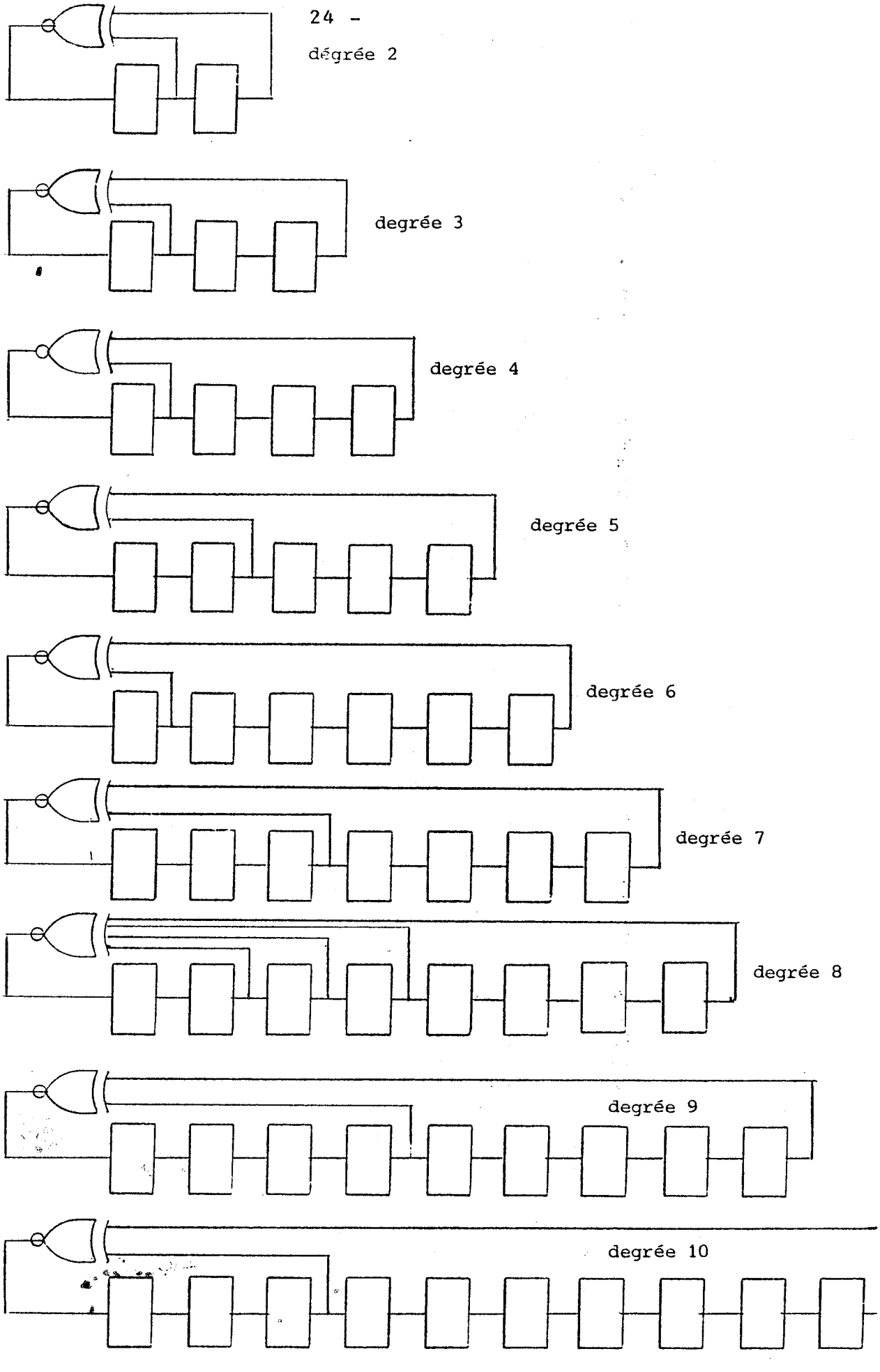


Figure 3-5· structure des compteurs polynomiaux

12-Codage par groupes et adresses

De manière à réduire la longueur des compteurs, la procédure de codage des états respecte un regroupement particulier des états. La séquence la plus longue est cherchée, et selon la longueur de cette séquence "LS", le programme calcule la capacité d'énumération du nombre minimum de bits qui couvre la LS. Une variable "n", énumérée à partir de 1, recherche le nombre de bits nécessaire. La procedure garde la variable n si:

- $\log_2 n >= LS$ pour les compteurs binaires

2- $\log_2 n-1 >= LS$ pour les compteurs polynomiaux

La capacité d'énumération basée sur la variable n est prise comme un compteur d'adresse. Une deuxième variable nommée pointeur de groupes est considérée pour condenser d'autres séquences de l'algorithme dans un tableau bi-dimensionnel. Par conséquent pour chaque séquence on trouve un groupe mais les états de cette séquence occupent des adresses differentes. La procédure de codage des états attribue à chaque état deux coordonnées: son groupe et son adresses dans le groupe. La séquenece la plus longue est dans le premier groupe (0) et les autres séquences occupent les autres zones libres restantes dans chaque groupe. Il est évident que la procédure tasse les séquences pour remplir tout les bits des adresses et utiliser le minimum de groupes.

En cas d'utilisation des compteurs binaires un codage incrémental est appliqué aux adresses des états de chaque séquence. En cas d'utilisation des compteurs polynomiaux ce codage est reconsidéré selon le comportement du comteur utilisé. Les specifications des séquences parcourues par un compteur polynomyal dependent du degré du polynome. Les tableaux auxilaires fournissent les codages nécessaires pour adapter le codage des adresses. Le tableau suivant presente le codage des états par deux variables n et g. Ces vaiables énumèrent le nombre de bits nécessaires pour

couvrir toutes les séquences de l'algorithme.

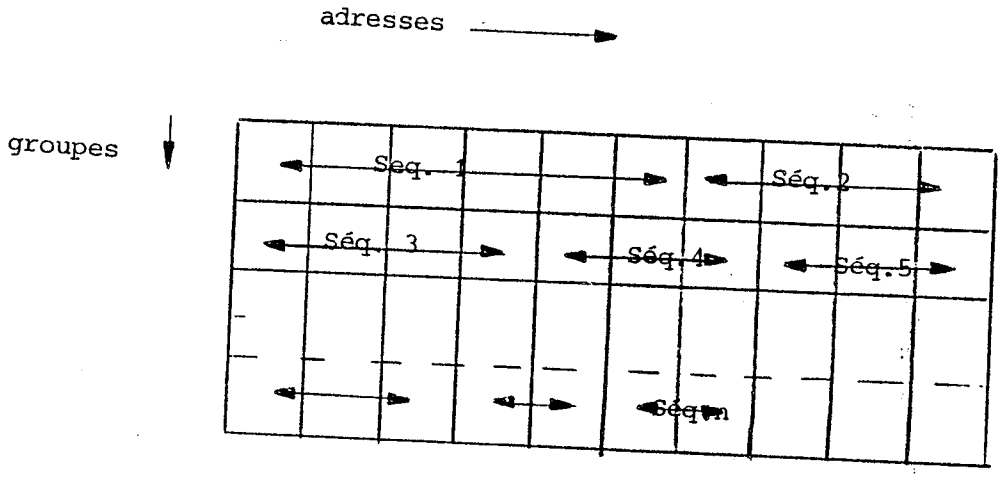


Figure 3-6-présentation de la répartition des séquences dans un tableau bi-dimensionnelle

****

Le codage par deux champs des états nécessite deux types de registres. L'un auto-incrémentable chargé d'énumerer les adresses et l'autre normale, permet d'enregistrer le groupe de chaque séquence.

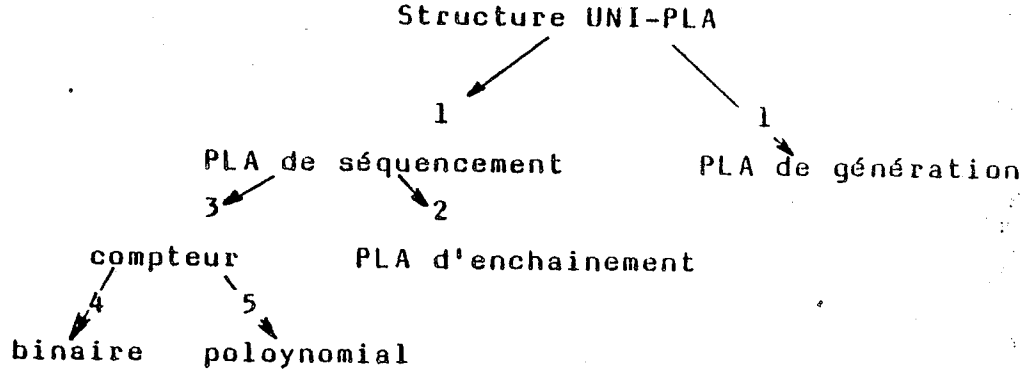
13-PLA de génération des commandes

L'aspect de génération des commandes est réalisé par un PLA. Celui-ci est constitué de deux matrices ET et OU. La matrice ET reçoit les bits de l'état courant et selon les différentes valeurs, active les monômes correspondants. La matrice OU, selon les monômes activés active les lignes de sorties des actions.

Le gain de surface est égal à la diminution du nombre de monômes représentant les transitions dans la solution uni-PLA au nombre d'états de l'algorithme. En effet il suffit d'avoir autant de monômes que d'états pour activer toutes les commandes

nécessaires.

14-Conclusion sur la stratégie d'optimisation d'un PLA



optimisation respecte:

1- Le séquencement est séparé de la génération. La surface consacrée à la génération est donc diminuée
2- Les états simples sont éliminés du PLA de séquencement
3- Les états simples sont énumérés par un compteur
4- Le codage par groupes et adresses ne nécessite qu'une partie des bits d'adresses pour être incrémenté
5- L'élimination de la commande d'incrémentation et la réduction de la surface du compteur binaire.

CHAPITRE IV

DESCRIPTION DU LANGAGE

INTERMEDIAIRE

La conception d'un automate nécessite un codage de tous les élements qui sont manipulés par son algorithme (phases, actions, conditions, états).

Si l'on étudie une partie contrôle, à partir de la sortie des commandes et des entrées des conditions, on peut définir une structure de données répondant à la structure des automates.
L'étude des commandes générées dans la partie "OU" du PLA; dans les premières étapes d'observation, nécessite la déclaration des signaux d'horlogerie (phases), qui définissent la manière, dont les actions sont synchronisées. La dècomposition de la partie opérative en registres de stockage temporaires et d'opérateurs, pour manipuler les données, classifie les phases en trois types suivants:

1- Phase de chargement

2- Phase de positionnement

3- Changement de phase (qui permet de changer la phase qui valide une action)

Les sorties de la partie contrôle, après être validées par les phases sont appelés les actions. Plusieurs types d'actions peuvent être distingués. La structure du PLA de génération des commandes ou PLA des paramêtres est basée sur les définitions des actions. Les actions suivantes sont décrites:

Actions

1- Actions binaires:

Les fils de commandes isolés, que ne peuvent prendre que deux valeurs '0' et '1', sont appelés des actions binaires.

2- Actions simples:

Les nappes de commandes regroupant plusieurs fils, capables de véhiculer des codes, sont appelés des actions simples.

3- Actions transcodées:

Ces actions correspondent au transcodage d'actions initiales.

4- Actions directes:

Elles correspondent à l'activation directe de commandes par des conditions

5- Actions conditionnelles:

Il s'agit d'actions dont l'execution est liée à l'occurence d'une condition . Leur utilisation correspond à une approche de Mealy de l'organisation de l'automate.

6- Actions paramétrés:

Ces actions favorisent la parametrisation des actions par les actions préalablement codées.

Les conditions d'entrée des PLA sont des conditions simples. Ces dernières correspondent à l'ensemble des fils d'entrée des PLA. On distingue plusieurs classes de conditions:

Conditions

1-Conditions simples:

Ces conditions correspondent à des nappes de fils véhiculant les codes des conditions.

2-Conditions complexes:

Ces conditions contient la combinaison (le produit) de plusieurs conditions simples.

3-Condition transcodée

Pour un transcodage de condition.

4-Conditions de propriétes

Ces conditions permettent aux actions, de choisir les sous-condition.

Etats

Les blocs d'états:

Ces blocs définissent l'enchainement de l'algorithme. Chaque état pointe sur l'ensemble des actions simples, binaires ou conditionnelles qu'il peut invoquer. Chaque bloc décrit les microcommandes à activer par la reconnaissance de cet état dans le PLA de génération des commandes.

La description fine des élements d'un automate.

On trouve dans la définition de chaque élément, un identificateur alphanumerique propre à l'utilisateur, et un nom interne présenté par un entier. Ces derniers sont compris entre 1 et 3000. Chaque élement, appartient à une classe qui sert à la génération automatique des noms internes à partir d'un identificateur.

I- PHASES
****

1-PHASE DE POSITIONNEMENT (CLASSE 1)

Definition:

<identificateur>
< 1 >
<nom interne>

Il s'agit de signaux d'horlogerie validant des positionnements.
Par exemple:
séléction de registres en lecture, commandes d'UAL ,....etc. La phase <99> est considérée comme toujours vraie. Elle sert, en fait, à indiquer qu'une action n'est pas validée par une phase.

2-PHASE DE CHARGEMENT (CLASSE 2)

Definition:

<identificateur>
< 2 >
<nom interne>

Il s'agit de signaux d'horlogerie validant des chargement des registres ou des écritures en mémoire.

3 CHANGEMENT DE PHASE (CLASSE 6)

Definition:

<identificateur>
< 6 >
<nom interne>
<phase de validation>
<action>

Permet d'utiliser une action avec une autre phase de validation que celle qui lui est normalement atachée.

II-ACTIONS

****

Les valeurs des bits des actions sont codées en utilisant un code ternaire.

- 1,0 pour les valeurs binaires.
- X lorsque la valeur de ce bit n'a pas d'importance.

1-ACTIONS BINAIRES (CLASSE 3)

Definition:

<identificateur>
< 3 >
<nom interne>
<phase>
<valeur inactive>

Ces actions correspondent à des micro-commandes codées par un bit. Par exemple une précharge de bus.

2-ACTIONS SIMPLES (CLASSE 4)

Definition:

<identificateur>
< 4 >
<nom interne>
<phase de validation>
<nombre de bits>
<nombre de valeurs>
<1-ère valeur (par defaut)>
<2-ème valeur>

<n-ème valeur>

Elle représentent des microcommandes codées, telles que celles utilisées par un UAL. Le schéma suivant montre une telle action AS-UAL.

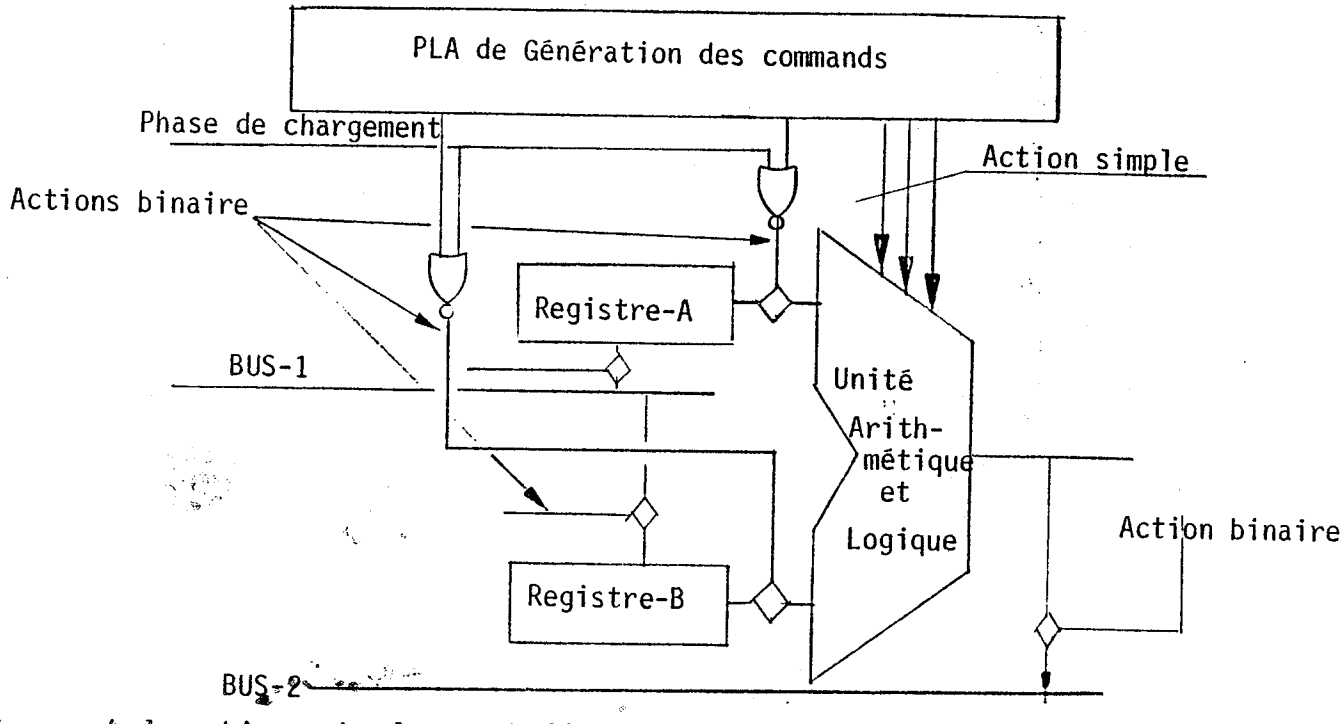


Figure 4-1 action simple contrôlant une Unité Arithmétique et Logique

****

3-ACTIONS TRANSCODEES (CLASSE 10)

definition:

<identificateur>
< 10 >
<nom interne>
<phase de validation>
<action à transcoder>
<nombre de bits>
<1-ère valeur>

<n-ème valeur>

Les valeurs d'une action peuvent être codées dans les PLA, sur un nombre different de bits de ceux de la command à réaliser. Les valeurs transcodées, sont mises en correspondence avec les valeurs de l'action initiale (à transcoder). L'action initiale n'est plus considérée comme validée par une phase, tandis que l'action transcodée l'est.

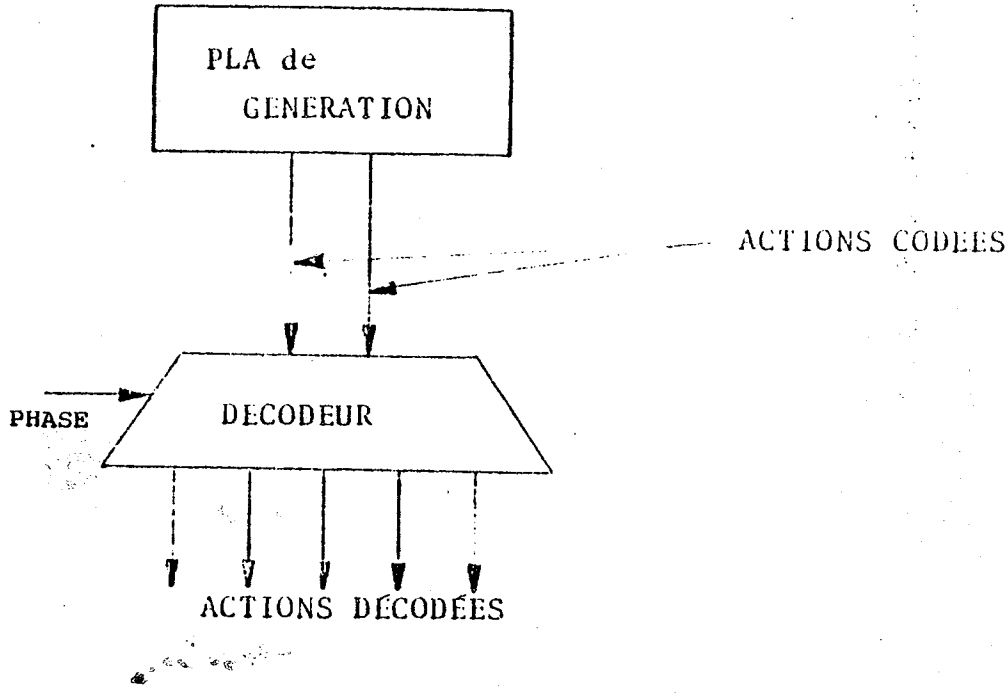


Figure 4-2 Schéma d'un transcodeur

****

4-ACTIONS DIRECTES (CLASSE 11)

Elles correspondent à la génération directe d'actions à partir des conditions.

.definition:

<identificateur>
< 11 >
<nom interne>
<phase de validation>
<condition>
<1- ère valeur>

<n- ème valeur>

Les valeurs de l'action sont mises en correspondance avec celles de la condition.

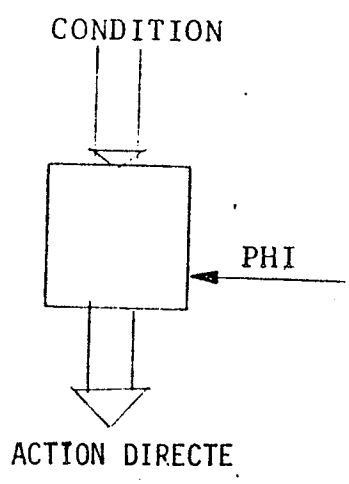


Figure 4-3 Mècanisme de calcul d'actions directes.

****

5-ACTIONS CONDITIONNELLES (CLASSE 5)

Ce sont des micro-commandes, conditionnelles, qui permettent au concepteur de decrire des formes de MEALY des algorithmes d'interprétation des instructions.

Definition:

<identificateur>
< 5 >
<nom interne>
<nombre d'actions>
<action binaire/simple 1->
<action binaire/simple 2-> actions initiales
<------------------------>
< condition >
<valeur par défaut> *
< 1-ème valeur action > (1-ère valeur de
<----------------------> la condition)
< n-ème valeur action >

Les dernières lignes du bloc décrivent la manière dont la condition valide les actions. La i-ème ligne (*) correspond à la i-ème valeur de la condition. Elle énumère les différentes valeurs des actions associées à cette valeur de la condition. Un 0 indique la valeur par défaut, c'est à dire que l'action n'est pas active. Cet indicateur ne peut évidemment prendre que les valeurs "0" et "1" pour les actions binaires.

6-ACTIONS PARAMETREES (CLASSE 12)

definition:

<identificateur>
< 12 >
<nom interne>
<phase de validation>
<nombre d'actions>
<action binaire/simple/transcodée/directe>
<----------------->
<----------------->
<action de validation>
<1-ère valeur action>
<-------------------->
<n-ème valeur action>

Les valeurs d'une action parametrée sont mises en regard des

valeurs de l'action de validation (idem actions conditionelles).

La phase de validation est appliquée à l'action resultante, ce qui remet en cause les phases de validation des actions initiales.

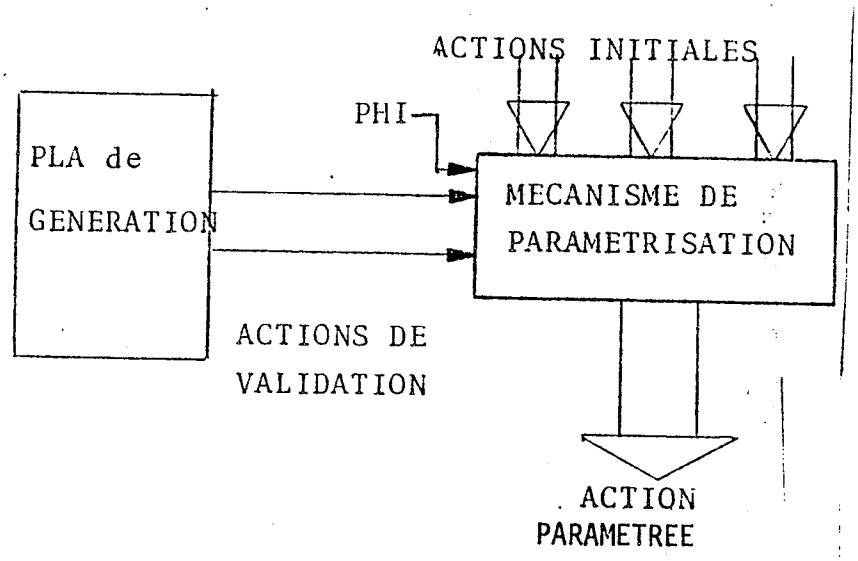


Figure 4-4 Principe de paramétrisation des actions

****

III-CONDITIONS

****

Les valeurs des conditions sont données en utilisant un code ternaire représentant l'état significatifs de leurs bits :

1,0 valeur binaires

X valeur ignorée

1-CONDITION SIMPLES (CLASSE 7)

definition:

<identificateur>
< 7 >
<nom interne>
<nombre de bits>
<nombre de valeurs>
<1-ère valeur>

<n-ème valeur>

Il s'agit de condition vehiculées par plusieurs fils.

2-CONDITIONS COMPLEXES (CLASSE 8)

Les conditions complexes nous permettent de considerer les différents produits des bits de conditions simples et leurs combinaisons, c'est à dire, la prise en compte de zones dans le tableau de Karnaugh, pour tout les bits de conditions considérés.

Definition:

<identificateur>
< 8 >
<nom interne>
<nombre de conditions impliquantes>
<1-ère condition>
<2-ème condition>
<n-ème condition>
<nombre de combinaisons>
<valeurs-1-ère combinaison >
<valeurs-2-ème combinaison >
<---------- >
<valeurs-n-ème combinaison >

Les valeurs d'une condition complexe sont representées par le configuration de bits des conditions simples. Chaque valeur comprend autant de bits qui constituent des conditions simples (voir V-6-2).

Exemple de description d'une condition complexe:

```
CONDITIONCPX   <- nom alphanumérique
     8         <- classe
 1702          <- nom interne
 2             <- nombre de condition simple engagée
 1405          <- 1-ère condition simple
 1408          <- 2-ème condition simple
 4             <- nombre de produits
 111X00010     <- 1-ère produit
 111X01110     <- 2-ème   "
 000111X1X     <- 3-ème   "
 000000111     <- 4-ème   "
   -1          <- mot clé de la fin de définition
```

Cet exemple decrit la condition complexe "CONDITIONCPX" de la classe "8". Celle-ci a un nom-interne 1702. Deux conditions simples: 1405 (8 bits) et 1408 (1 bits) sont engagées, et 4 produits de ces conditions sont utilisés. Le premièr produit "111X00010", considere le codage des bits de codition 1405, sous forme: 111X0001 et l'état de la codition 1408 est à "0". Les autres produits sont interpretés de la même façon.

3-CONDITIONS TRANSCODEES (CLASSE 13)

definition:

```
<identificateur>
< 13 >
<nom interne>
<condition à transcoder>
<nombre de bits>
<1-ère valeur>
<------------>
<n-ème valeur>
```

Les nouvelles valeurs de la condition peuvent être codées sur un nombre different de bit de la condition à transcoder. Les nouvelles valeurs sont mises en correspondance avec les valeurs de la condition à transcoder.

Exemple:

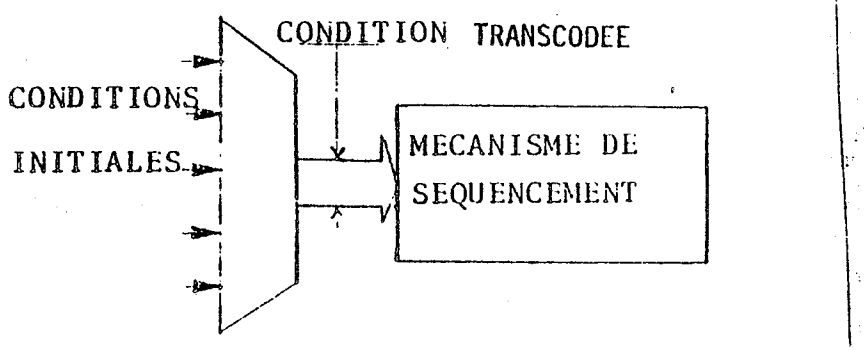


Figure 4-5 Mécanisme de conditions transcodées

****

La condition initiale à transcoder peut être une condition complexe regroupant des conditions simples ou complexes.

4-CONDITION DE PROPRIETE (CLASSE 14)

définition:

<identificateur>
< 14 >
<nom interne>
<nombre de conditions>
<condition simple/transcodée>

<action de validation>
<1-ère valeur>
<i-ème valeur>
<n-ème valeur>

Elles correspondent à la séléction de sous condition par une action.

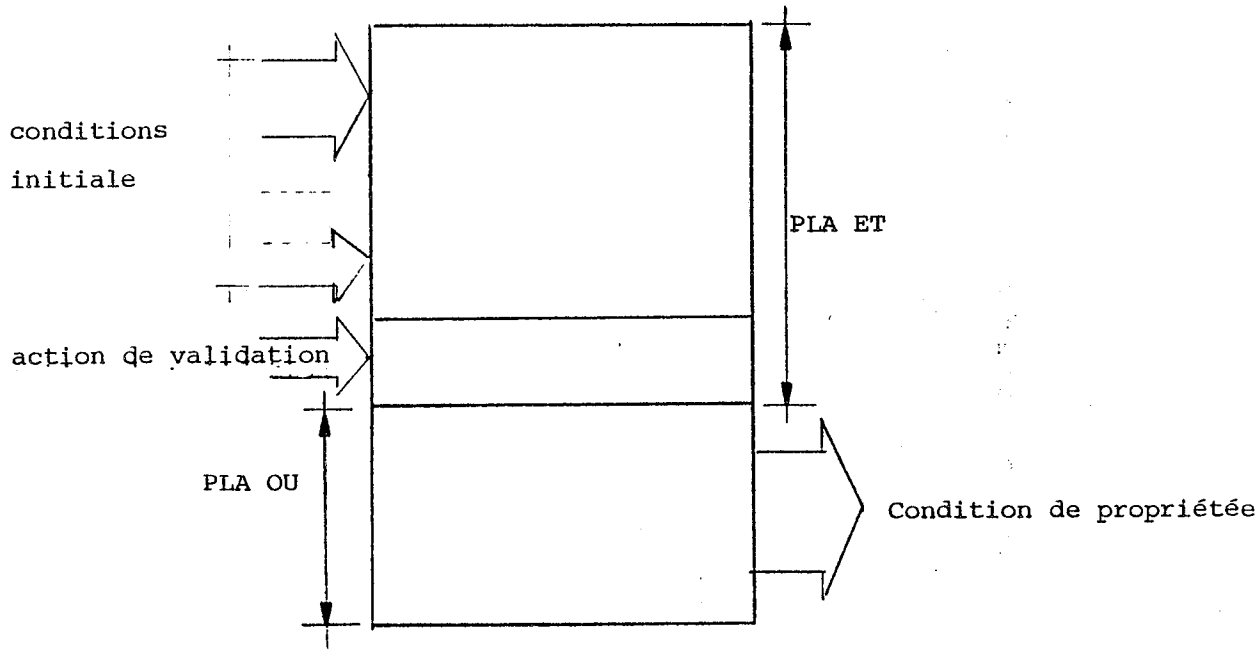
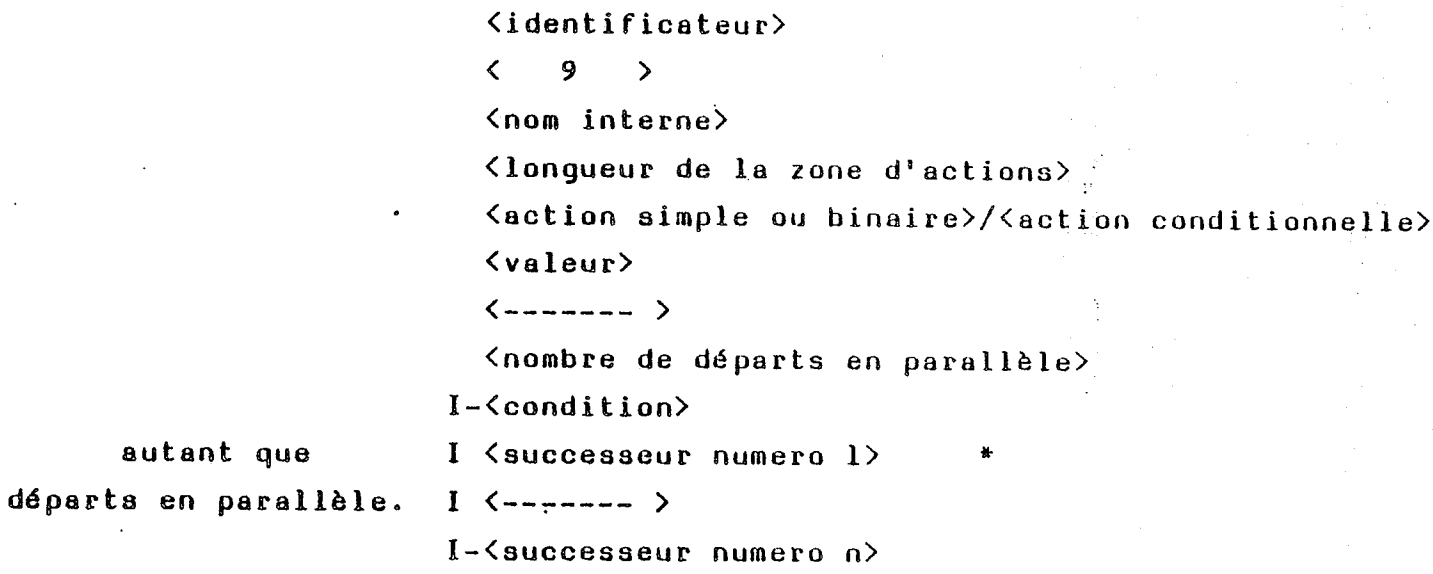


Figure 4-6 Les conditions de propriétées

****

V-BLOC D'ETAT (CLASSE 9)
****

Definition:



* Successeur par rapport à une valeur de la condition

Les actions non mentionées sont supposées être dans leur valeur par défaut ou inactive.

CHAPITRE V

STRUCTURE DE DONNEE DU COMPILATEUR

1-Introduction:

Le compilateur de PLA permet de générer la partie controle d'un microprocesseur. Cette partie est constituée de plusieurs PLA dont les specifications sont decrites en langage intermèdiaire. Chaque PLA est constitué de 2 ou plusieurs éléments du langage dont les liaisons sont assurées grâce à des identificateurs.

2-Structure de donnée de système

Les differentes architectures de parties contôles peuvent être constitués de PLA ou de ROM. La décomposition de l'algorithme d'un automate en éléments simples, tels que ceux décrits dans le quatrième chapitre, nous permet de générer les PLA. Cependant, le problème de l'accès à ces éléments, qui contiennent beaucoup d'information, va se poser. Nous avons donc prévu un mécanisme d'accès correspondant aux noms internes des éléments (voir tableau de codage). Chaque élément aura deux noms:

-un identificateur alphanumerique donné par l'utilisateur.

-un entier associé à cet identificateur permettant leur gestion rapide.

3-Noms internes

L'utilisation des nombres entiers de 1 à 2000, comme noms internes, simplifie la programmation. Cet intervalle de nombres est divisé en differents sous-intervales, appelés classes, pour differencier les types des éléments.

Le tableau suivant indique les zones attribuées aux éléments.

ZONE DE NOMS INTERNE	CLASSE	ELEMENTS
1:40	2	Phases de chargement
41:80	1	Phases de validation
81:100	6	Actions changées de phase
101:150	10	Actions transcodées
151:200	11	Actions directes
201:400	4	Actions simples
401:600	12	Actions paramétrées
601:800	5	Actions conditionnelles
801:1100	3	Actions binaires
1101:1200	13	Conditions transcodées
1201:1399	14	Conditions de propriétés
1400	7	Condition toujours vraie
1401:1700	7	Conditions simples
1701:2000	8	Conditions complexes
2001:3000	9	Bloc d'etats

****

Tableau de codage des éléments

Le deuxième point d'interet est l'utilisation des noms internes comme des pointeurs pour accéder à leur structure fine. La structure detaillée des éléments ne contient que les entièrs graçe aux noms internes, sauf pour le cas exceptionnel des conditions complexes. La description fine des éléments sera rangée dans une grande table de dictionnnaire. Celle-ci a la taille de milliers d'entiers.

4-les procédures d'accés et de génération des tables.

Deux procédures et deux fonctions sont toujours utilisées pour la génération et l'exploitation des tables par une technique de H-CODE:

1-la fonction HASH
2-la procédure CREEH
3-la procédure CHERCHE
4-la fonction EXISTE

Ces quatre procédures sont utilisées pour acceder aux tables de noms internes au cours de l'execution du programme. La fonction HASH favorise l'accès aléatoire à la structure de données. La procédure CREEH, remplit les tableaux de dictionnaire et de référence. La procédure CHERCHE, trouve un élément déja enregistré dans ces tableaux. Ces deux procédures utilisent la fonction HASH pour réaliser l'operation de HASH-CODE.

La fonction EXISTE, cherche dans le tableau de correspondance l'identificateur d'un nom-interne donné.

5-La description algorithmique textuelle

La description algorithmique doit être stockée sous formme d'un fichier sur la disquette utilisateur de la "PASCALINE". Les règles suivantes doivent être respectées:

- Il faut associer à chaque élément un identificateur avec une longueur maximum de 10 caractères. Des identificateurs plus long peuvent être utilisés mais seulement les 10 premières

lettres sont considerées par le programme.

- les entiers décrivant la structure fine d'un élément commencent toujours par la classe de l'élément, puis son nom interne. Ces derniers se trouvent sur la ligne suivante de l'identificateur. les autres entiers suivent le nom interne selon la definition de l'élément.
- La fin de la description fine des éléments sera reperée par "-1".
- La fin de la description correspond au mot clef "FIN"

Remarque (cas des conditions complexes): Les produits de ces conditions doivent être rangés un par ligne. Le code terminaison "-1", doit etre placé sur la ligne suivante du dernier produit déclaré.

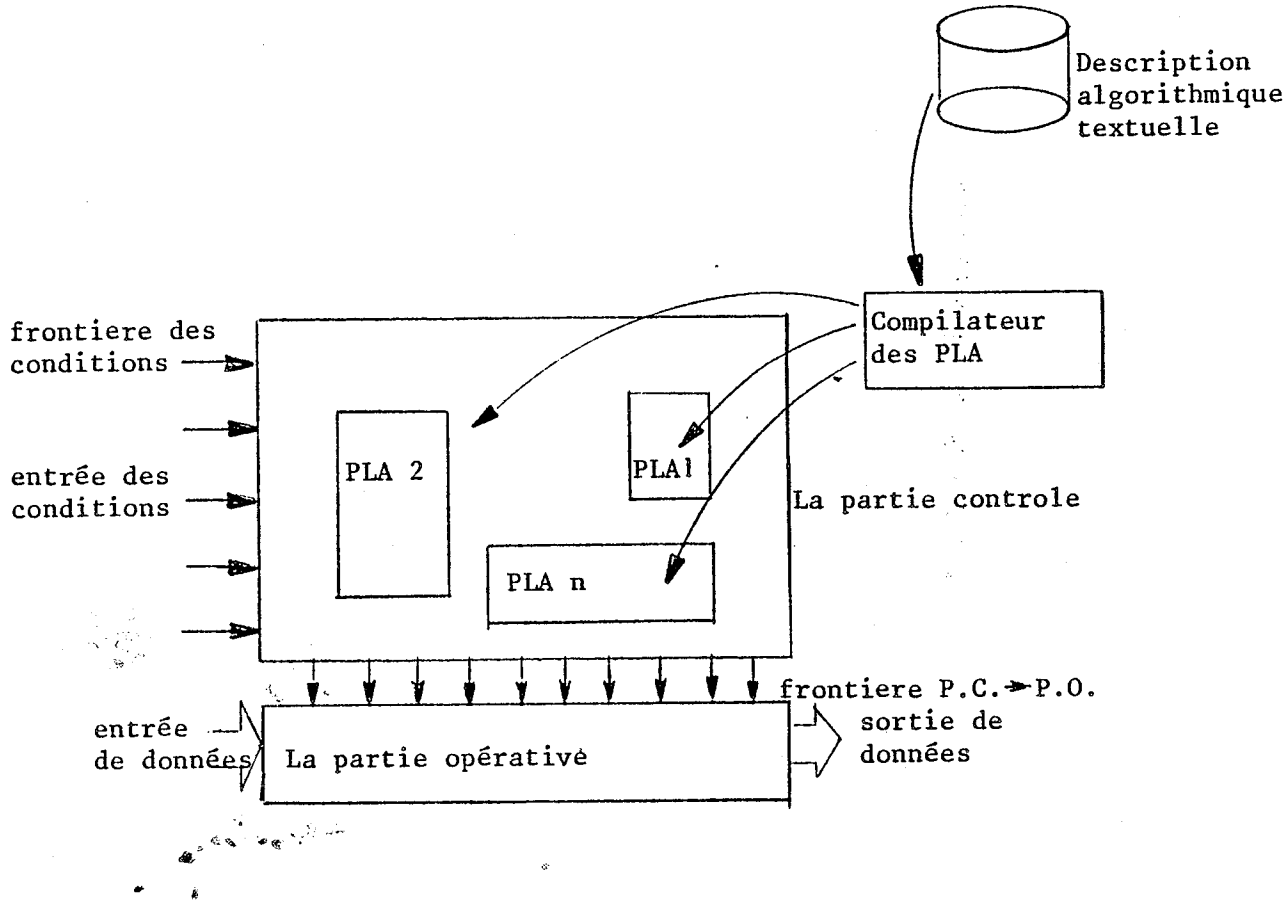


Figure 5-1 Génération des PLA par le compilateur

****

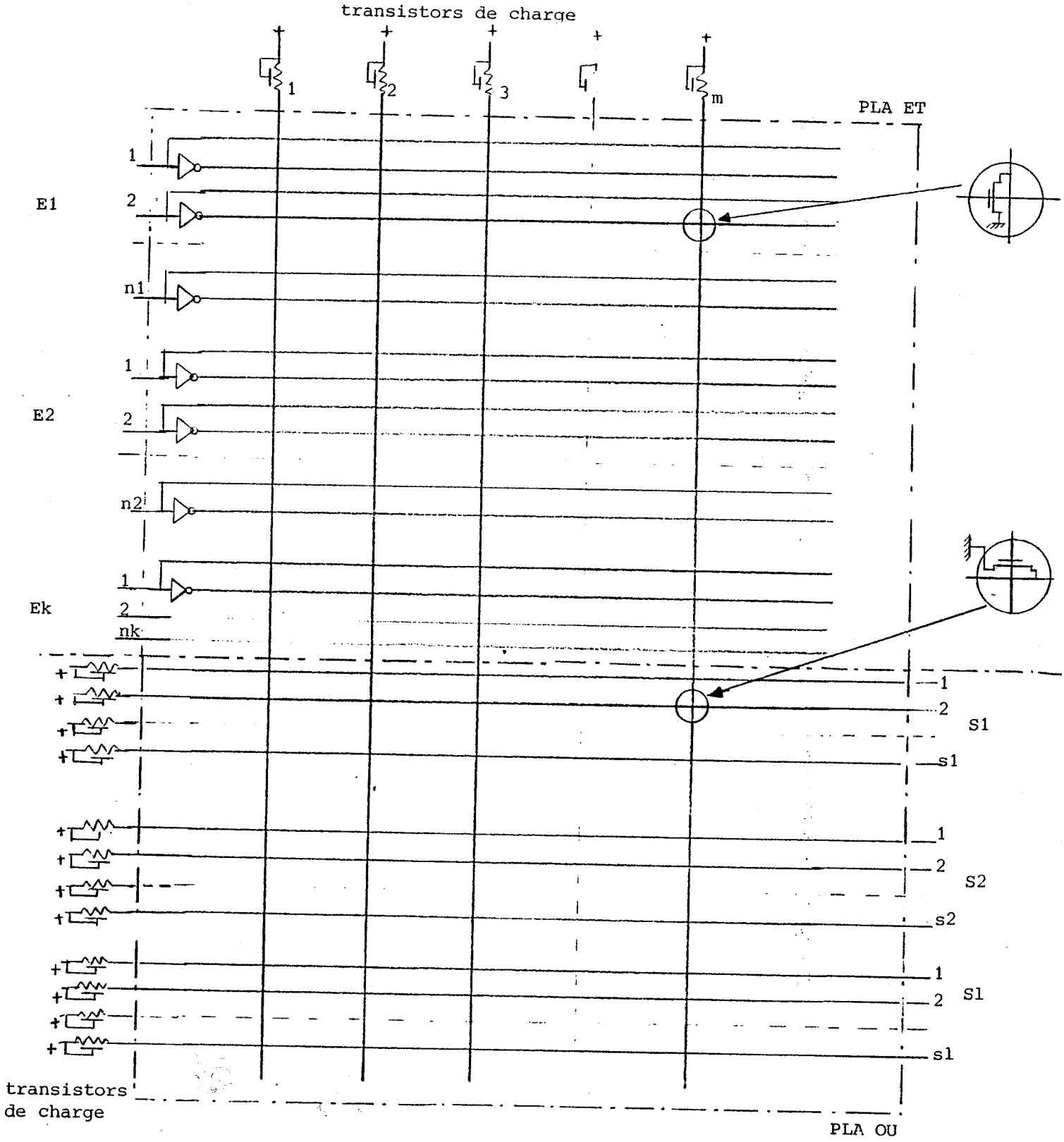
CHAPITRE VI

REALISATION MATERIELLE DES ELEMENTS

DU LANGAGE INTERMEDIAIRE

1-Introduction:

Les paragraphes suivants présentent les correspondances matérielles de tous les éléments du langage intermédiaire. Pour mieux comprendre la réalisation d'un PLA à travers les définitions, la structure parametrée du PLA cible est présentée dans la figure 6-1. Les deux parties ET et OU de ce PLA ont le même nombre de monômes "m". Les "k" groupes d'entrées sont constitués de nk bits qui correspondent aux 2*nk lignes du PLA-ET. Le nombre de groupes de sortie est égal à 1. Chaque groupe est constitué de sl bits. Les transistors de charge alimentent les lignes de sortie du PLA-OU et les monômes. L'activation des monômes par les lignes d'entrée est effectuée par l'implantation d'un transistor sur le croisement de cette ligne et du monôme envisagé. Cette activation correspond au profil des bits d'entrée presentés par les valeurs dans la définition du langage intermédiaire.



m: nombre de monomes
k: nombre du groupes d'entrée
nk:nombre de bits de chaque groupe d'entrée

l: nombre du groupe de sortie
sl:nombre de bits de chaque groupe de sortie

Figure 6-2 Schéma d'un transcodeur

****

Exemple de présentation des éléments

1-Les Phases

Ces éléments permettent de définir l'interface entre la partie contrôle et la partie opérative. Les phases définissent le systéme d'horlogerie du processeur.

2-1-La réalisation matérielle

Dans la réalisation matérielle il s'agit d'une porte de synchronisation pour chaque action de sortie.

Le nombre de portes dépend du nombre d'actions. Les différents outils d'assemblage des blocs peuvent être utilisés pour une connexion automatique de la P.C. à la P.O.

La figure 6-2 montre une partie opérative et les lignes d'horlogerie du système. Ces lignes relient les générateurs d'instants du système aux portes de synchronisation. La génération automatique de cet interface par un outil d'assemblage de blocs, tel que LUBRICK ref. [SCH-83], nécessite la génération d'un fichier descriptif à partir des actions et des phases.

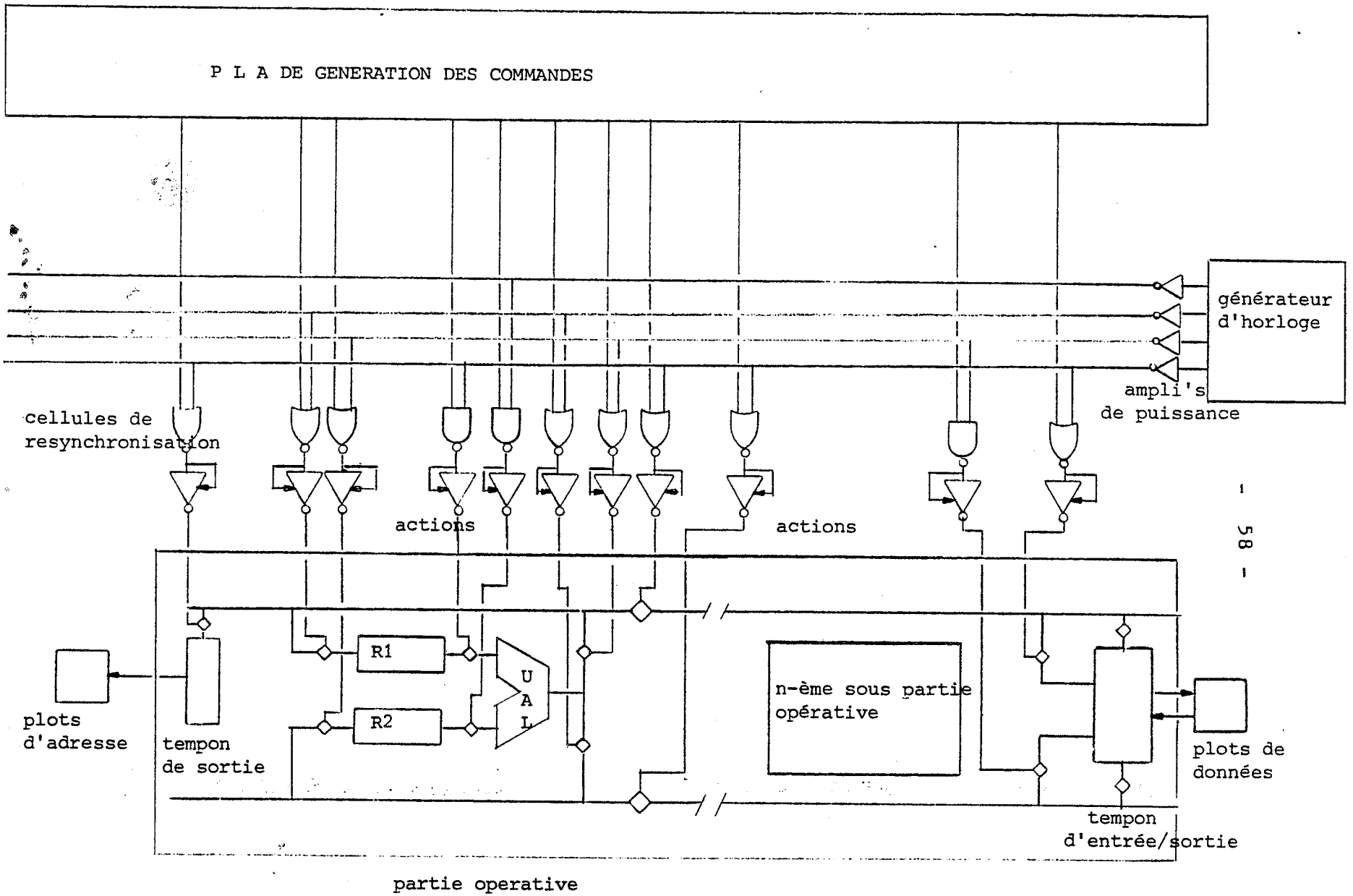


Figure 6-2 Schéma de l'interfaçe entre la P.O. et la P.C.

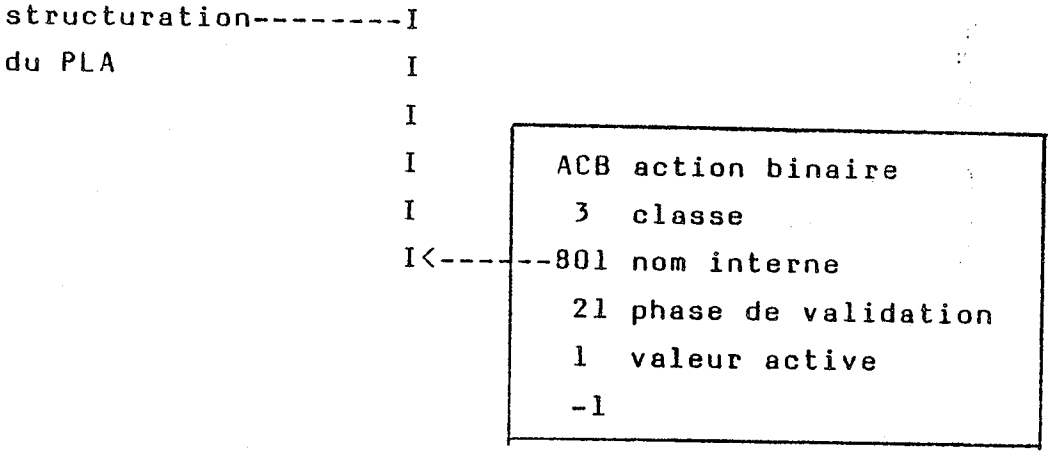
****

2-Actions binaires

Les commands binaires sont générées par un PLA de génération. Le bloc de source qui appelle une action binaire a préalablement généré la structure du PLA de la figure 6-1. La partie ET du PLA est générée et la matrice OU est initialisée d'avance. Les actions binaires constituent les lignes de sortie du PLA OU (lignes de sorties Sk). L'activation de cette sortie correspond à l'implantation d'un transistor sur le croisement de la ligne de la sortie et le monôme activé. L'activation de ce monôme, dépend d'une des valeurs des bits d'entrée du PLA ET qui est défini par l'élément de structuration du PLA.

2-1 Exemple.

```
P1 phase de sync.
2  classe
21 nom int.
-1
```



Cet exemple présente, l'action binaire "ACB" de la classe 3 et de nom interne "801". Celle-ci est synchronisée par la phase de nom interne "21". La valeur active de cette action est 1 logique. La figure 3 présente l'équivalent materiel de cette définition.

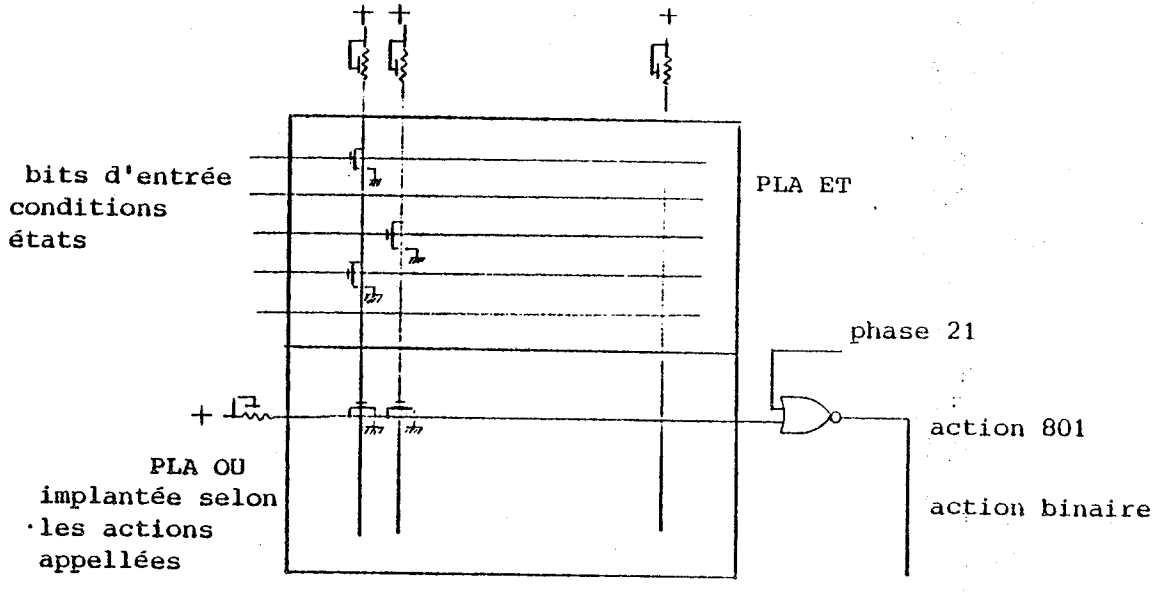


Figure 6-3-réalisation des actions binaires

****

remarques:

1- La structure de ce PLA est définie par les éléments qui définissent les lignes d'entrées du PLA ET et l'implantation des monômes dans cette partie.

2- La définition des entrées met en correspondance le bit d'action et le monôme activé.

3-Action simple

Objectif:

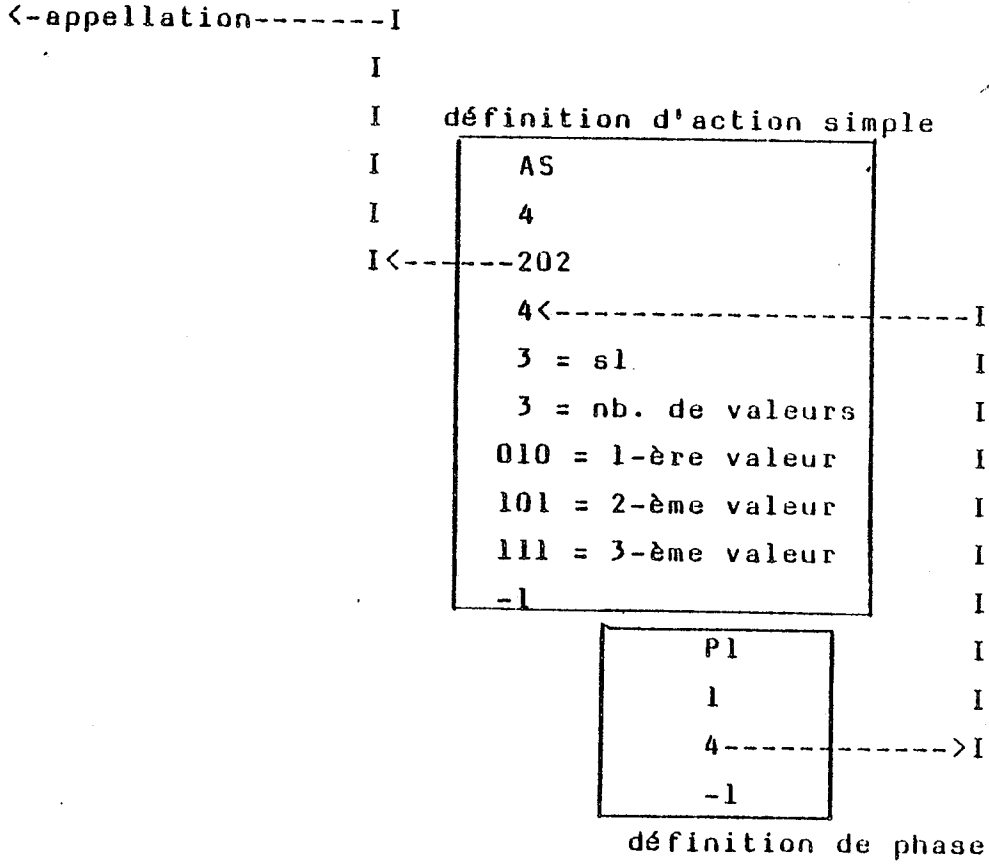
L'action binaire est une forme simplifiée d'action simple. Celle-ci présente plusieurs valeurs sous forme codées. Les informations suivantes sont déduites des actions simples:
-nombre de bits qui constituent cette action
-Les différentes valeurs ou configurations des bits qui presentent la manière de coder de cette action.

3-1 L'élément liés à l'action simple

Les actions simples appellent les phases de synchronisation. D'autres structures telles que les blocs d'états, appellent ces actions, pour générer un PLA de génération des commandes. La présentation matérielle de ces actions correspond à l'ensemble des fils de sortie du PLA qui est synchronisé par une phase.

3-2 Exemple

L'action simple "AS" générée par le PLA de figure 6-1, peut être présentée par l'exemple suivant:



Dans cet exemple, l'action simple "AS" de la classe "4" et le nom interne "202" est constituée de 3 bits. 3 valeurs: 010,101 et 111 sont utilisées.

L'élément de structuration qui appelle une action simple, défini la correspondance des valeurs à l'implantation des monômes.

3-3 Spécification du compilateur vis à vis des actions simples

Chaque appellation d'action simple de sl bits, designe sl lignes de la matrice OU. Les appellations des actions simples attribuent les lignes non-utilisées aux actions appélées.

4-Action transcodée

Objectif:

Cet élément génère un transcodeur,qui permet de décoder ou d'encoder les actions.

Une action de n bits peut véhiculer 2 I**I n valeurs. En cas de valeurs non utilisées, on peut transcoder ces valeurs en les exprimant dans un autre code qui comprend moins de bits. Au contraire on peut générer un décodeur pour transcoder les informations codées en binaire. Les parametres k et l de la figure 6-1 sont égaux à 1, indiquant un groupe d'entrée et de sortie.

4-1 L'élements liés aux actions transcodées

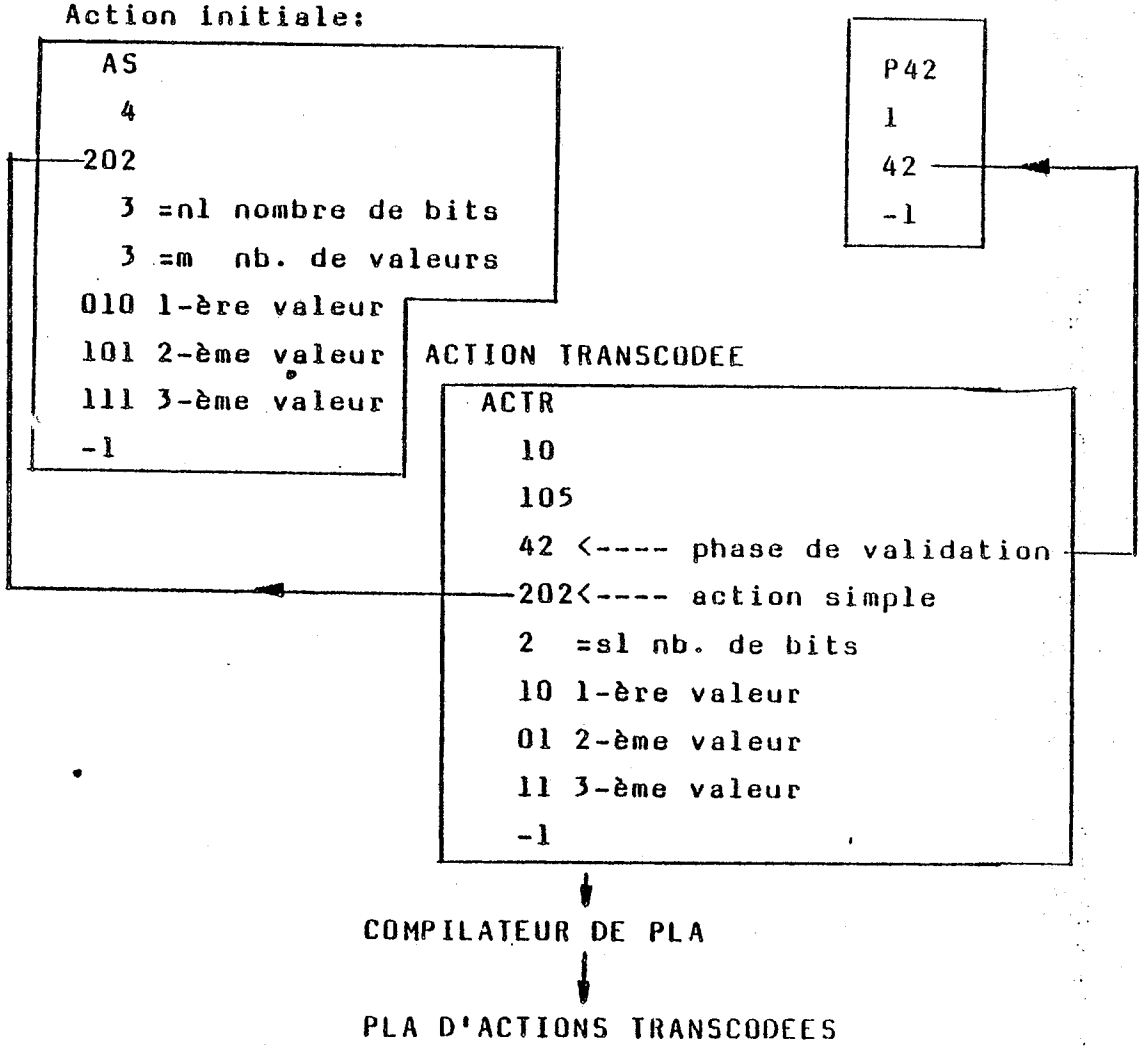
Chaque action transcodée, appelle l'action d'entrée. Celle-ci peut être une action simple de n bits et m valeurs.

La phase de synchronisation appellée, valide le décodeur pour synchroniser l'action de sortie.

4-2 Le PLA réalisé

Cette action génère la partie ET avec 2*n lignes (bits directs et complémentés) et m monômes. Le nombre de ses sorties correspond au nombre de bits (K) déclaré dans la structure de l'action transcodée.

4-3 Exemple d'action transcodée



Dans cet exemple, l'action transcodée "ACTR" de la classe "10" et le nom interne "105", est un transcodage de l'action simple 202. L'action "202" est transcodée sur 2 bits dont ces valeurs sont présentées par 10, 01 et 11.

5-Action Directe

Objectif:

Une telle action est générée à partir d'une condition. Un PLA est employé pour lier la condition d'entrée à l'action directe de sortie.

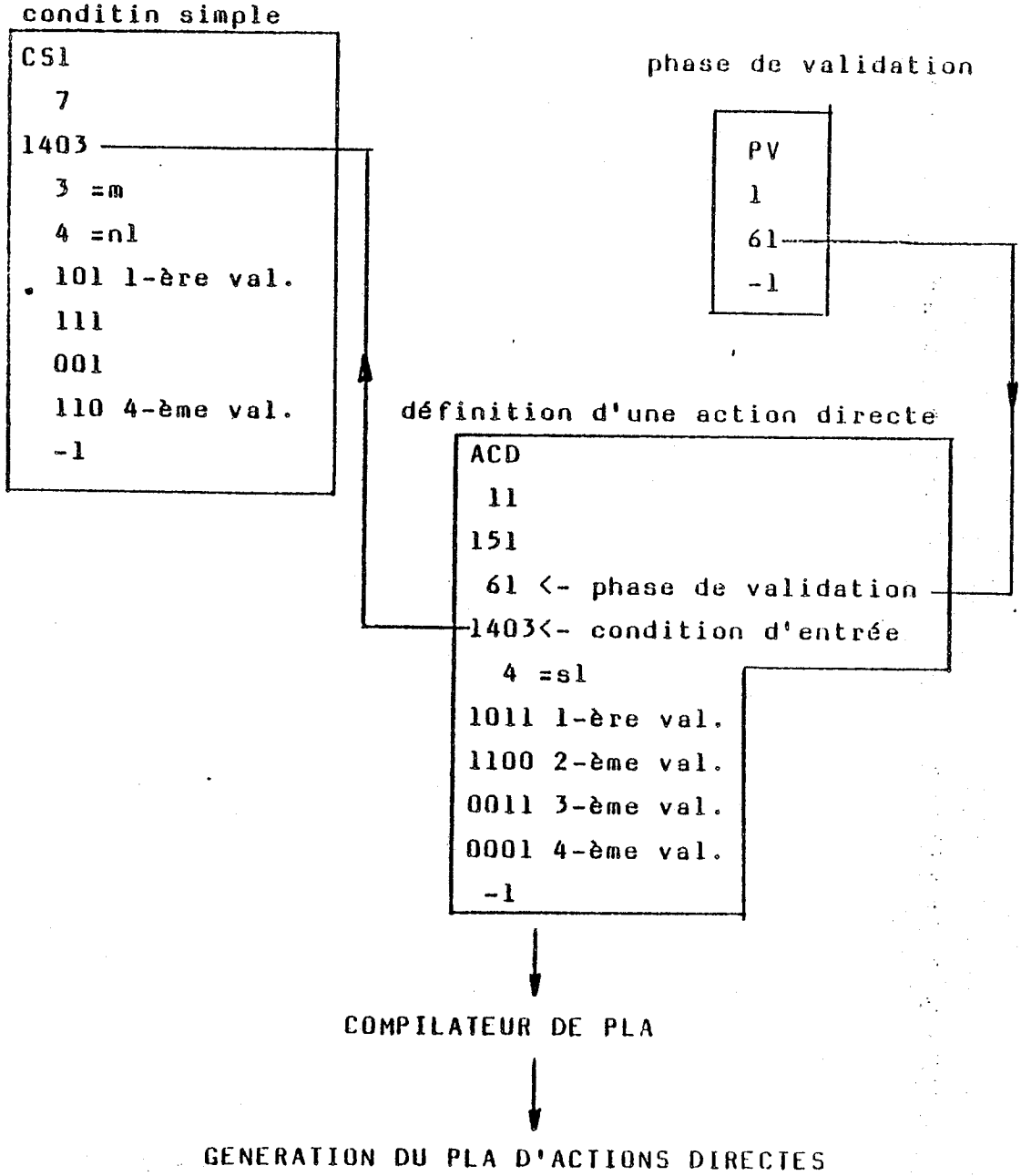
5-1-L'élément liés à cette structure

La definition de chaque action directe fait intervenir sa phase de synchronisation et la condition engagée. La structure des PLA ET et OU, est déduite de la condition d'entrée et de la structure de l'action directe. L'appel de la condition, permet de définir le nombre de bits d'entrée (nl) et le nombre des monômes du PLA ET. Selon les valeurs de la condition on trouve les valeurs de cette action qui définissent l'implantation des transistors dans le PLA OU. L'implantation des transistors du PLA ET, dépend des valeurs de la condition appélée. Les paramêtres du PLA de figure 6-1, k et l sont égaux à 1 en cas d'action directe.

5-2-Exemple

Une condition simple de 3 bits et 4 valeurs, peut etre mise en correspondance avec une action de 4 bits et 3 valeurs.

Exemple:



Dans cet exemple, l'action directe "ACD" de la classe "11" et de nom interne 151 est déduite de la condition simple "1403". Elle a 4 valeurs. La première est 1011 et la dernière 0001. La phase de synchronisation "PV", valide le PLA d'actions directes.

6-ACTION CONDITIONNELLE

Objectif:

Les structures des actions précédentes, ne génèrent q'une seule action. En cas de génération de plusieurs actions à partir d'une ou plusieurs conditions, la structure des actions conditionnelles doit être employée. Autrement dit, les paramêtres de figure 6-1 sont: k=1 et l>=1

La mise en valeur des actions par les conditions, permet de générer les actions paramêtrées par les conditions. En effet, l'algorithme d'interprétation des états est simplifé, par la paramêtrisation des états et la regénération des actions d'états qui sont éliminés de l'algorithme d'interprétation.

6-1-Les éléments liés à cette structure

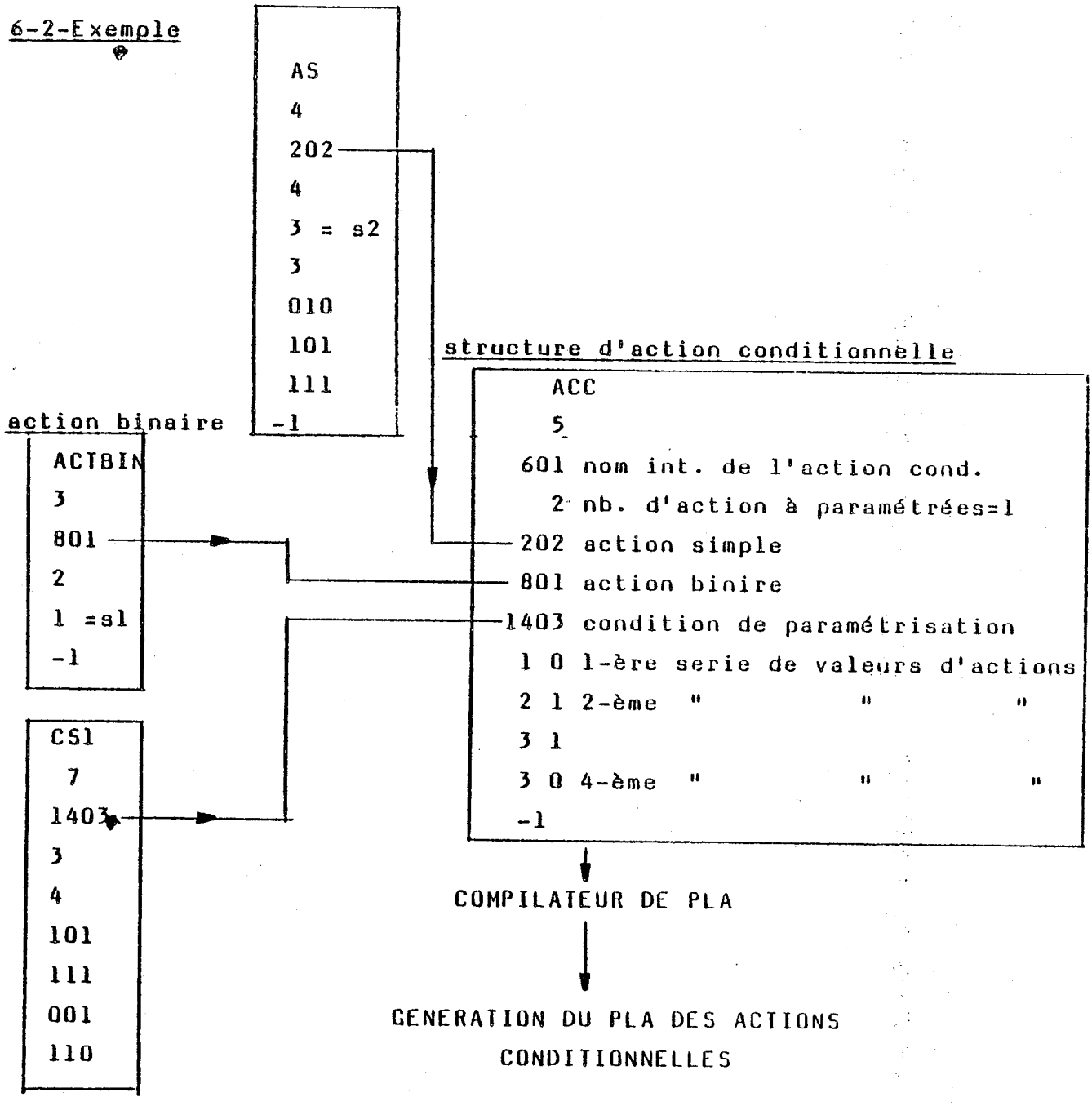
Cette structure utilise trois types d'éléments:

1- Condition de paramêtrisation. Une condition simple ou complexe initialise et génère la partie ET du PLA de paramêtrisation. Celle-ci défini le paramêtre nl du PLA de figure 6-1.

2- Actions à parametriser. Ces actions définissent les bits de sortie du PLA OU. Les nombres de bits des actions employées, sont calculés à partir de leurs définitions. L'appel des identificateurs des actions établit le lien pour calculer les valeurs correspondantes.

3- Bloc d'état, qui appelle l'action conditionnelle, pour générer un ensemble d'actions paramétrées. Cet état répresente un ensemble d'états, que l'on a regroupé pour simplifier l'algorithme.

6-2-Exemple



Dans cet exemple, l'action conditionnelle "ACC" de la classe "5" a le nom interne "601". Celle-ci paramêtrise 2 actions:

1- l'action simple 202

2- l'action binaire 801

La condition de parametrisation a pour nom interne "1403". Les valeurs des actions (202 et 801) sont positionnées par les valeurs de la condition "1403" indiquées dans la définition. Chaque valeur de la condition 1403 correspond aux valeurs des 2 actions.

6-3-Specification du compilateur par rapport aux actions conditionnelles

Le compilateur énumère tous les états de l'algorithme et chaque fois qu'on trouve une action conditionnelle, deux matrices ET et OU sont générées. La procédure chargée d'analyser les actions conditionnelles est appelée "GMCPAR". Cette procédure appèle deux autres procedures, chacune est chargée de générer une matrice. La procédure IMPLCND génère l'implantation des bits pour détecter la valeur de la condition (PLA ET). La procédure IMPLACTS est chargée de mettre en valeur les bits d'action (PLA OU). La génération de ces actions est faite bloc par bloc.

6-4-Le lien entre le PLA de génération des commandes et les actions conditionnelles

Les actions générées par les blocs d'actions conditionnelles, sont les mêmes que celle générées par le PLA de génération donc il faut un tableau de correspondance qui garde l'identité des bits d'action pour les deux matrices OU. L'une est générée par les blocs d'état (à voir dans la partie consacrée au blocs d'état) et l'autre par les actions conditionnelles. La figure suivante montre cet effet.

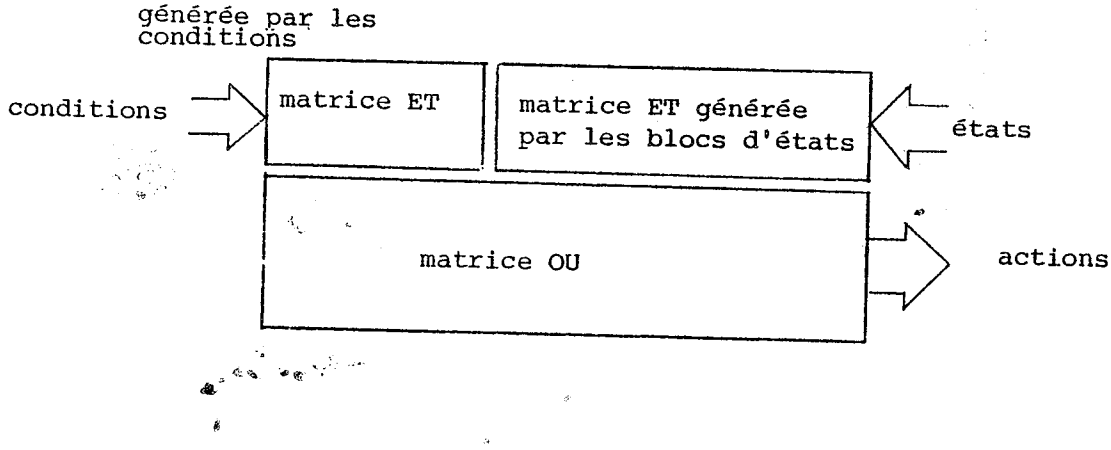


Figure 6-4 PLA de génération avec la matrice OU commune

****

7-ACTION PARAMETREE

Objectif:

Plusieurs actions sont mises en oeuvre par une action de validation. Cette action est un élément de structuration du PLA de la figure 6-1 avec k=1 et l>1. La structure des actions paramêtrées est équivalente à celle des actions conditionnelles sauf que, la condition de selection est remplacée par une action de validation.

Les sous-blocs de la partie opérative qui executent une operation spécifique au cours de l'exécution de l'instruction, reçoivent les actions paramêtrées. On peut citer l'exemple des PLA de paramétres pour l'UAL ou celui du registre code condition.

La génération des commandes qui dans une réalisation du type deux PLA était implantée avec un PLA de commandes unique peut ici être réalisée avec un PLA de validation et plusieurs PLA de paramêtres. Par conséquent, on obtient un gain du surface assez important.

7-1- L'élément lié à l'action paramêtrée

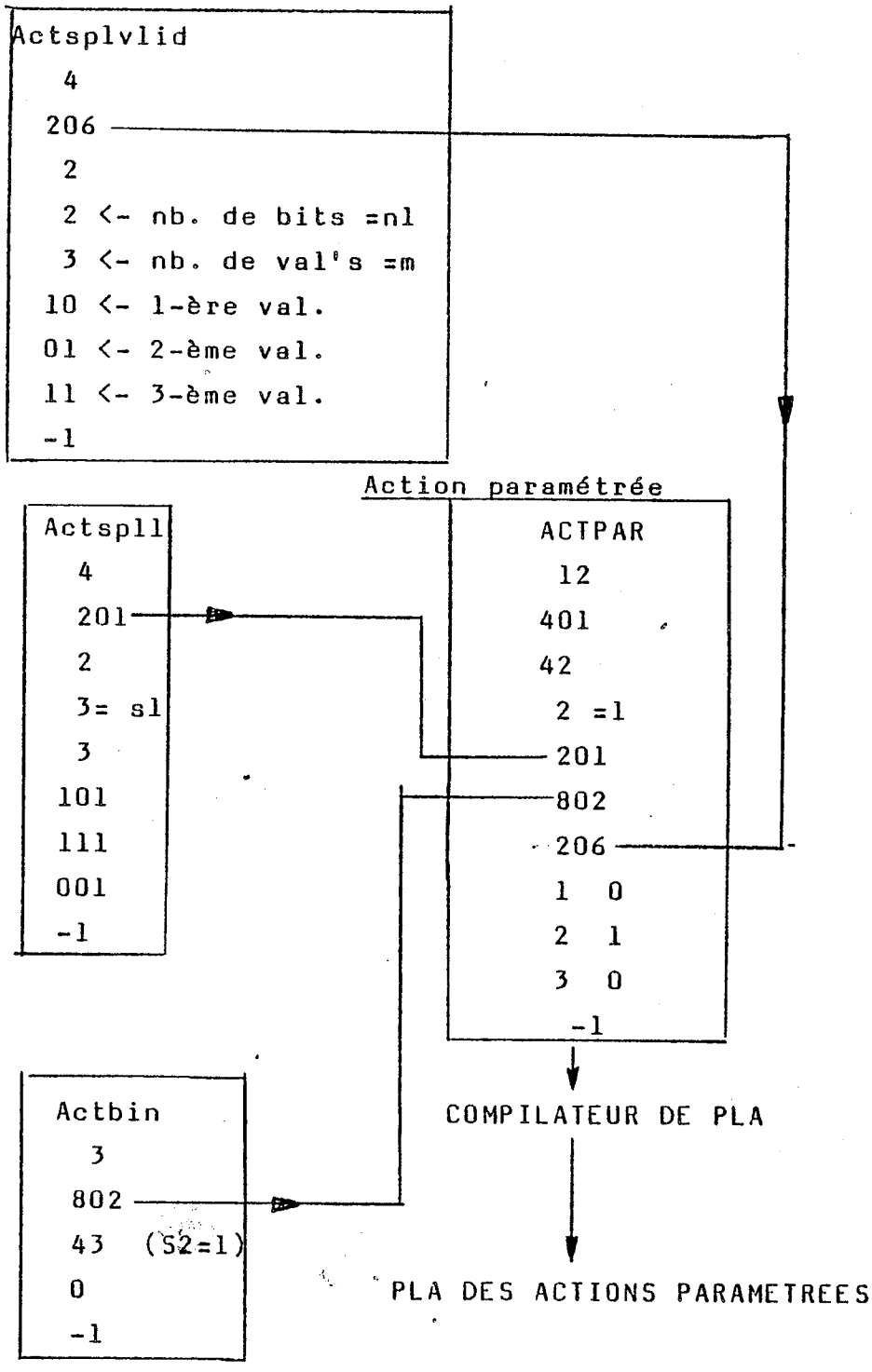
Les éléments suivants sont appelés par une action paramétrée:

1-L'action de validation, issue d'un PLA de validation.

2-Les actions de sortie de ce PLA.

3-Une phase de validation.

7-2-Exemple



Dans cet exemple l'action paramêtrée "ACTPAR" de la classe "12" a le nom interne "401". Deux actions "actspll" et "actbin" sont paramêtrées par l'action de validation "actsplvlid".

8- CONDITION SIMPLE

Objectif:

Les conditions simples sont le lien entre le monde extérieur qui commande la partie contrôle et les PLA générés. Chaque condition simple défini les bits et leurs profils. Les entrèes du PLA de la figure 6-1 sont définies par ce type de condition.

Par exemple dans un microprocesseur le code opératoire est appliqué à travers des bits d'actions simples. Selon le type de processeur on peut définir une condition simple de 8,16 ou 32 bits. Les commandes binaires, telles que l'initialisation, les interruption, l'arret,...etc., sont ainsi exprimées par cette structure.

8-1-Exemple

L'exemple cité pour le cas des actions directes explique la manière de prise en compte d'une condition simple par cette structure.

```
                        CS1
                        7
<---appellation ------1403 nom interne
                        3 =n1
                        4 =m
                        101 1-ère valeur
                        111 2-ème valeur
                        001 3-ème valeur
                        110
                        -1
```

Dans cet exemple la condition simple "CS1" de la classe 7 a le nom interne "1403". Celle-ci est constituée de 3 bits qui véhiculent 4 valeurs. La première valeur est "101" et la dernière est "110".

9-CONDITION COMPLEXE

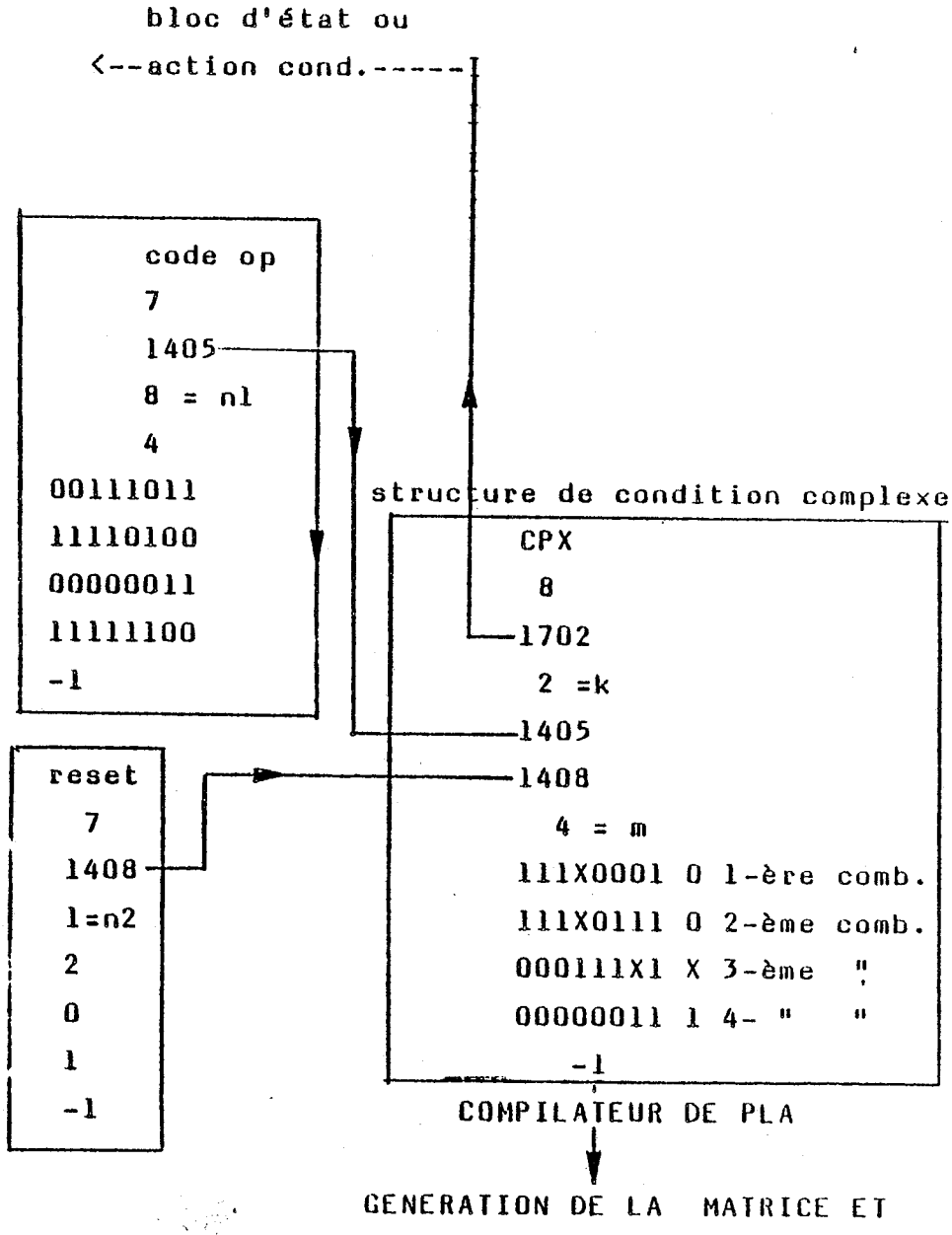
objectif:

La structure des conditions complexes permet de définir les produits de plusieurs conditions simples pour activer les monômes du PLA envisagé. Celle ci peut représenter les produits des bits des conditions pour representer l'implantation des monômes du PLA-ET. Les "k" groupes d'entrées du PLA de la figure 6-1 et l'implantation des bits des monômes sont défini par cette condition.

9-1-L'éléments liés à cette structure

Chaque condition complexe, appelle des conditions simples pour accèder au nombre de bits des conditions qui commandent le PLA. Chaque condition complexe est appellée par un élément de structuration du PLA, tel que les blocs d'états ou les actions directes.

9-2-Exemple



remarque: Les valeurs déclarées dans les conditions simples ne sont pas utiliées par la condition complexe. Dans cet exemple, la condition complexe "CPX" de la classe 8 et le non interne "1702" considère 2 conditions simples "1405" et "1408". 4 produits de ces conditions sont utilisés par la condition CPX.

(* CH61 *)

10-CONDITION TRANSCODEE

Objectif

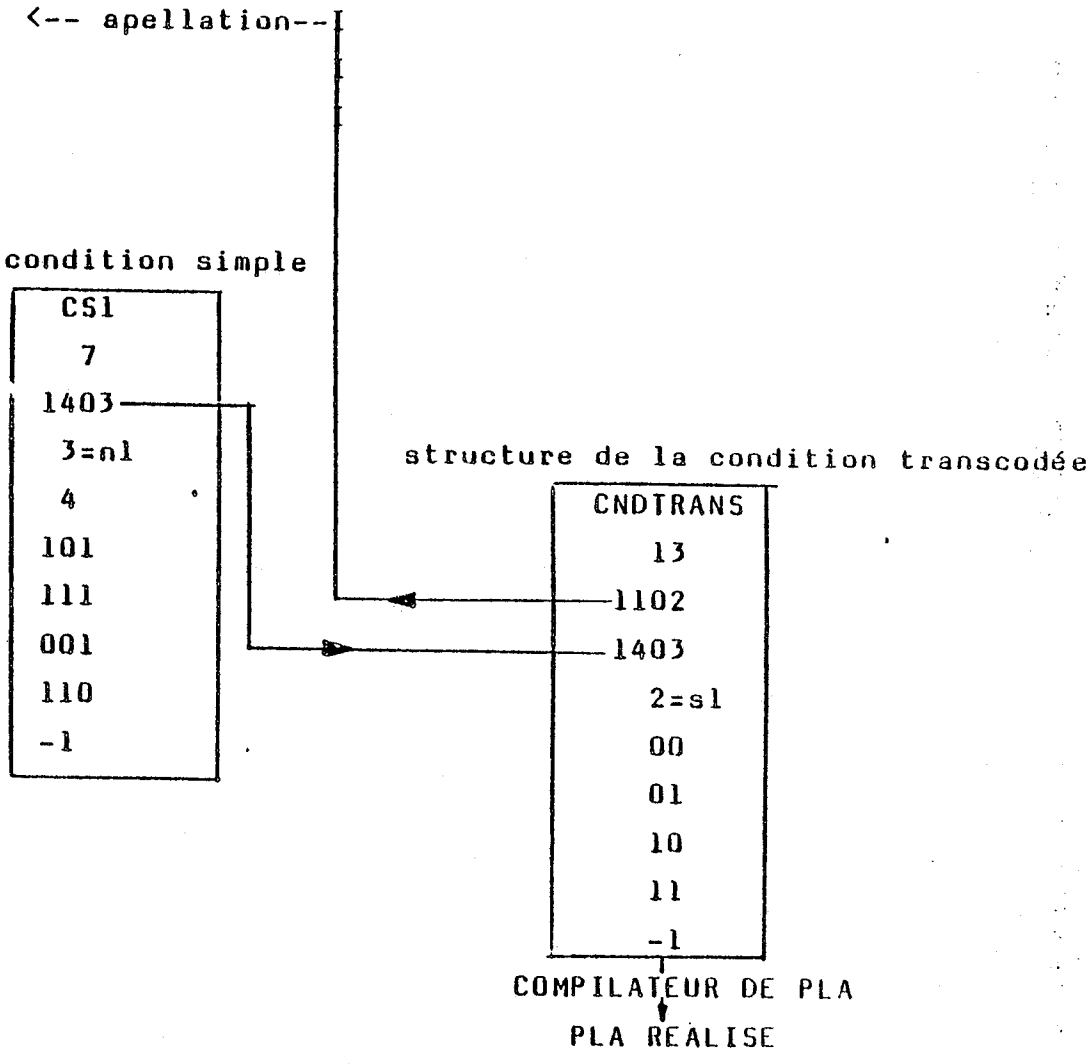
Un bloc de transcodeur sous forme d'un PLA est réalisé à partir de cet élément. Celui-ci permet de changer le codage des bits d'une condition pour les représenter par un nombre différent de bits. On peut diminuer le nombre de bits, en profitant des valeurs non-utilisées d'une condition. Une condition de n-bits peut véhiculer 2 nvaleurs. En cas de:

$2^n - 2^{n-1}$ valeurs non-utilisées, le n-éme bit peut être

supprimé, en transcodant les nouvelles valeurs sur n-1 bits. Cette structure appelle la condition à transcoder. Celle-ci va permettre de construire le PLA ET et les valeurs de condition transcodée pour l'implantation du PLA OU.

Le PLA est réalisé de la même façon que le PLA des actions transcodées, sauf que l'action d'entrée est remplacée par une condition. Le PLA de la figure 6-1 en cas de condition transcodée obtient les paramétres suivants: k=1=1.

Exemple:



Dans cet exemple, la condition transcodée "CNDTRANS" de la classe "13" et de nom interne 1102, transcode la condition simple 1403. 4 valeurs sont transcodées sur 2 bits avec les nouveaux codes 00, 01, 10 et 11.

11- CONDITION DE PROPRIETE

Objectif

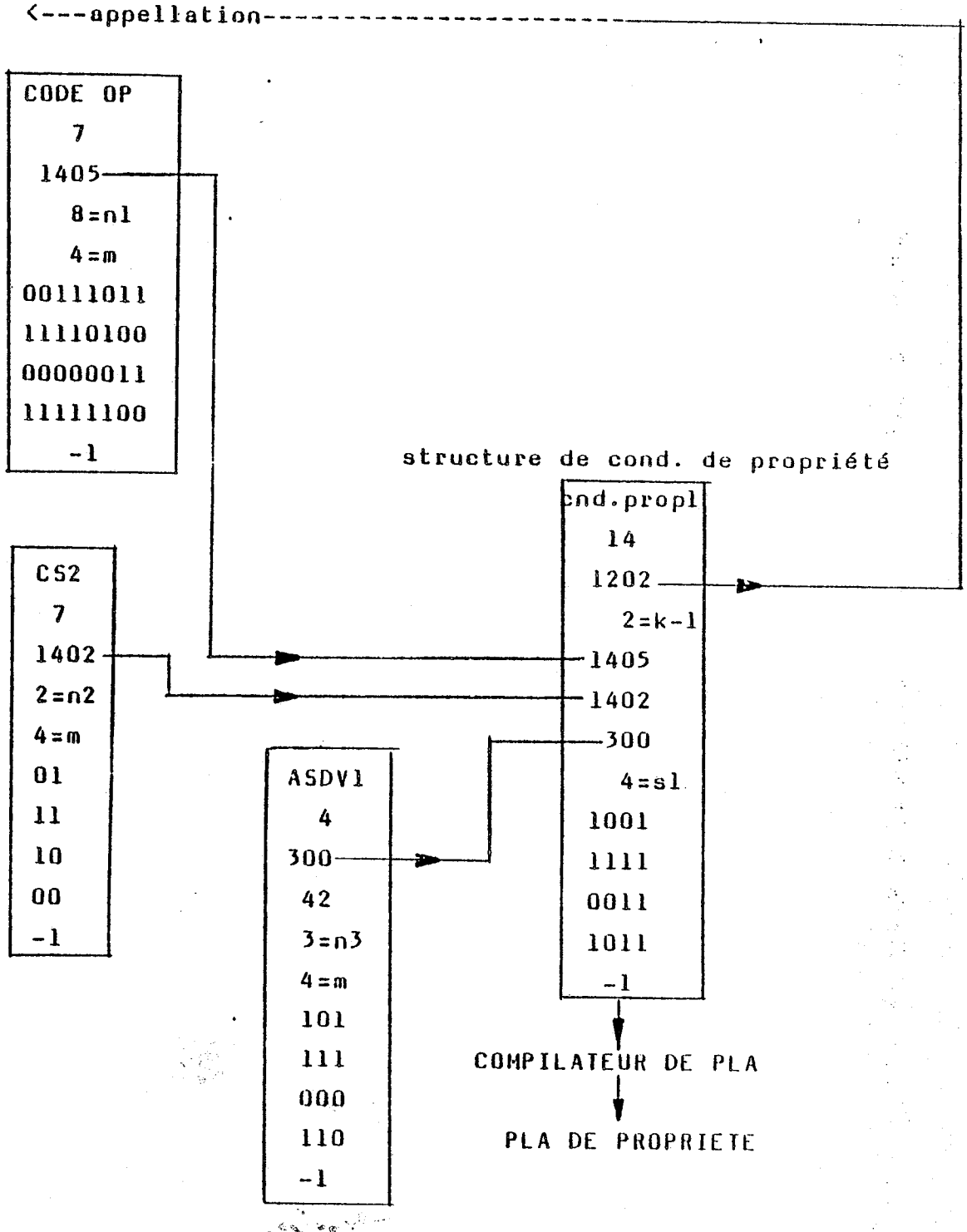
Cette structure permet de générer un PLA de validation. On trouve deux types d'entrée pour le PLA ET, les conditions et une action de validation du type simple. Le PLA ET est généré à partir des conditions et de l'action de validation. Alors l'activation de chaque monôme dépend des valeurs des conditions et de l'action. Chaque monôme actif met en valeur les n bits de sortie. La manière dont ces bits sont positionnés est présentée par les valeurs de condition de propriété.

11-1- Eléments liés

Chaque condition de propriéte appelle plusieurs conditions. Ces dernières peuvent être du type simple ou transcodée. Donc on peut accèder au nombre de bits et aux valeurs de chacune des conditions.

Une action de validation du type simple permet d'acceder au nombre de bits et à la valeurs de cette action. Les sorties sont les bits du PLA-OU qui constituent la condition de propriété.

11-2- Exemple



La condition de propriété "cnd.propl" de la classe 14 et le npm interne 1202 engage deux conditions simples 1402 et 1405. L'action de validation de nom interne 300, génére 4 produits de cette condition 1001, 1111, 0011 et 1011.

12-BLOC D'ETAT

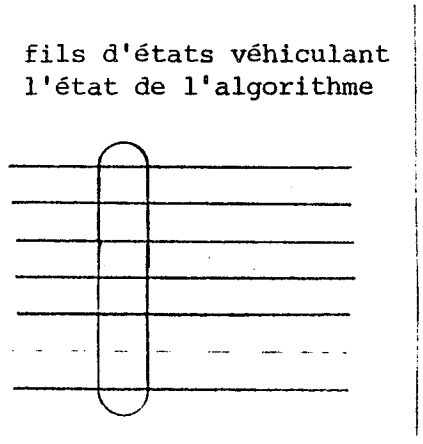
Objectif:

Les blocs d'état permettent au concepteur de définir l'algorithme d'interprétation des instructions; afin de générer un PLA d'enchainement et un PLA de génération des commandes. La spécification d'un état de l'algorithme est prévue par chaque bloc d'état. Les paragraphes suivants détaillent la manière de définir un algorithme à l'aide de blocs d'états.

12-1-L'état de l'algorithme

La définitiom du bloc d'état propose un nouvel élément par rapport aux éléments déjà étudiés. D'une manière générale pour coder E états d'un algorithme on a besoin de n bits ou la valeur de n est déterminée par la relation :

$2^n <= Ek$



Fils véhiculant l'état de l'algorithme

****

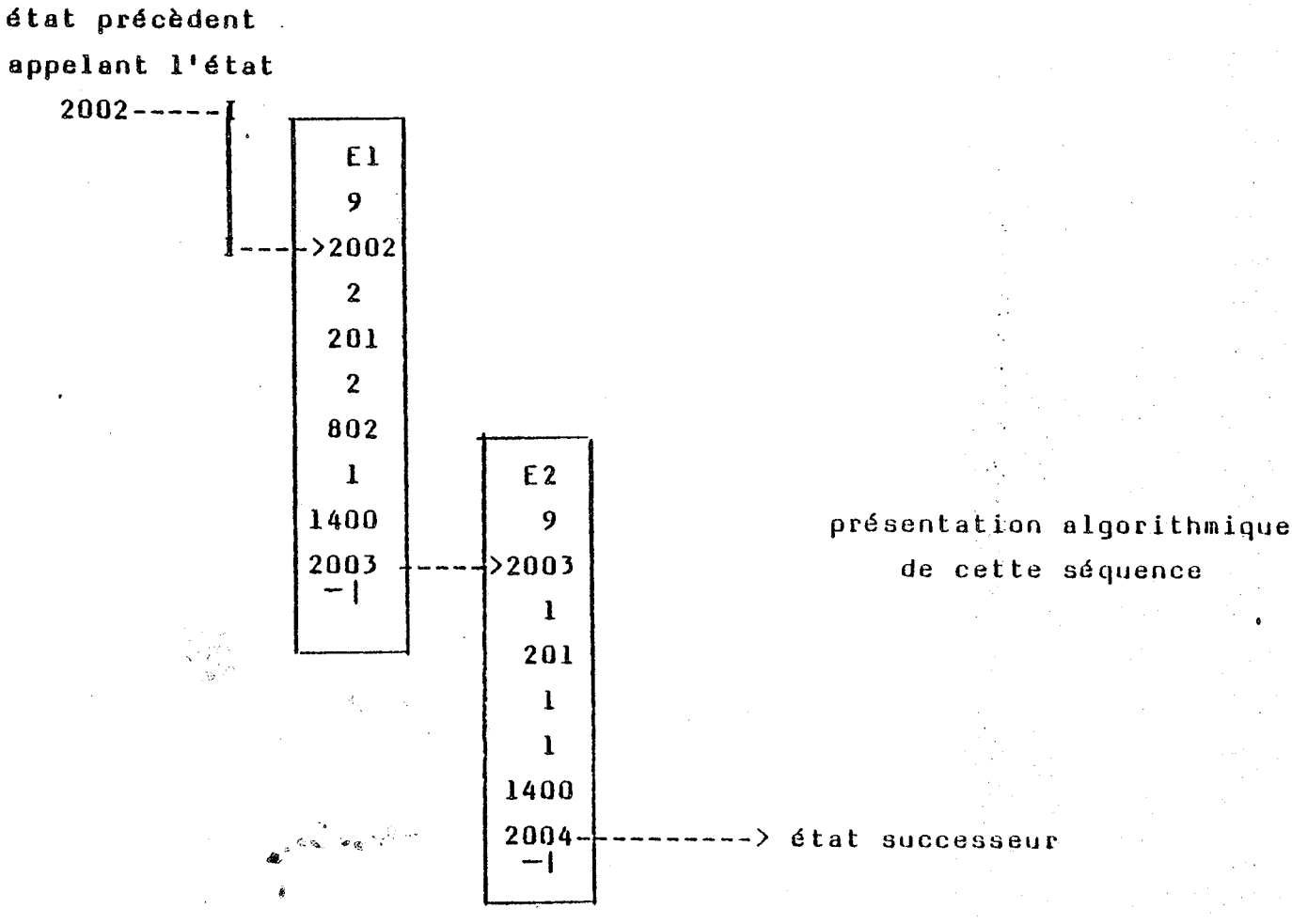
Les profils de l'ensemble des fils d'états sont détectés

par la partie ET du PLA d'enchainement (entrée Ek). L'état suivant est généré par les lignes de sorties sl.

12-2-L'enchainement de l'algorithme

Chaque bloc d'état définit le(s) nom(s) interne(s) de ses états successeurs ce qui détermine l'enchainement des états dans l'algorithme d'interprétation. Dans la réalisation matérielle, il s'agit de reconnaitre l'état présent dans la matrice ET du PLA d'enchainement. Ceci correspond à l'activation d'un des monômes de ce PLA. Le monôme activé, génère l'état suivant de l'algorithme.

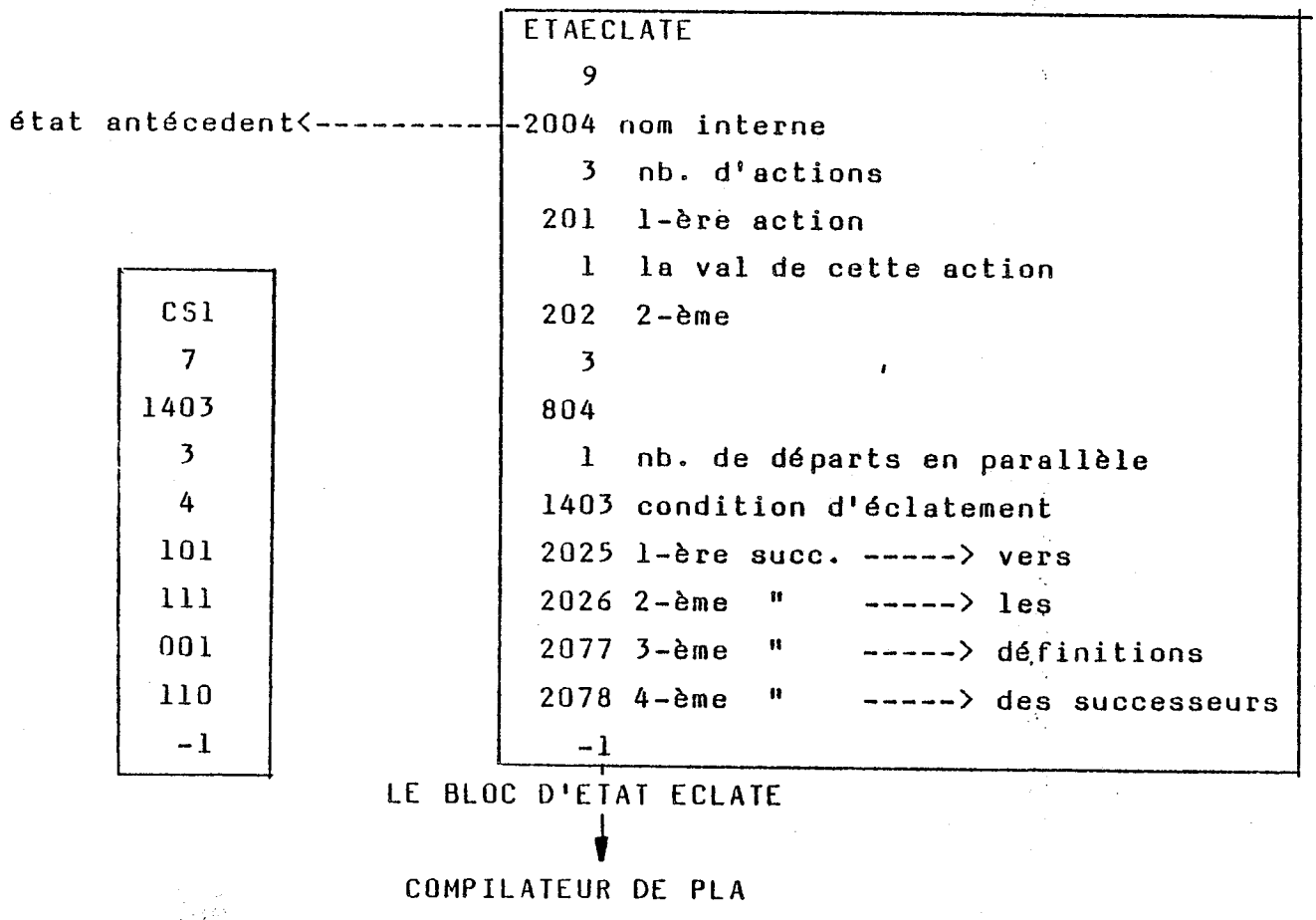
12-3-Exemple d'une séquence



12-4-Branchements

Les blocs d'état avec une condition de transition du type simple ou complexe correspondent aux états éclatés dans l'algorithme d'interpretation. Un état avec une condition de n valeurs, débute n séquences dont les états de départ sont déclarés dans la structure d'état éclaté.

12-5-Exemple d'un état éclaté



remarque 1:

La n-ième valeur de la condition choisit le n-ième successeur de l'état éclaté.

remarque 2:

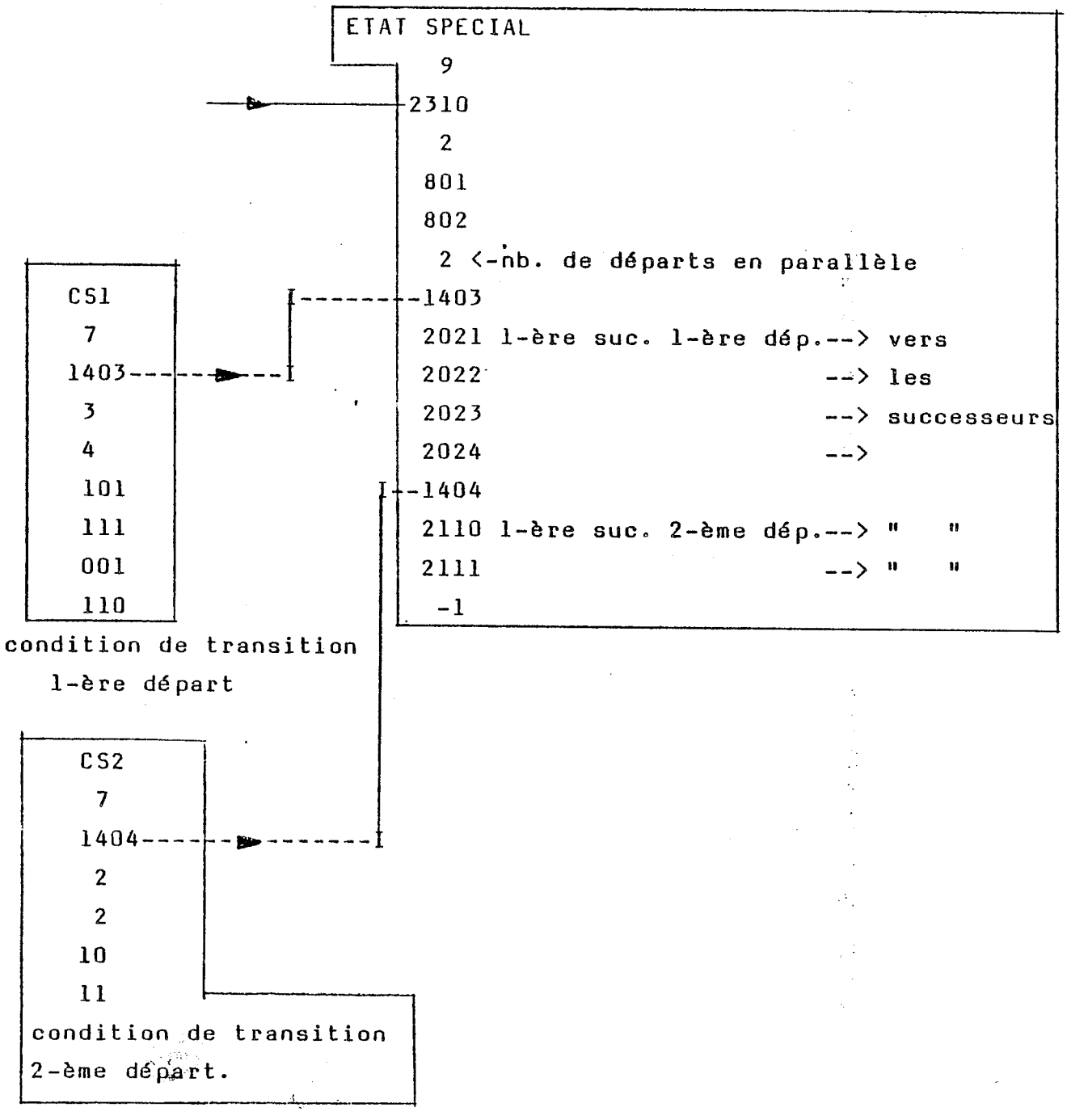
chacun des états successeur (2025, 2026, ..etc.) débute une séquence

12-6-Déclaration de Parallèlisme

Le degré de parallèlisme est donné par le nombre de départs en parrallèle, décrits dans chaque bloc d'état. En cas de départs en parrallèle, deux monômes seront activés en même temps. Ces dernièrs peuvent représenter un état éclaté avec plusieurs successeurs. La réalisation matérielle correspond à deux automates travaillant en parallèle, ceux-ci peuvent etre dèclanchès à partir d'un tel bloc d'état.

La version courante du compilateur ignore le parallèlisme et ne génère qu'un seul automate multi-PLA. Les extensions futures vers le parallèlisme sont prévisibles à travers la structure des blocs d'états.

12-7-Exemple


ETAT SPECIAL
9
2310
2
801
802
2 <-nb. de départs en parallèle
1403
2021 1-ère suc. 1-ère dép.--> vers
2022 --> les
2023 --> successeurs
2024 -->
1404
2110 1-ère suc. 2-ème dép.--> " "
2111 --> " "
-1
CS1
7
1403
3
4
101
111
001
110
condition de transition
1-ère départ
CS2
7
1404
2
2
10
11
condition de transition
2-ème départ.

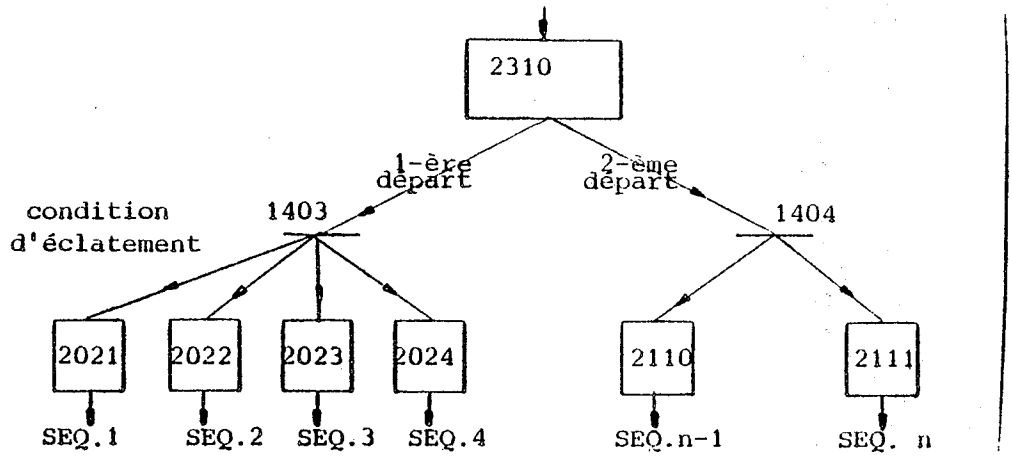


Figure 6-5-Schéma d'un état éclaté avec 2 départs en parallèles

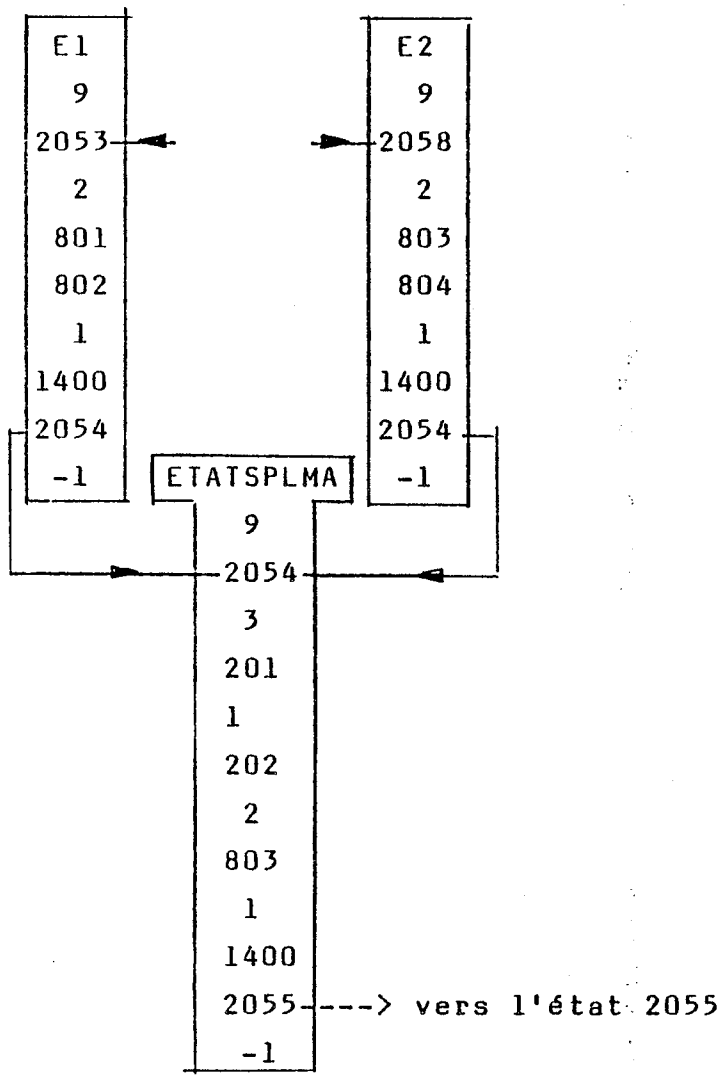
****

12-8-Etat simple multi-antécédents

Un cas particulier est le cas où des chemins secondaires se terminent au milieu d'un chemin pricipal. Dans ce cas, l'état de fin de séquence est du type simple. L'exemple suivant montre un tel cas.

12-9-Exemple

branche
principale



12-10-Fonctionnement détaillé du PLA d'enchainement et du compteur

L'architecture d'une telle structure est presentée par la figure 6-6 où on distingue la coopération de deux éléments (PLA et compteur) pour assurer le séquencement.

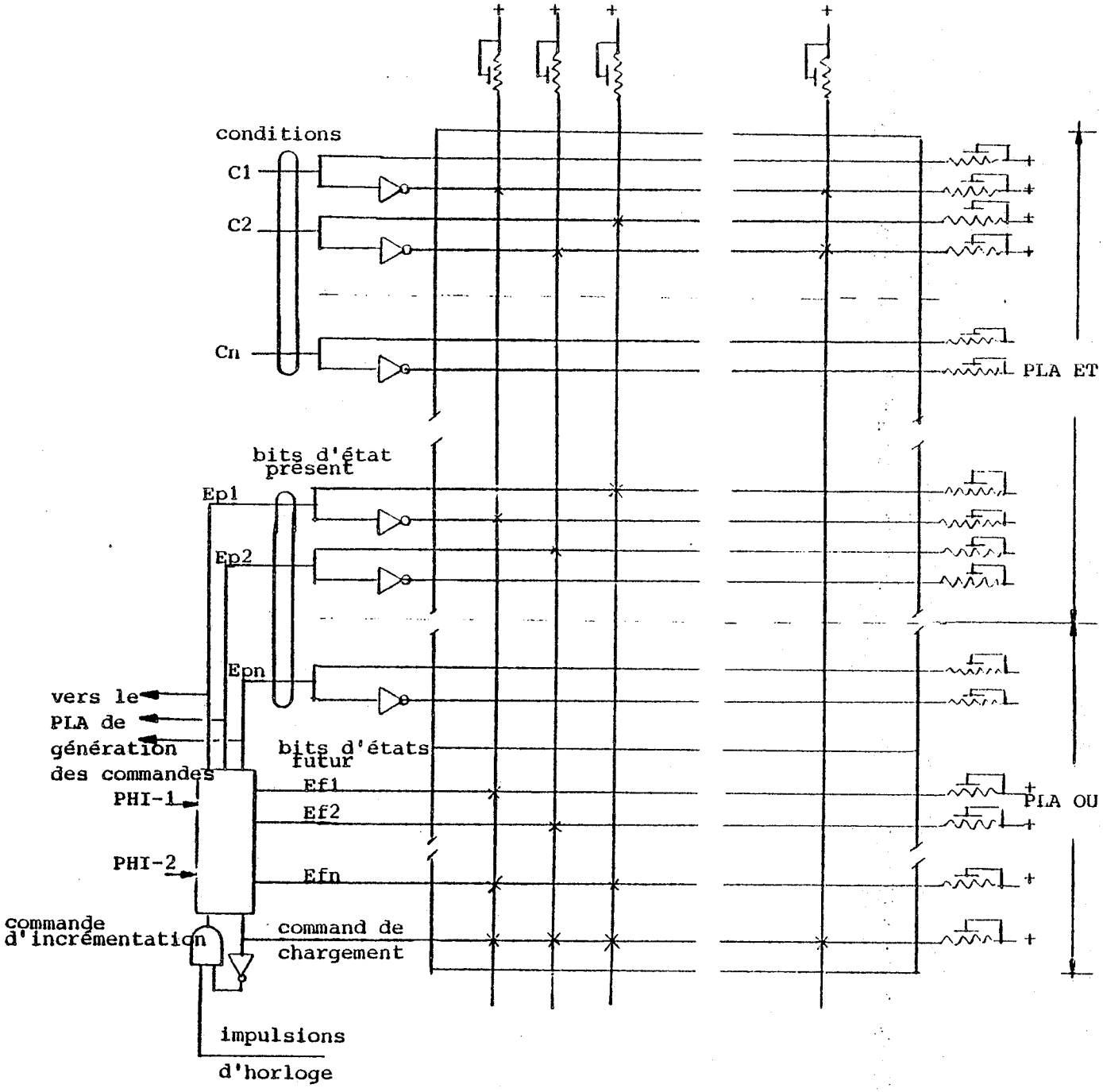


Figure 6-6 structure de séquencement

****

L'état suivant est chargé dans le compteur par le PLA d'enchainement (début d'une séquence). Le compteur énumère tous les états de cette séquence. Alors, le codage des états doit correspondre a la nature du compteur. En cas d'utilisation de compteurs binaires, un codage linéaire doit-être respecté. L'énumeration des états continue jusqu'au moment où l'état énuméré est un état éclaté. Comme le PLA ET peut détecter tous les états

éclatés, alors le PLA d'enchainement reprend le séquencement et force le compteur avec un nouvel état. On appelle cycle d'énumération des branches.

La réalisation automatique du PLA d'enchainement selon ce fonctionnement nécessite un classement descedant pour réaliser les routines du compilateur.

1-Il faut dissocier l'algorithme en branches.

2-Identifier les types des états.

3-coder les états.

4-Implanter les PLA d'enchainement et génération selon:

-l'architecture envisagée.

-les codes des états.

-les conditions.

C'est ainsi qu'un état éclaté avec "4" départs, déclenche 4 branches. la figure 6-7, indique une telle dissociation de l'algorithme pour l'état éclaté de 12-5.

branchel	branche2	branche3	branche4	
n1	n2	n3	n4	<--longueur de seq.
2004	2004	2004	2004	<--état éclaté
1403	1403	1403	1403	<--cond. de trans.
1	2	3	4	<-no. de val.cond.
2025	2026	2077	2078	<-1-ère état de seq.
"	"	"	"	
"	"	"	"	
état éclaté	état éclaté	état éclaté	état éclaté	<-état terminaison

Figure 6-7 dissociation d'un éclatement en branches

****

Chacune des branches de la figure 6-7, contient autant de données qu'en génère les monômes du PLA d'enchainement. Les nx (longueur de seq.) permet d'envisager les zones d'adresses pour les séquences et éventuellement les coder dans l'espace des groupes. Un codage qui considère la nature incrémentale du compteur utilisé (en cas d'un compteur binaire). Le nom interne et le numéro de la valeur de condition d'éclatement permet d'implanter la matrice ET

du PLA d'enchainemnt selon les configurations des bits déclarés dans la structure de condition. Enfin, le nom interne de l'état éclaté et son code sont utilisés pour implanter la partie ET du PLA d'enchaînement. Le code du premier état de séquence sera implanté dans la partie OU de ce PLA.

12-11-Cas particuliere des branches

Deux cas particuliers de branches peuvent être distingués:
1- Un état éclaté qui reboucle sur lui même. Dans ce cas-là, l'un des successeurs de l'état courant, est l'état lui-même ce qui sera interprêté par une branche de séquence de longueur zéro. La réalisation matérielle correspond à la génération d'un monôme dans le PLA d'enchainement. Ce monôme détecte et génère l'état rebouclé par les deux parties ET et OU du PLA.

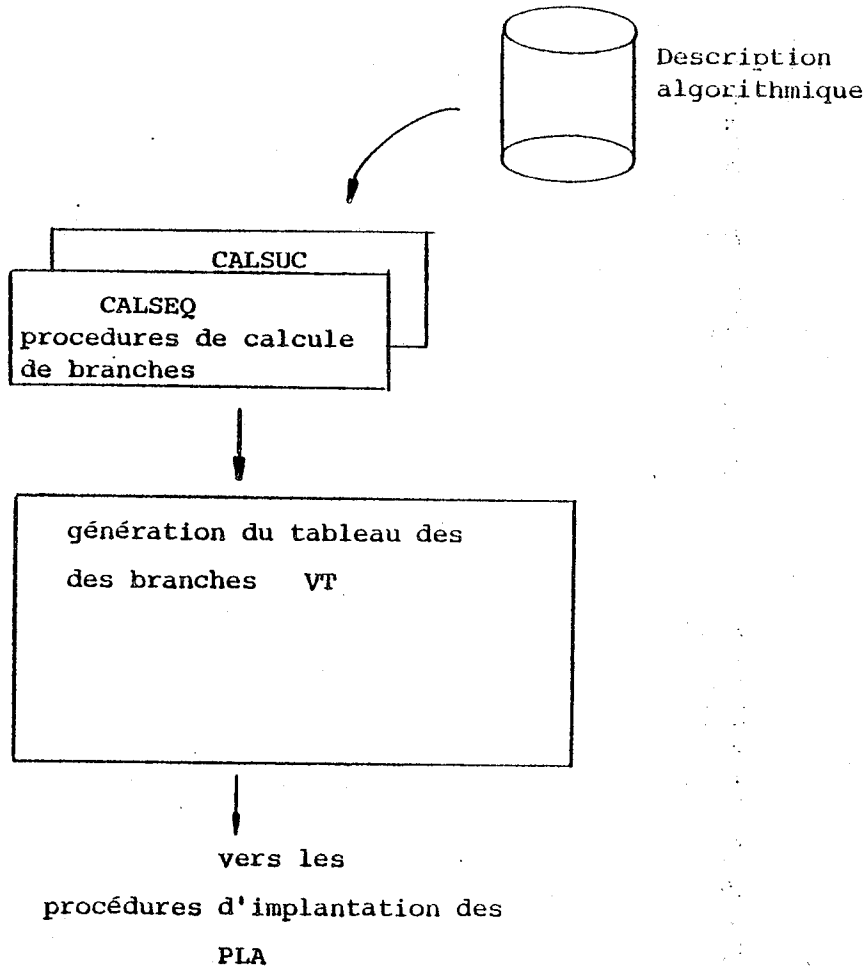
2- Les branches secondaires: Ce sont les branches dont les états de fin de séquence appartiennent à une autre branche. Le programme génère un monôme qui respecte la succession des séquences secondaires.

12-12-Les procedures de dissociation de l'algorithme d'interpretation

Deux procédures sont chargées de dissocier l'algorithme d'interpretation des instructions en branches qui permettront d'assembler le PLA d'enchainement. Ces procédures remplissent un tableau bidimesionel: VT du type entier. Dans chaque colonne de ce tableau, on trouve les donnees nécessaires pour générer un PLA d'enchainement. Le codage des états est ainsi déduit de ce tableau. Chaque élément de ce tableau est de la forme VT[L,M], où L énumère les éléments de chaque branche et M énumère les branches. Chaque case X d'une colonne M (VT[X,M]) comprend les données suivantes:

VT[0,M]	no. de valeur de la condition de transition
VT[1,M]	=1 longueur de la séquence
VT[2,M]	nom interne de l'état éclaté
VT[3,M]	nom interne de la condition de transition
VT[3+1,M]	1-ier état nom interne des états simples
VT[3+2,M]	2-ième état constituant une séquence de longueur 1.
VT[3+1,M]	1-ième état
VT[11,M]	adresse du premièr état dans le nappe de micro-adresses
VT[12,M]	groupe " " "

remarque: Les deux dernières cases de chaque branche sont résérvées pour les coordonnées du 1-èr état de chaque séquence dans le tableau de codage des états.



Génération du tableau VT

****

12-13-Génération des specifications du PLA d'enchainemnt

Aprés avoir codé les états, tous les renseignements nécessaires sont disponibles pour générer le PLA d'enchainement. On distingue deux types de spécifiactions pour generer ce PLA.
1-Specifications, concernant l'implantation des transistors de chaque monôme. Celle-ci se divisée en deux catégories:

1-1 L'implantation du monome dans la partie ET, ce qui comprend la détéction de la valeur de la condition et du code d'état.

2-1 Génération de l'état futur dans la matrice OU.

2-Specification concernant la génération des monômes suplémentaires pour les branches secondaires de l'algorithme d'interpretation.

La procedure GENMATENCH, partant des tableaux existants génére un autre tableau: TPLA qui comprend les specifications concernant le PLA d'enchainement. La figure 6-8 indique les tableaux employés pour la génération des specifications du PLA d'enchainement.

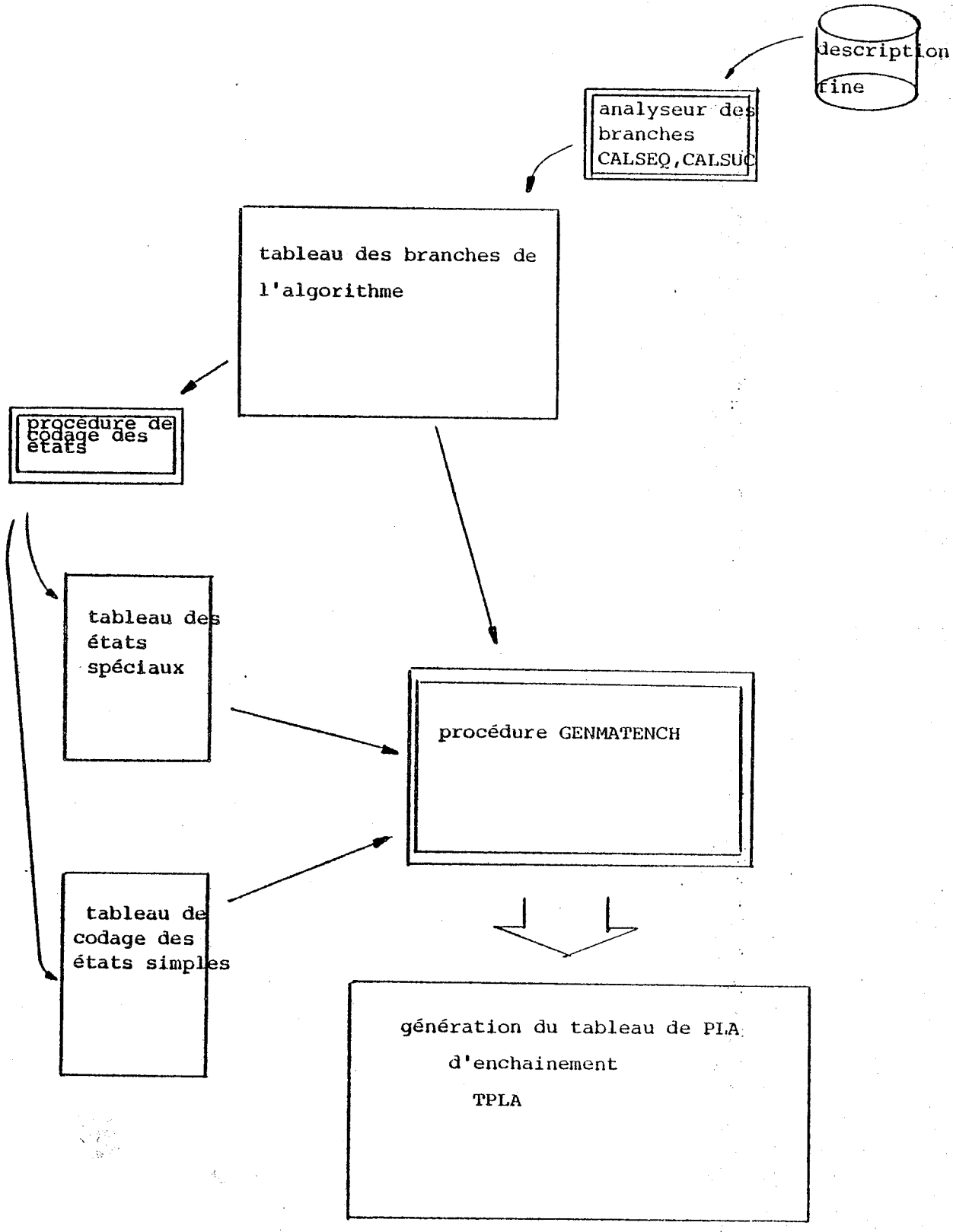


Figure 6-8 Génération des specifications du PLA d'enchainement

****

12-14-Structure du tableau du PLA d'enchainement

Les specifications de chaque monôme par lesquels on peut implanter un PLA d'enchainement sont les suivantes:

1-PLA ET	Champ du tableau
1-Nom interne de l'état détécté par le PLA ET	ETAET
2-Adresse de cet état	ADRET
3-Groupe de cet état	GRET
4-Nom interne de condition d'éclatement	NICND
5-La valeur de cette condition qui autorise le branchement	VC

2-PLA OU

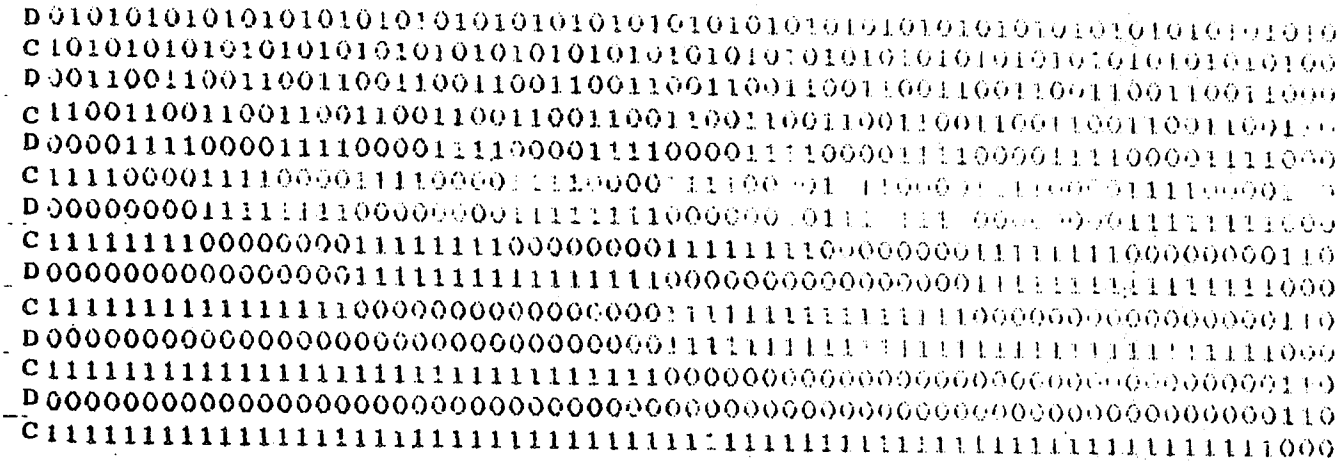
1- Nom interne de l'état forçé au compteur	ETAOU
2- Adresse de cet état	ADROU
3- Groupe de cet état	ADRET

Ces variables sont les champs du tableau d'entiers TPLA[MON].

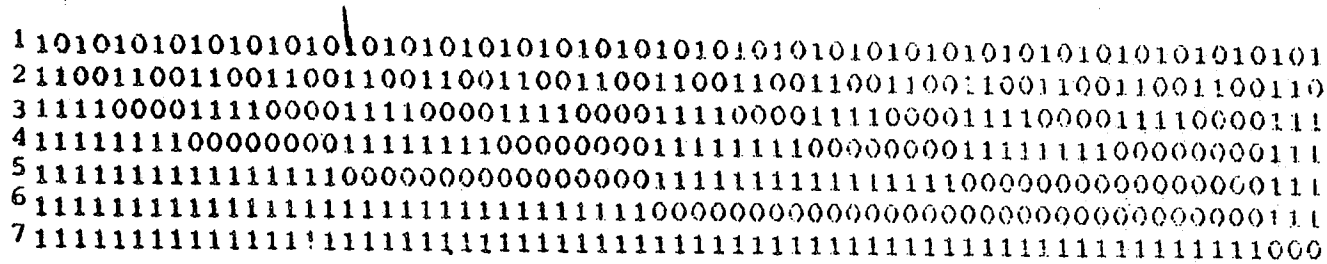
12-15-La génération des codes des états sous forme binaire

La procédure GENMATBASE, est chargée de générer deux matrices ET et OU contenant le codage binaire des entiers: de 0 à nb.max des groupes (si nb. de groupes est supérieur à nb. d'adresses). En effet ces deux matrices constituent un convertisseur de code adapté, dont le numéro de colonne correspond à la valeur décimale du code et l'implantation de cette colonne dispose le configuration des bits en équivalent binaire. Comme deux aspects: la détéction du code d'état (detection des bits directs et complémentés) et sa génération doivent être considérées, alors deux matrices de base només MATET1 et MATOU1 sont générées. La matrice MATET1 est une matrice de détéction des valeurs binaire dont le numéro de colonne correspond à l'équivalent décimale de cette valeur. La deuxième matrice MATOU1 met en valeurs binaire ses lignes de sortie. Cette valeur est équivalent décimal de son numéro de colonne. La figure 6-9 présente l'implantation des deux matrices

MATET1 et MATOU1.



Implantation de la matrice de base MATET1



Implantation de la matrice de base MATOU1

Figure 6-9 Implantations des matrices de base .

****

12-16-Implantation du PLA d'enchainement

La procedure IMPLMATENCH, est chargée d'afficher les deux matrices ET et OU du PLA d'enchainement. Partant du tableau TPLA et des specifications des conditions d'éclatements elle affiche l'implantation des monômes sur l'écran ou sur l'imprimante. Elle peut également construire un fichier, qui servira en entrée du

système PAOLA

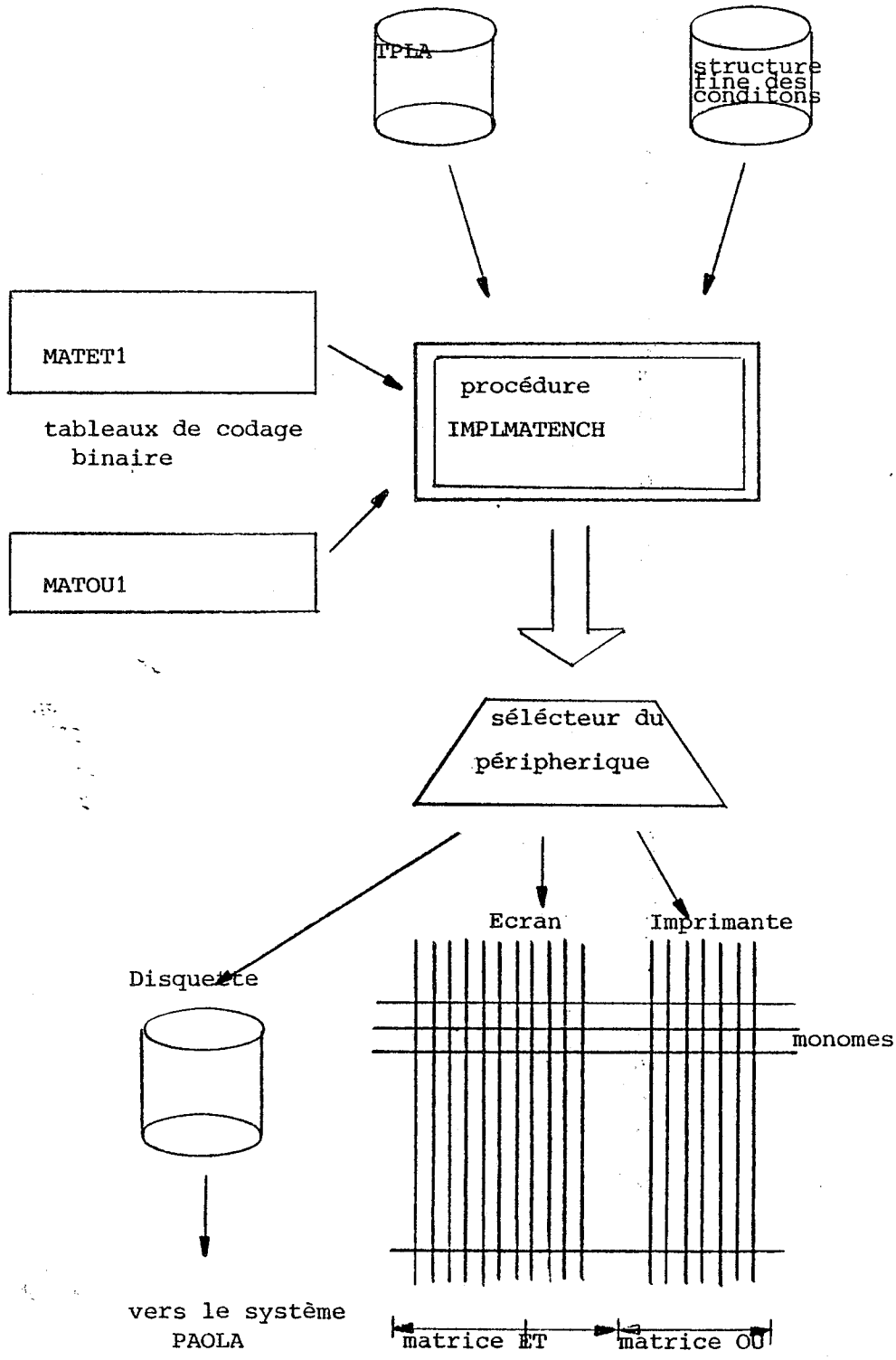


Figure 6-10-Affichage du PLA d'enchainement

****

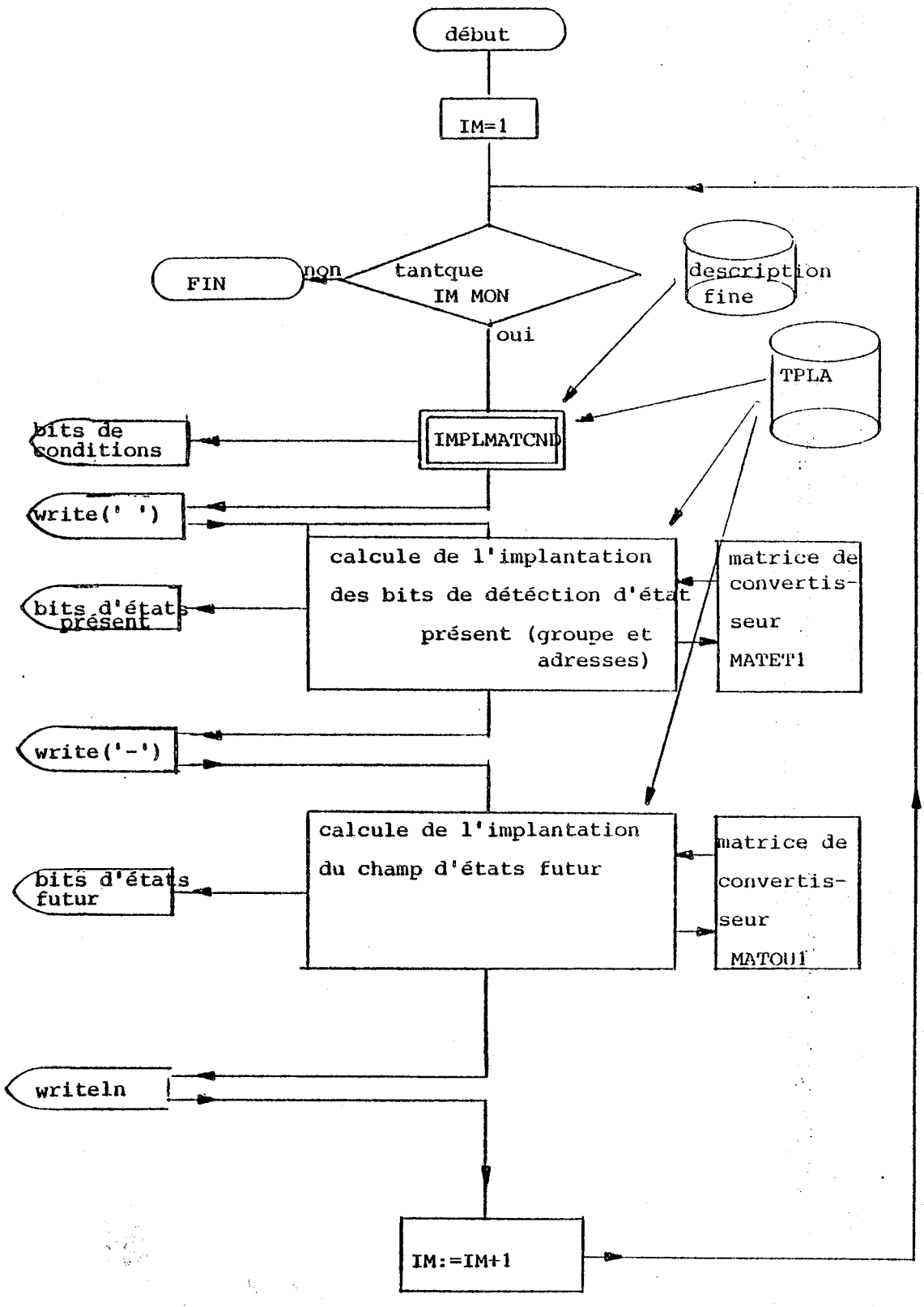


Figure 6-11-L'algorithme simplifié de la procedure IMPLMATENCH

****

Dans cet algorithme la boucle principale tantque IM<MON, énumère de 1 à "nombre maximum" des monômes les specifications du PLA d'enchainement. Le tableau TPLA, fourni les données nécessaires à la procédure IMPLMATENCH. Les differentes parties de chaque monômes (détection de condition, détection d'état présent et l'état suivant) sont générées sur le péripherique concérné.

12-17 Affichage des matrices

La procédure GEF est chargée de effectuer un aiguillage pour envoyer le contenu de deux matrices sur des peripheriques disponibles du système de microordinateur. Il s'agit soit d'affichage sur l'écran du terminal ou l'imprimente pour l'étape de verification manuelle, soit de stockage permanent sur la disquette d'utilisateur. Le fichier généré sert d'interfaçe entre le système PAMELA et l'optimisateur topologique PAOLA.

12-18-Génération du PLA de génération des commandes

Le tableau des entiers TPLAGC est destiné à enregistrer la specification du PLA de génération des commands. Les differents champs de ce tableau sont:

specification du PLA ET	
nom interne de l'état detecté par monôme	NOMIN
adresse de cet état	AD
groupe de cet état	GRP
specification du PLA OU	
pointeur par lequel on accède à la structure fine de l'état	POINT
longueur de zone d'actions dans le bloc d'état NOMIN	LACT

Comme chaque bloc d'état comprend le nom interne et la valeur des actions la variable POINT permet d'acceder à la description fine des actions activés par chaque état.

12-19 La procédure d'affichage du PLA de génération des commandes

Partant du tableau TPLAGC, la procédure IMPLAGCOM, affiche les matrices du PLA de génération des commandes. Les monômes de ce PLA sont générés ligne par ligne, en énumèrant le tableau TPLGC. La figure 6-12 présente l'algorithme de cette procédure.

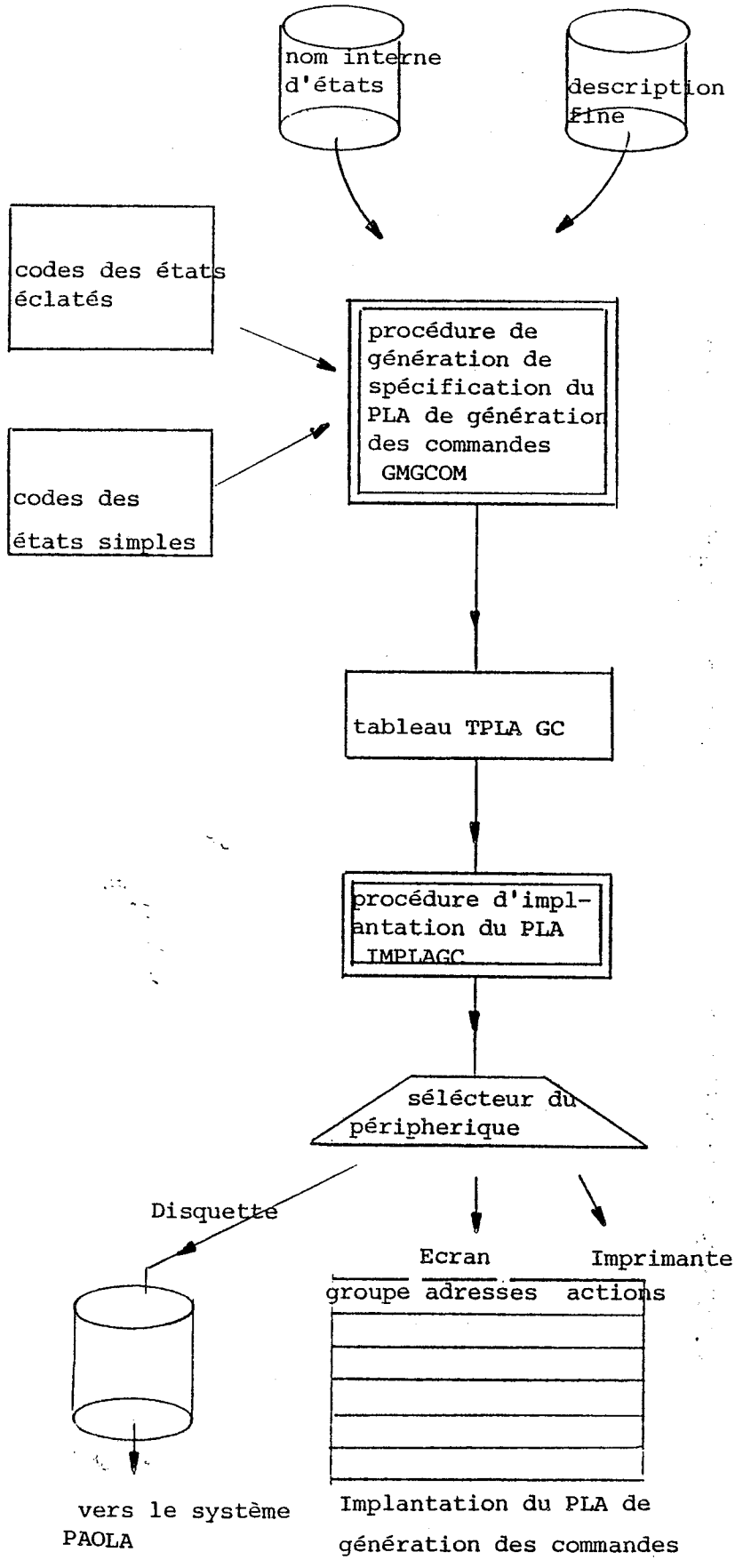


Figure 6-12-structure globale d'affichage du PLA de génération des commandes

****

CAHPITRE VII

APPLICATION DU COMPILATEUR DES PLA

AU CAS DU MICROPROCESSEUR

MC6800

1-Introduction

Le micrprocesseur MC6800 est choisi comme exemple pour tester l'efficacité de cet outil.

Le MC6800 est un microprocesseur 8 bits, paru sur le marché depuis 1974. Le circuit est réalisé en technologie NMOS, 6 microns GSN3, occupant une surface de 18.7 mm 2. La partie operative restant inchangée, les PLA de la partie contrôle sont générés par le compilateur. L'algorithme d'interpretation est celui du MC6800 original et l'évaluation de la surface est aussi basée sur la technologie utilisée dans le MC6800 original. Les architectures réalisées sont celles qui sont décrites dans le troisième chapitre.

2-1 Spécification

La Figure 7-1 montre le brochage de ce microprocesseur. Quarante broches relient le processeur au monde exterieur. On distingue 8 plots pour les données, 16 plots pour les adresses. Les plots R/W, DBE et VMA s'occupent de la gestion du bus. Les entrées complémentées NMI,IRQ et HALT sont les ordres supérieur qui contrôlent le processeur. Les deux signaux complémentaires PHI-1 et PHI-2 véhiculent les impulsions pour synchroniser le processeur.

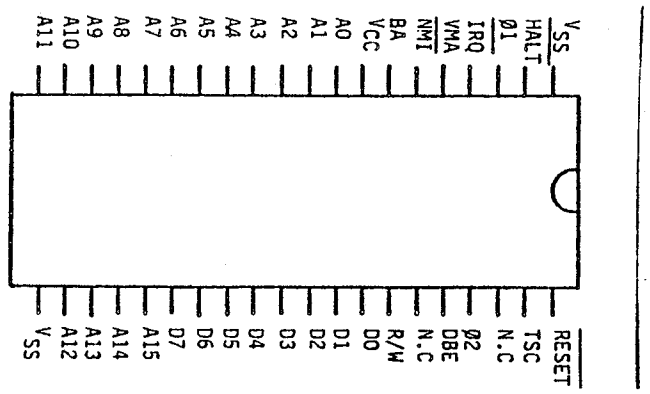


Figure 7-1- Brochage du MC6800

****

2-2 Architecture interne (basée sur une étude de M.Obrebska [OBR 82]ët M.NEMMOUR [NEM 79])

Ce microprocesseur se découpe en deux blocs: partie contrôle et partie opérative. Les deux parties ont une interface nette qui permet de les distinguer. Un ligne découpe le circuit en deux parties dont la partie contrôle sera générée par le compilateur de PLA, tandis que la zone de la partie opérative reste inchangée. La photo de la puce de ce microprocesseur est presentée par la figure 7-2. Les differents blocs sont presenté par la figure 7-3.

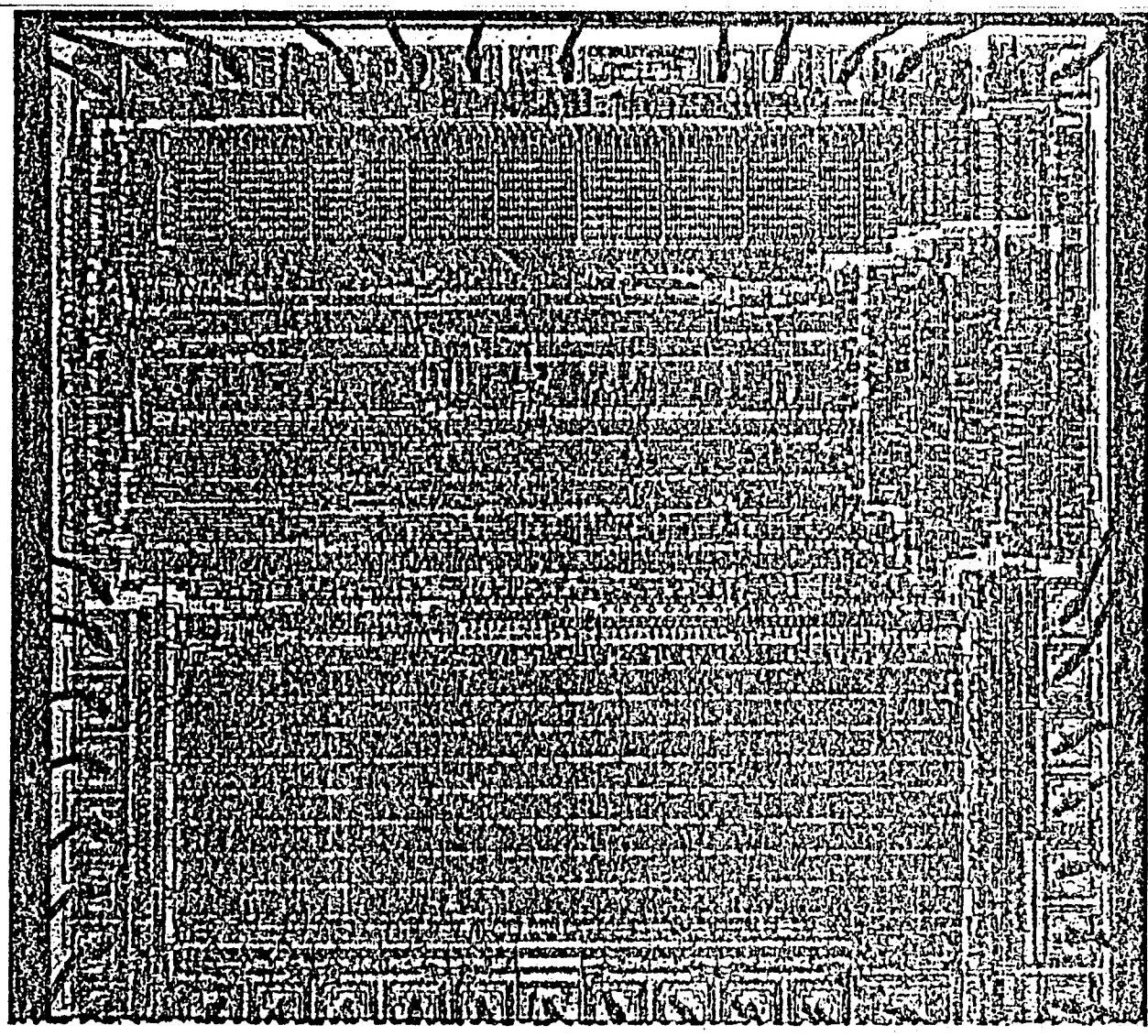


Figure 7-2-La photo du MC6800

****

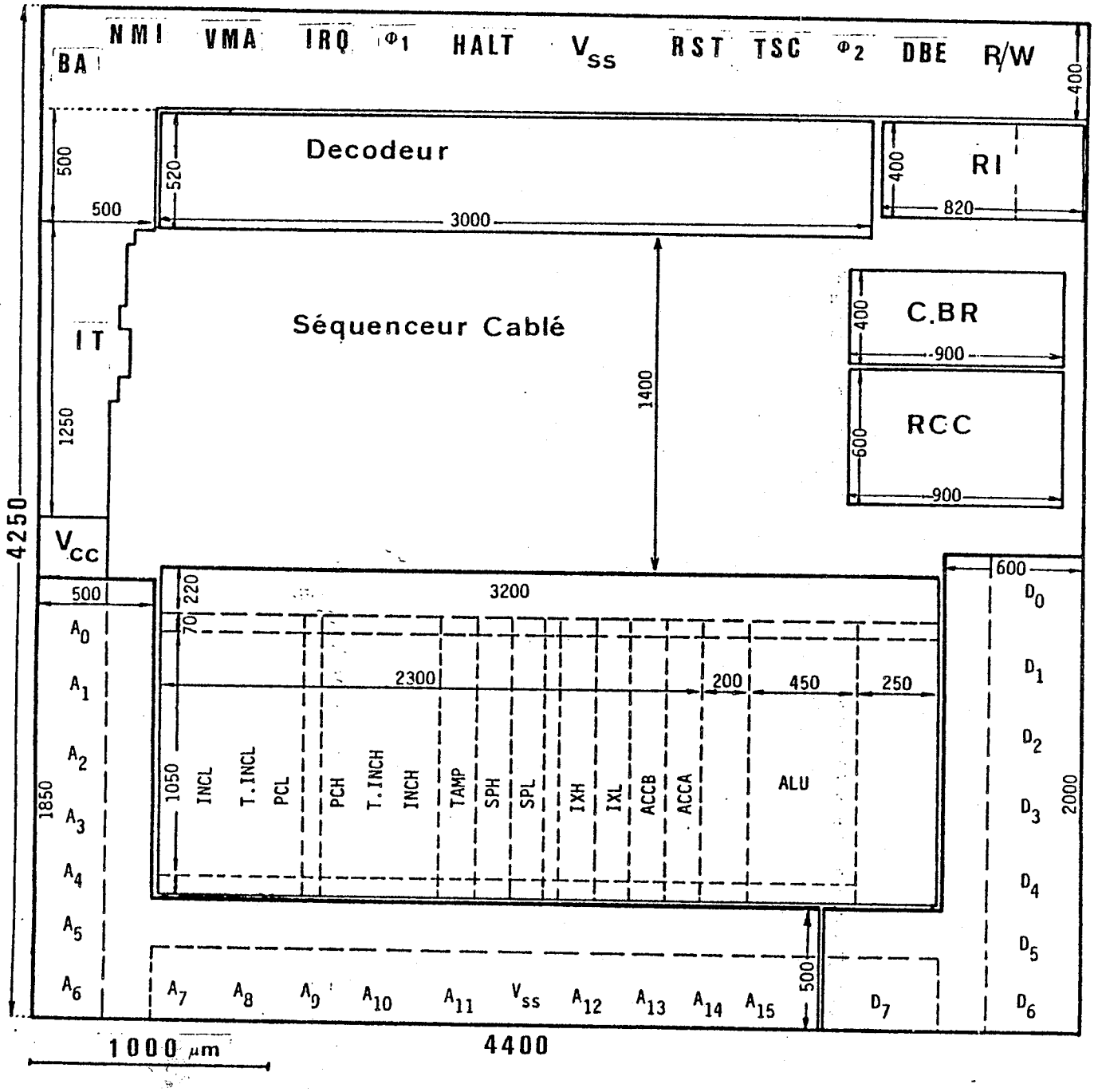


Figure 7-3-Topologie du MC6800 extraite de [OBR 82]

****

2-3 Organisation de la partie opérative

La partie opérative présente une structure trés régulière, constituée d'un ensemble de registres de 8 bits:

-un compteur ordinal formé de deux parties PCL (8 bits) et PCH (8 bits)

-un registre index (IXL et IXH, 2X8 bits)

-un registre temporaire (TEMP 8 bits)

-Deux registres d'accumulateurs (ACCA et ACCB)

Une UAL de 8 bits est employée pour la manipulation de données deux incrémenteurs-décrémenteurs (INCL et INCH, 2X8 bits) éffectuent les opérations sur les adresses. La figure 7-4 presente le schéma fonctionnel de la partie opérative de ce microprocesseur.

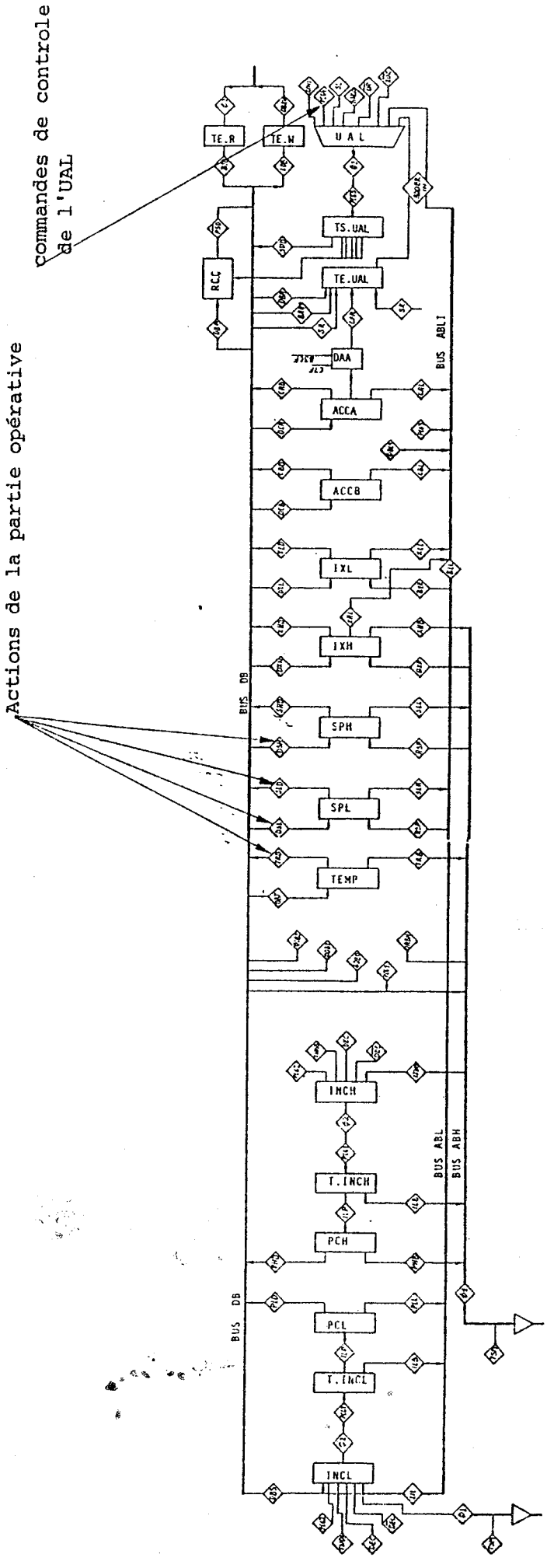


Figure 7-4- Organisation de la partie opérative du MC6800
Extraite de [OBR 82]

****

2-4 Partie contrôle

La partie contrôle réalisée en logique anarchique. On distingue un décodeur d'instruction et un séquenceur cablé. Les blocs suivant constituent la partie contrôle:

- Registre Instruction RI (8 bits)
- registre de code condition RCC (6 bits)
- Circuit de calcul de condition de branchement CBR
- éléments de mémorisation et de validation pour les signaux: RST',HALT', NMI', IRQ', TSC et DBE. on trouve aussi l'ensemble de circuit de BA, VMA et R/W.

2-5 Strcucture du séquenceur

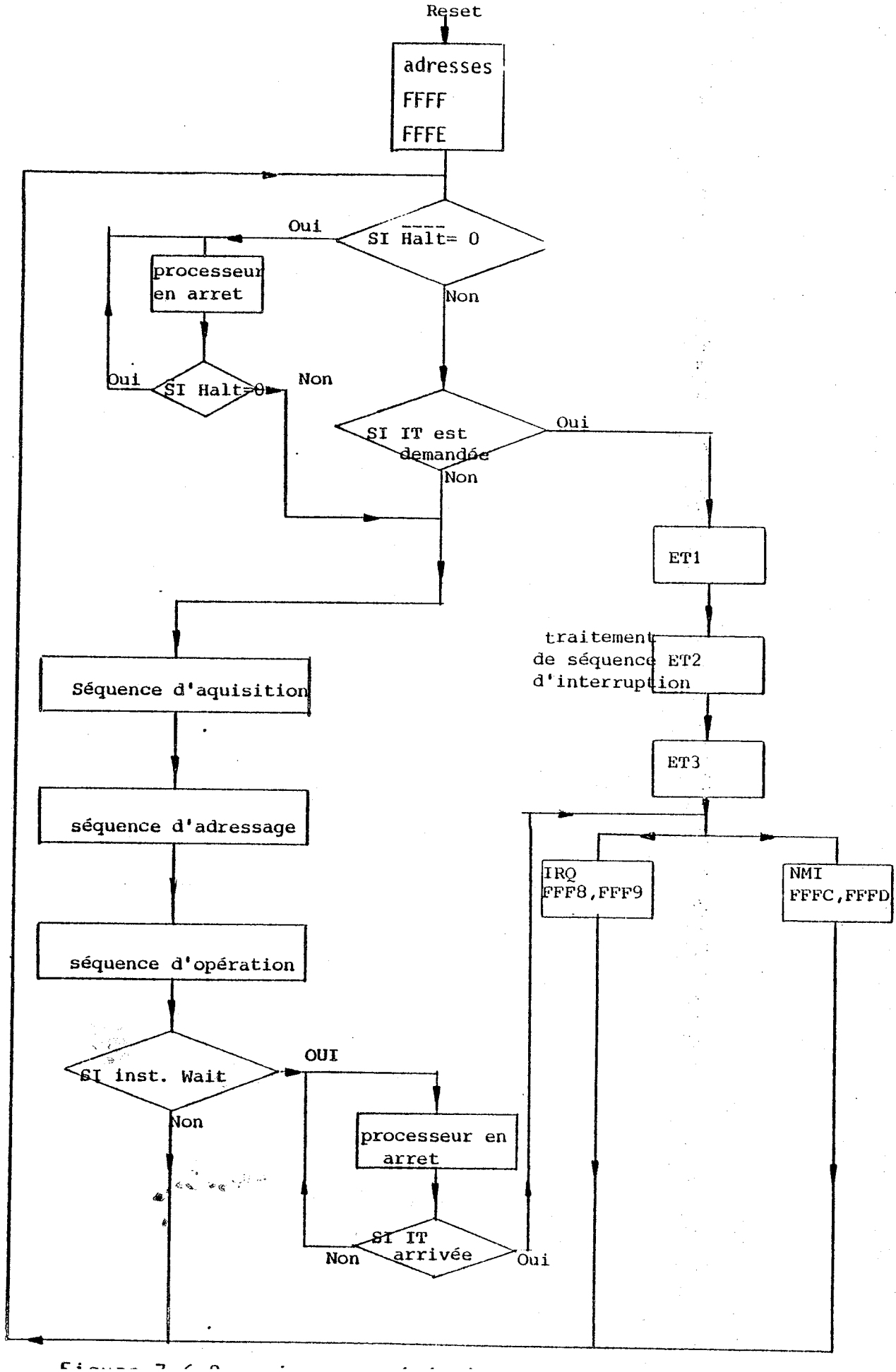
La partie contrôle du MC6800 est réalisé à l'aide des registres à décalages qui constituent plusieurs automates chargés d'interpréter le code opération.

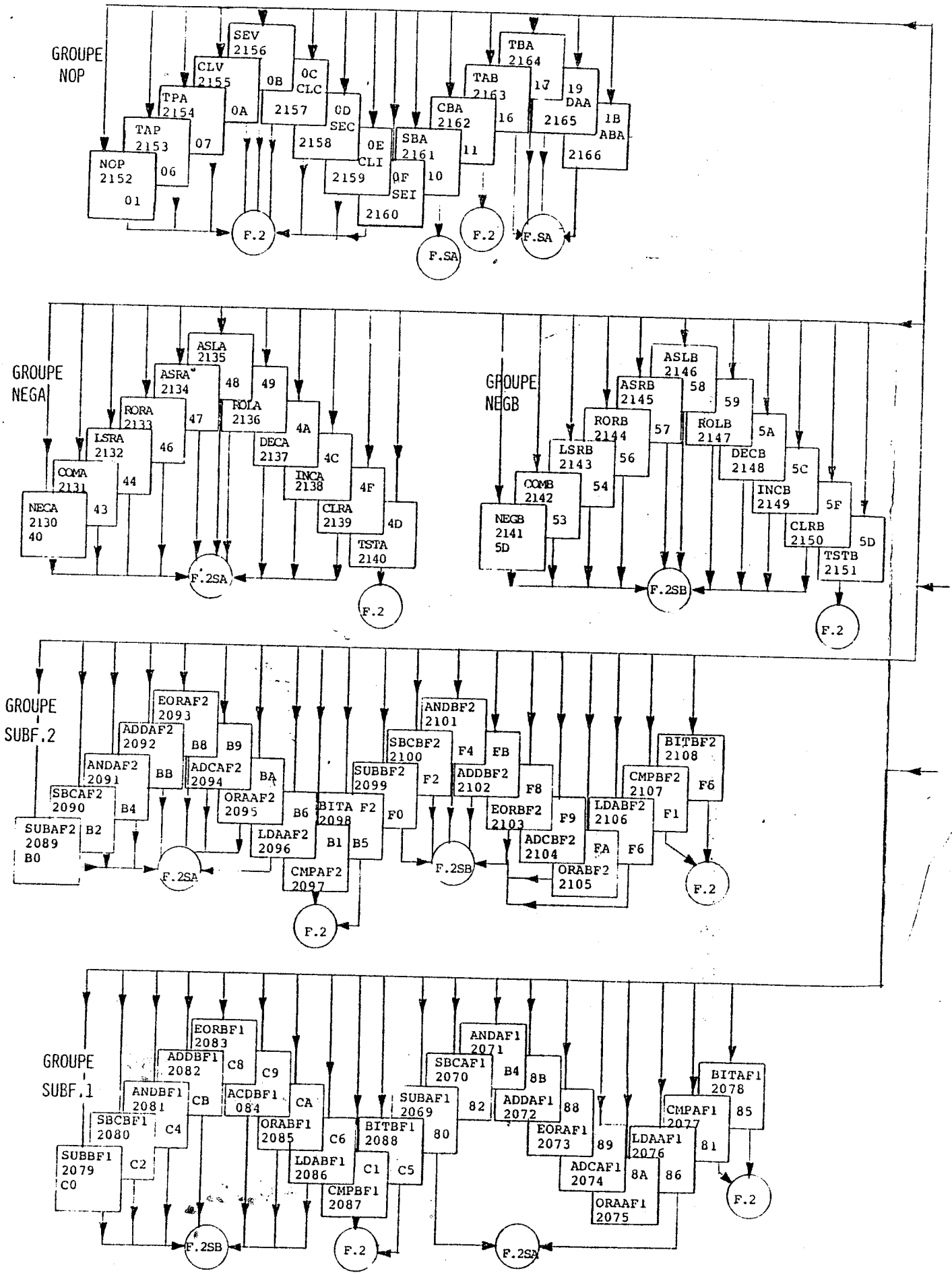
L'algorithme suivant (figure 7-6) est parcouru par le séquenceur. Les signaux de contrôle imposent les tests nécessaires pour guider l'algorithme. L'initialisation (activation: RESET'=0) amène le processeur aux adresses FFFE et FFFF. Partant d'une séquence d'initialisation pintée par ces deux adresses, on teste l'état HALT'. En cas de mise en Halt, l'algorithme est conduit vers l'état Halt où on en sort pas si la broche Halt est activée. Si Halt est inactive, on teste le bit d'interruptions IT', par lequel le traitement de séquences d'interruption est lancé. Dans le cas ou IT'=HALT'=1 (inactive), le traitement des codes opération commence. L'execution de l'instruction en cours se divise en trois séquences:

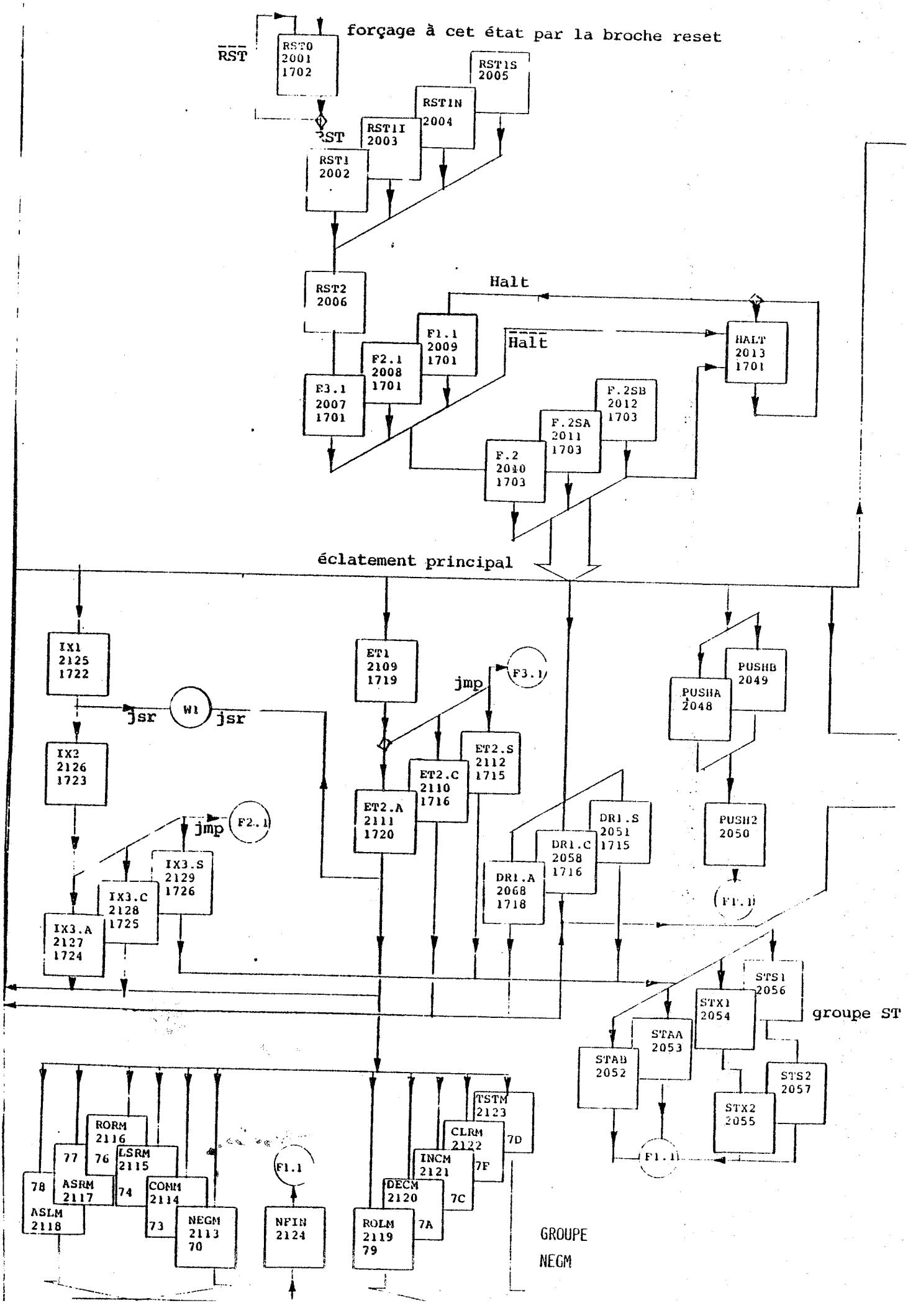
1- Aquisition
2- Adressage
3- Opération

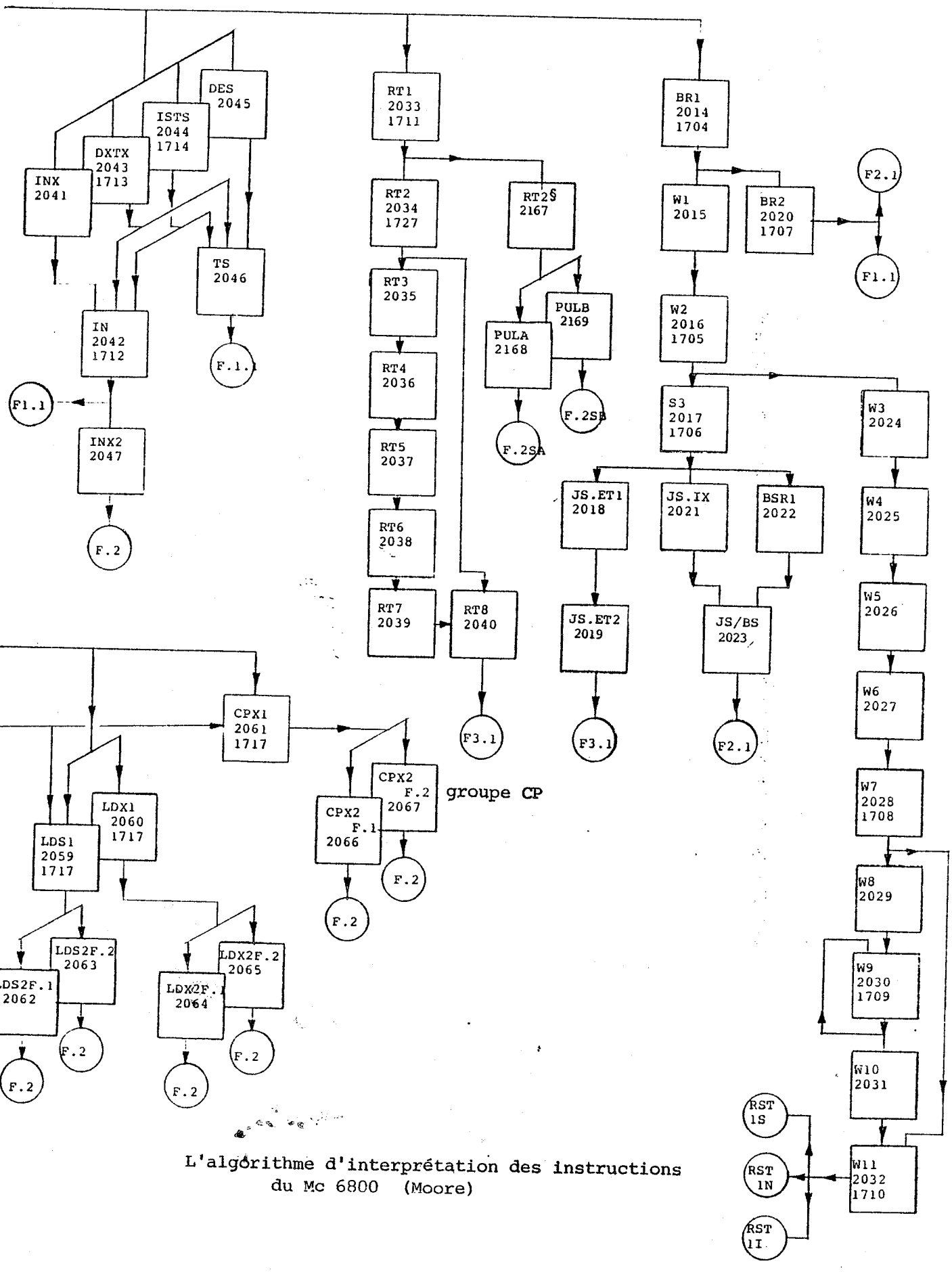
L'instruction WAIT est testée à la fin du traitement du code opération. En cas de demande de WAIT et en absence d'interruptions, le processeur est en état d'arrêt. L'arrivée d'interruptions conduit le processeur vers la fin de traitement des interruptions

IRQ et NMI. Le processeur est forcé à l'adresse FFF8 et FFF9 pour IRQ et FFFC et FFFD pour une demande d'interruption non-masquable (NMI).


Reset
adresses
FFFF
FFFE
SI Halt= 0
Oui
Non
processeur
en arret
Oui
SI Halt=0
Non
SI IT est
demandée
Oui
Non
ET1
traitement
de séquence
d'interruption
ET2
ET3
Séquence d'aquisition
séquence d'adressage
séquence d'opération
IRQ
FFF8,FFF9
NMI
FFFC,FFFD
SI inst. Wait
OUI
Non
processeur en
arret
SI IT
arrivée
Non
Oui


GROUPE NOP
NOP 2152 01
TAP 2153 07
TPA 2154 0A
CLV 2155 0B
SEV 2156
2157
0C CLC
2158
0D SEC
2159
0E CLI
2160
0F SEI
SBA 2161 11
CBA 2162 16
TAB 2163 17
TBA 2164
2165
19 DAA
2166
1B ABA
F.2
F.SA
GROUPE NEGA
NECA 2130 40
COMA 2131 44
LSRA 2132 46
RORA 2133 47
ASRA 2134 48
ASLA 2135
ROLA 2136 49
DECA 2137 4A
INCA 2138 4C
CLRA 2139 4F
TSTA 2140 4D
F.2SA
F.2
GROUPE NEGB
NEGB 2141 5D
COMB 2142 53
LSRB 2143 54
RORB 2144 56
ASRB 2145 57
ASLB 2146 58
ROLB 2147 59
DECB 2148 5A
INCB 2149 5C
CLRB 2150 5F
TSTB 2151 5D
F.2SB
F.2
GROUPE SUBF.2
SUBAF2 2089 B0
SBCAF2 2090 B2
ANDAF2 2091 B4
ADDAF2 2092 BB
EORAF2 2093 B8
ADCAF2 2094 B9
ORAAF2 2095 BA
LDAAF2 2096 B6
CMPAF2 2097 B1
BITA F2 2098 B5
SUBBF2 2099 F0
SBCBF2 2100 F2
ANDBF2 2101 F4
ADDBF2 2102 FB
EORBF2 2103 F8
ADCBF2 2104 F9
ORABF2 2105 FA
LDABF2 2106 F6
CMPBF2 2107 F1
BITBF2 2108 F6
F.2SA
F.2SB
F.2
GROUPE SUBF.1
SUBBF1 2079 C0
SBCBF1 2080 C2
ANDBF1 2081 C4
ADDBF1 2082 CB
EORBF1 2083 C8
ACDBF1 084 C9
ORABF1 2085 CA
LDABF1 2086 C6
CMPBF1 2087 C1
BITBF1 2088 C5
SUBAF1 2069 80
SBCAF1 2070 82
ANDAF1 2071 B4
ADDAF1 2072 8B
EORAF1 2073 88
ADCAF1 2074 89
ORAAF1 2075 8A
LDAAF1 2076 81
CMPAF1 2077 86
BITAF1 2078 85
F.2SB
F.2
F.2SA
F.2


forçage à cet état par la broche reset
RST
RST0 2001 1702
RST1S 2005
RSTIN 2004
RST1I 2003
RST1 2002
RST2 2006
Halt
Halt
F1.1 2009 1701
F2.1 2008 1701
E3.1 2007 1701
HALT 2013 1701
F.2SB 2012 1703
F.2SA 2011 1703
F.2 2010 1703
éclatement principal
IX1 2125 1722
jsr
W1
jsr
IX2 2126 1723
jmp
F2.1
IX3.S 2129 1726
IX3.C 2128 1725
IX3.A 2127 1724
ET1 2109 1719
jmp
F3.1
ET2.S 2112 1715
ET2.C 2110 1716
ET2.A 2111 1720
PUSHB 2049
PUSHA 2048
PUSH2 2050
DR1.S 2051 1715
DR1.C 2058 1716
DR1.A 2068 1718
F1.1
STS1 2056
STX1 2054
STAA 2053
STAB 2052
STS2 2057
STX2 2055
F1.1
groupe ST
RORM 2116
77
76
LSRM 2115
78
ASRM 2117
74
COMM 2114
ASLM 2118
73
NEGM 2113 70
F1.1
NFIN 2124
TSTM 2123
7D
CLRM 2122
7F
INCM 2121
DECM 2120
7C
ROLM 2119 79
7A
GROUPE
NEGM



L'algorithme d'interprétation des instructions
du Mc 6800 (Moore)

2-7 La recherche de l'instruction

La recherche de l'instruction prend toujours 2 cycles d'horloges, dont le premièr cycle correspond à la lecture du code opération en mémoire et le second à son décodage. Pendant ce deuxième cycle, on en profite pour lire l'octet suivant dans un registre temporaire IE.R. En effet on gagne du temps dans les instructions qui utilisent la valeur de deuxième octet comme opérande ou comme une partie d'adresse.

Le signal RESET', conduit l'algorithme à l'état RST0. En fait, dans la réalisation cablée du séquenceur du MC6800, les registres à décalages sont remis à zero par contre dans la réalisation basée sur les PLA, l'adresse de l'état reset est forcée par un circuit externe. Celui-ci est commandé par la broche d'entrée RESET'. L'état RST1 dans la figure 6 correspond en réalité à quatre états differents de l'algorithme d'interprétation. La lecture des adresses suivantes s'effectue de la manière décrite ci-dessous.

-l'état RST0, alors on lit l'octet à l'adresses FFFE dans l'état RST1
-detection de NMI, " " "" FFFC dans l'état RST1I
-detection de IRQ, " " " FFF8 " "
RST1N
-detection de SWI, " " " FFFA " "
RST1S

La mise en arrêt du process est realisée par l'activation du signal HALT' qui met en haute impédance les plots 3 états et positionne les signaux BA et VMA. Par conséquent l'algorithme est conduit vers l'état HALT.

L'éclatement principal prend en compte les instructions concérnées par l'adressage direct, indexé et étendu.

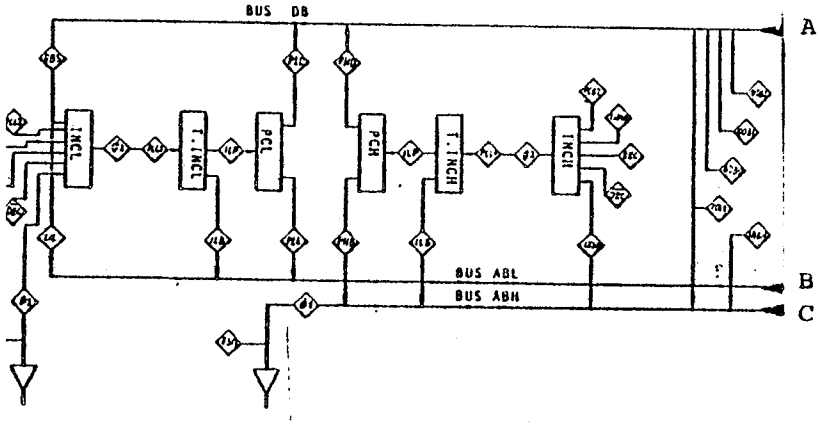
L'explication détaillée de chaque état de l'algorithme se trouve dans la thèse [OBR 82] et le rapprot [NEM-79]

3-1 Description algorithmique textuelle de MC6800 (MOORE)

Le texte est décrit en annexe 1 où l'on trouve la présentation de l'algorithme à travers des définitions du langage intermèdiaire. Les deux phases P1 et P2 sont déclarés d'avance. L'action simple "AS-CARRY" comprend la déclaration des commandes de contrôle de carry entrant dans l'UAL. Ainsi que le "AS-UAL" met en fonction l'UAL. La partie opérative du MC 6800 commandée par ces actions est presenté dans la figure 7-7.

LES DIFFERNETS MODES DE L'UAL

Val.	I0	I1	I2	Mode
1ère	0	0	1	par défaut ADD
2ème	1	0	1	AND
3ème	0	0	0	OR
4ème	1	1	1	SL



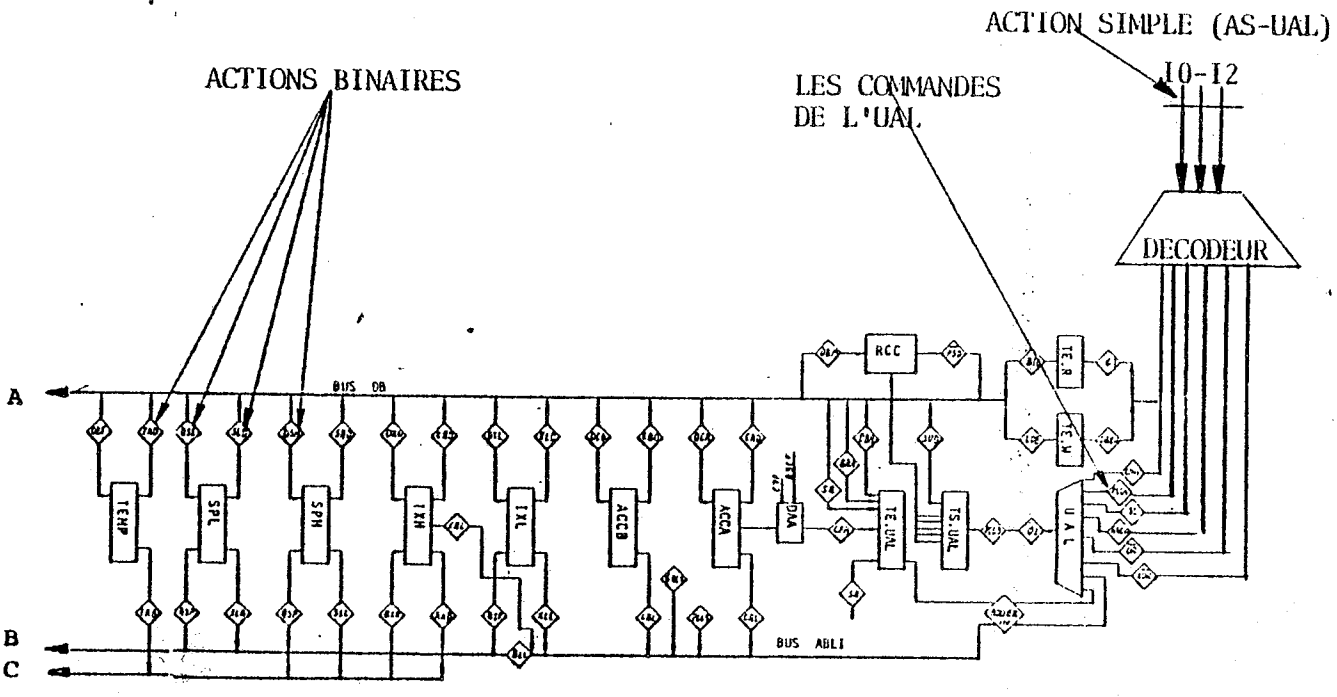


Figure 7-7-Presentation des actions simples et binaires

****

Les bits de commande sous formes d'actions binaires s'occupent

des noms internes [801 : 847]. Les conditions (RESET, HALT,...etc), sont decrites par l'intermèdiaire des conditions simples. Les termes de produits des bits sont presentés apres dans la déclaration des conditions complexes. Ces conditions occupent les noms internes de [1401 : 1413]. Parmi ces conditions, on distingue la condition RI avec un nom interne 1403 qui présente le code opération. Les conditions complexes mettent ensemble les conditions simples afin de presenter les termes de produits des bits engagés dans l'éclatement des états de l'algorithme. Ces conditions sont déclarées sous les noms internes [1701 : 1728]. Les états de l'algorithme sont décrits à partir de nom interne 2001. L'état 2169 est la dernière étape de l'algorithme. Dans les blocs d'états, le nombre de départs en parallèle est 1. Les identificateurs utilisés dans l'organigramme de la figure 6 comme nom d'états, sont utilisés comme noms alpha-numériques dans la description textuelle. On distingue les états éclatés de F.2, F2S.A et F2S.B. Pour chacun de ces états, plusieurs états successeurs sont déclarés. La condition d'éclatement est de nom-interne 1703.

3-2 Les PLA générés par compilateur

La compilation de l'automate de MOORE de ce microprocesseur génére un PLA d'enchainement avec 499 monômes et un PLA de génération des commandes avec 169 monômes. La surface de PLA de génération est tellement creuse qu'elle occoupe 3/2 celle du microprocesseur. La comparaison des PLA avec le MC6800 original est démontré dans la figure 7-8. est

augmantation de surface 2/3

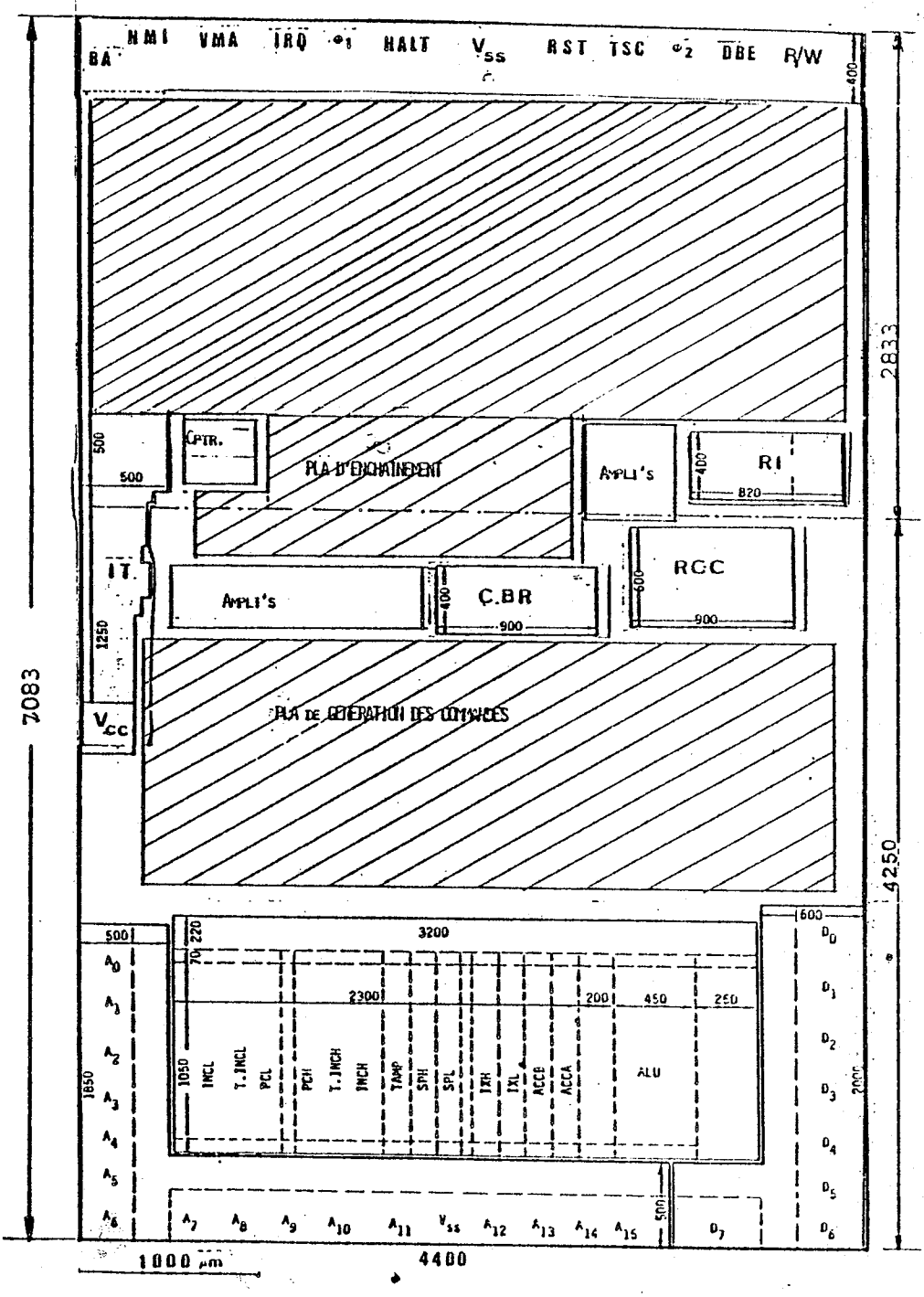
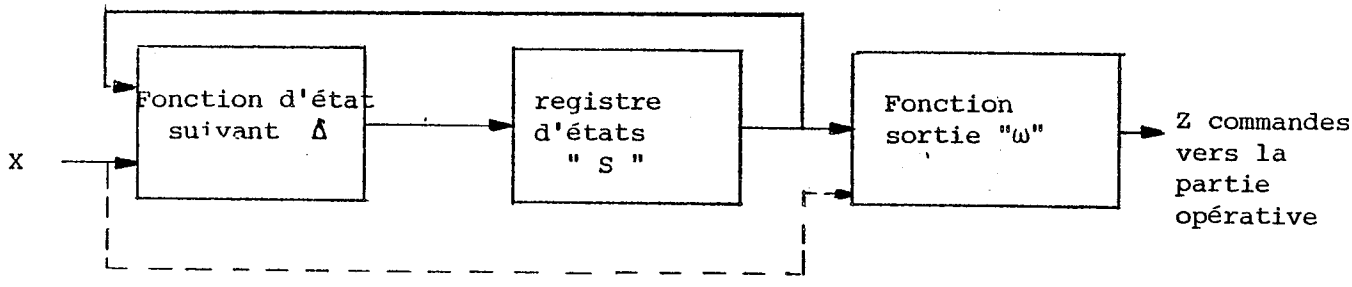


Figure 7-8-topologie du MC6800 (MOORE) généré par le compilateur

****

4-1 La réalisation Mealy de MC6800

Plusieurs architectures peuvent être considérées pour une même machine d'états finis. Un controleur du type multi PLA est modélisé par trois parties: La fonction de l'état suivant DELTA, la fonction de sortie OMEGA et le registre d'états (S). La figure 7-8 présente la réalisation MOORE d'une machine d'états finis.



1- X: Les paramétres d'entrées et comptes rendus

2- Les pointillées, fournissent une réalisation Mealy

Figure 7-8- présentation générale des machines d'états fini

****

La sortie Z de fonction, génére les séquences des mots de contrôle vers la partie opérative. L'entrée X est connectée à la source des parametres provenant du monde exterieur. Pour une machine MOORE , S est utilisée comme un argument pour la fonction OMEGA. Pour la machine MEALY, on utilise S et X comme arguments. Les pointillés de la figure 7-8 représentent le passage des parametres pour réaliser une machine de MEALY.

Le résultat non-acceptable de la réalisation MOORE du MC6800, peut être amélioré grace à la paramétrisation des états sous forme Mealy. Les états ayant le même profil de branchements pêuvent être presentés par un même état dans l'algorithme d'interpretation. Cet aspect est presenté par les réctangles superposés de la figure 7-6.

Le premier état de chaque ensemble peut être utilisé dans l'enchainement de l'algorithme et grâçe au conditionnement des actions on peut les générer en utilisant le code opération.

Graçe aux actions conditionnelles, une telle conversion de l'automate de MOORE à son equivalent MEALY est réalisable.

4-2 L'algorithme d'interpretation de MEALY de MC6800

La nouvelle version du MC6800 sous forme de MEALY est présentée dans la figure 7-9. Les états avec un profil de branchement commun sont décrits. La description algorithmique M68MLY contient cette structure sous forme d'un fichier de texte et se trouve dans l'annnexe 2. A part les actions simples et binaires, les conditions aimples et complexes et les blocs d'états dans la réalisation MOORE, les actions conditonnelles sont ajoutées. Les états de recherche de l'instruction et les differents groupes de l'algorithme de la figure 6, sont simplifés et seulement 67 états sont utilisés dans l'enchainement de l'algorithme d'interpretation.

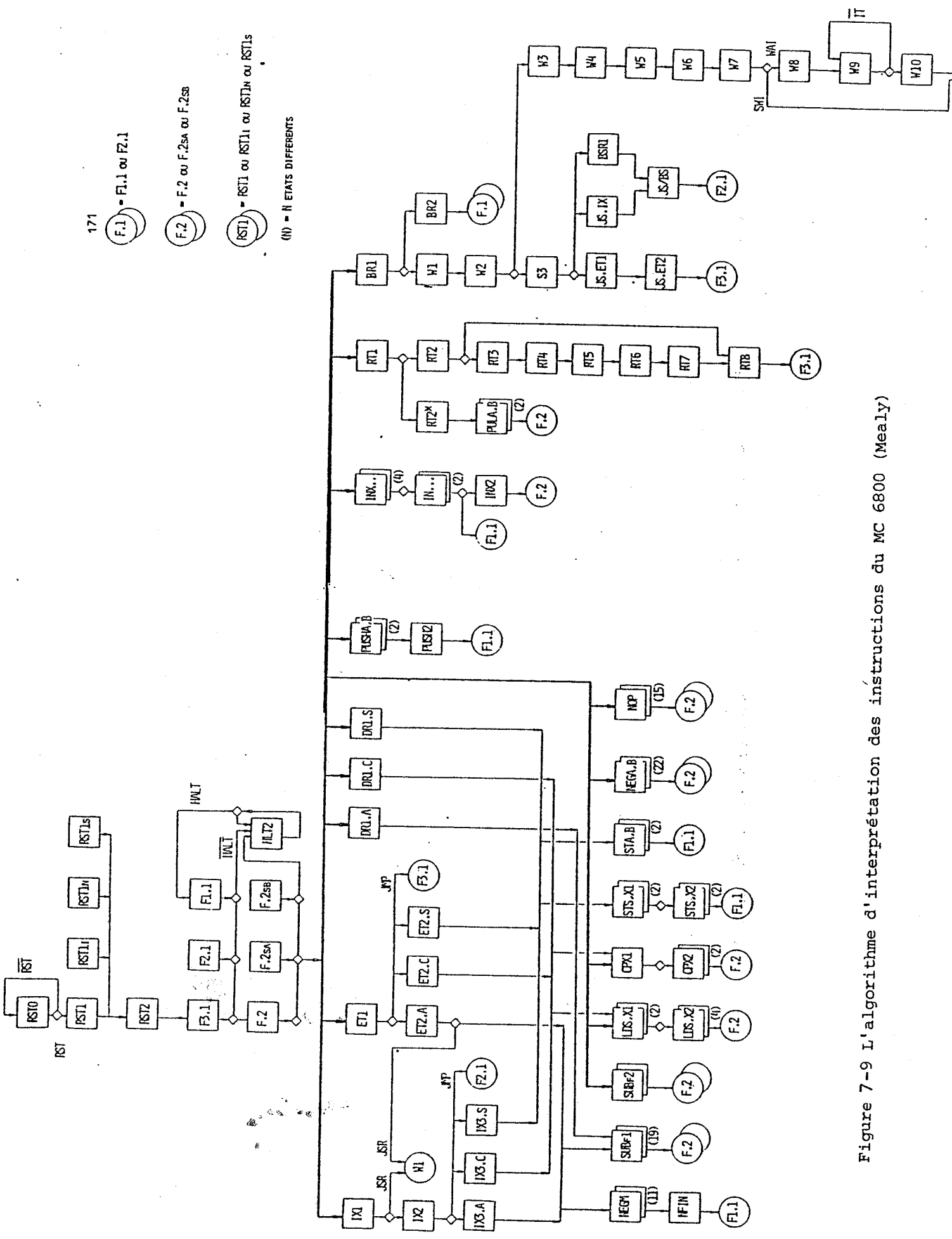


Figure 7-9 L'algorithme d'interprétation des instructions du MC 6800 (Mealy)

Parmi les actions conditionnelles, les groupes AC609-SUB, AC611, AC612, AC614 permet au compilateur de générer un PLA avec l'architecture decrite dans le chapitre III.

4-3 Resultats intermédiaires du compilateur

La structure du tableau des branches VT pour la réalisation MEALY est presenté dans l'annexe 3. La séquence le plus longue est constituée de 7 états. Celle ci est la séquence de traitement d'interruptions déclenchée par l'état éclaté "2034". La condition de transition pour nom-interne 1727. La première valeur autorise la transition. Le premier état de cette séquence 2035 a pour successeur 2036. Le dérnier état 2007 est un état éclaté qui termine cette séquence. Les spécification des autres branches de cet algorithme sont decrites. Les deux cases VT[11,M] et VT[12,M] précisent la position d'adresse et groupe de première état 2035 dans le tableau des codes d'états. Le tableau de codage des états est presenté par la figure suivante.

	ADRESSES DES ETATS -->							
	0	1	2	3	4	5	6	7
0	2035	2046	2047	2048	2049	2040	2007	2150
1	2024	2025	2026	2027	2028	2045	2046	2152
2	2048	2050	2010	2058	2059	2059	2052	2026
3	2061	2056	0	2002	2002	2018	2019	2047
4	2021	2023	0	2147	2148	2054	2055	2052
5	2113	2124	0	2041	2042	2015	2016	0
6	2022	0	2029	2030	2042	0	2041	0
7	2042	2037	2043	2044	2013	2051	2109	2125
8	2034	2001	0	0	2014	0	0	2110
9	2111	2112	0	2058	0	0	2126	2127

Figure 7-11- Le tableau de codage des états

****

Le tableau TPLA (Celui d'enchainement) comprend 71 monômes. Le PLA de génération des commandes comprend 169 monômes dont 71 monômes sont activés par les bits d'états et le reste est commandés par 16 bits directs et complémentés du code opération. L'identité des bits

est imprimé devant chaque ligne des PLA pour repérer les connexions des PLA avec leurs environement. Une verification simple des bits avec la description algorithmique originale met en évidence l'exactitude de cet outil de génération automatique.

L'implantation des transistors pour les deux PLA d'enchainement et de génération des commandes d'aprés le fichier source M68MLY est presenté par la suite.

Figure 7-12-Implantation des transistors du PLA d'enchainement

Figure 9-13-Implantation des transistors du PLA de génération des commandes

Figure 7-14 PLA's de génération des commandes activés par le code opération

Le PLA de génération des commandes est constitué de deux parites qui activent le PLA OU commun. Les entrées des bits d'états (14 bits) et les 16 bits de code condition activent les deux PLA ET.

4-4 Comparaison des résultats avec le MC6800 original

Génération des masques final à l'aide de systéme PAOLA nous permet de calculer la surface exacte des PLA aprés l'optimisation topologique. Les masques sont presentés par la suite et le plan de masse finale est mise en comparaison avec le topologie originale du MC6800. Une augmentation de 22 pourcent est nécesaire pour une génération automatique de ce microprocesseur.

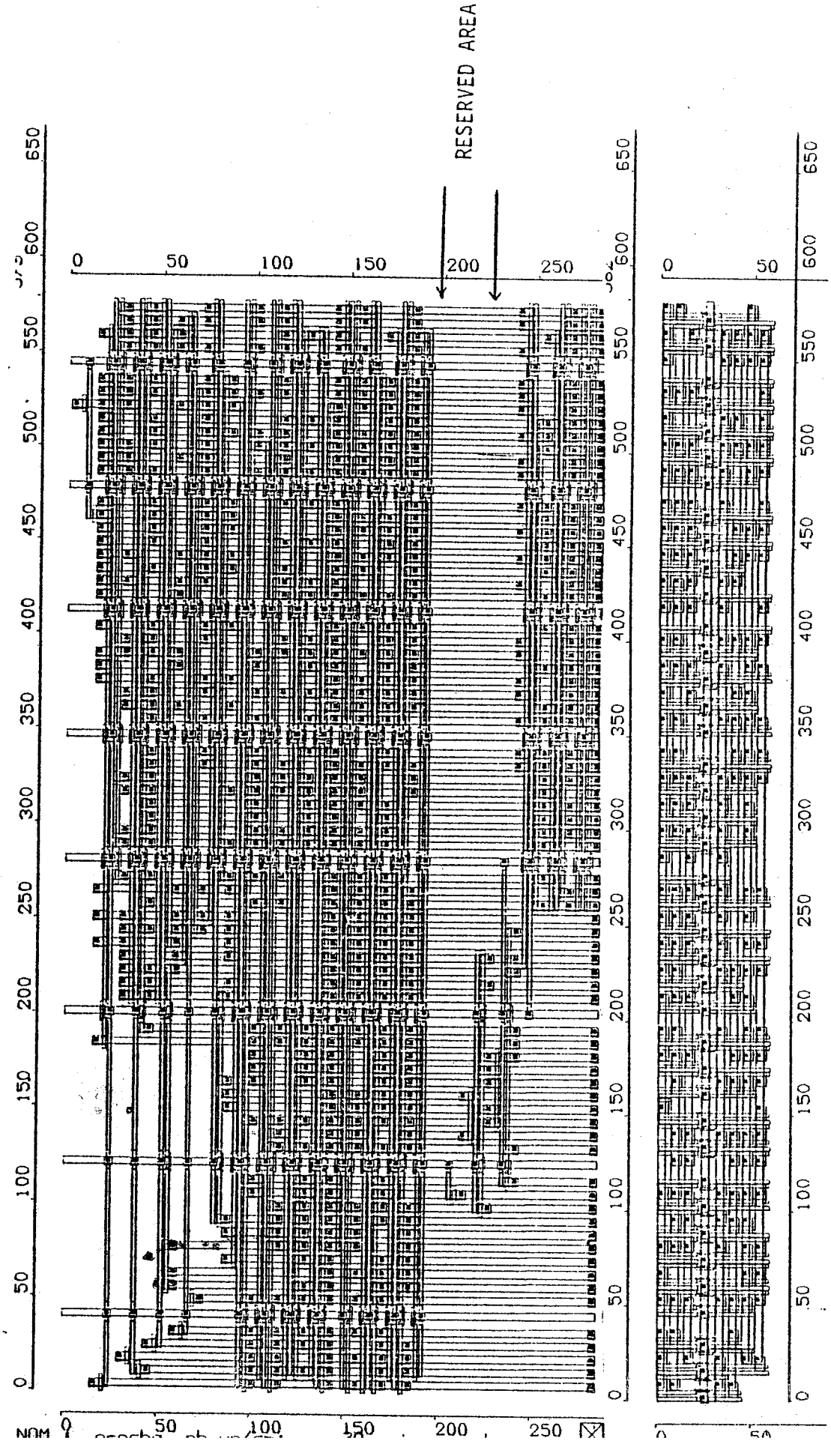


Figure- 7-15 Le masque du PLA d'enchainement

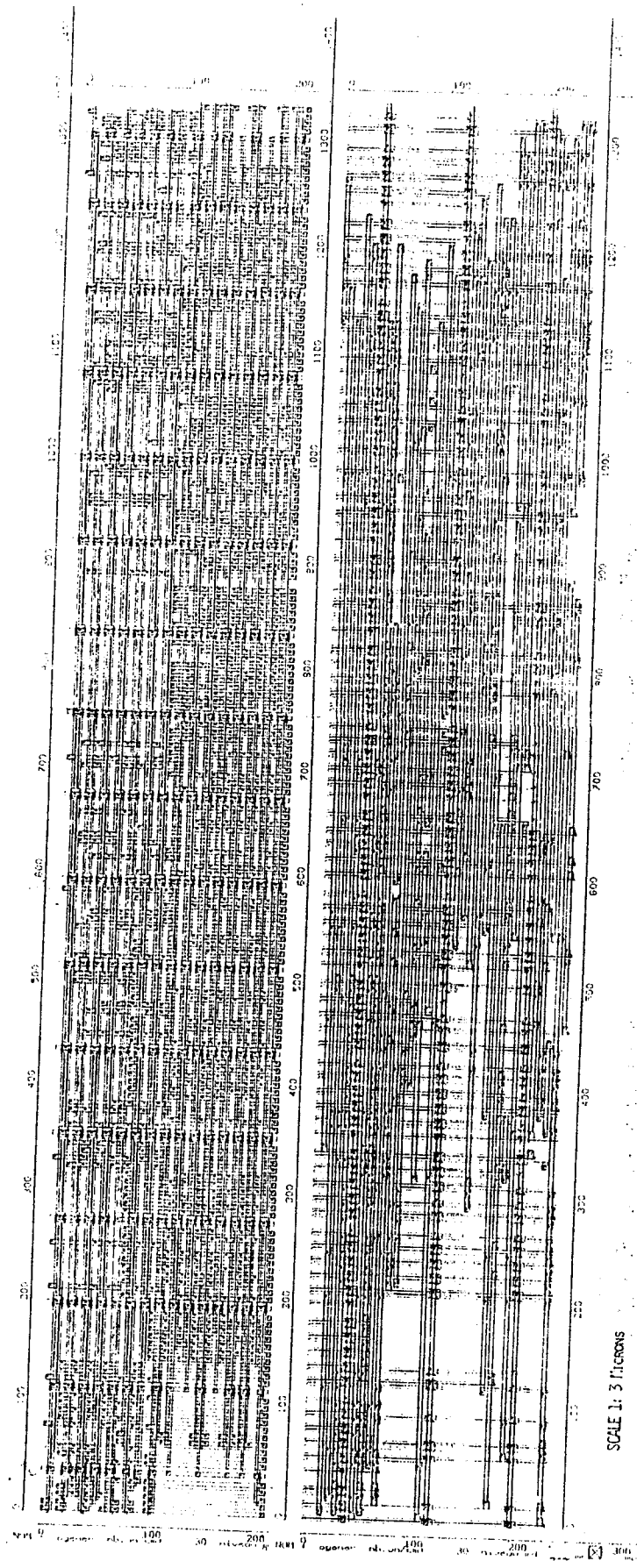


Figure 7-16- Masque du PLA de génération des commandes

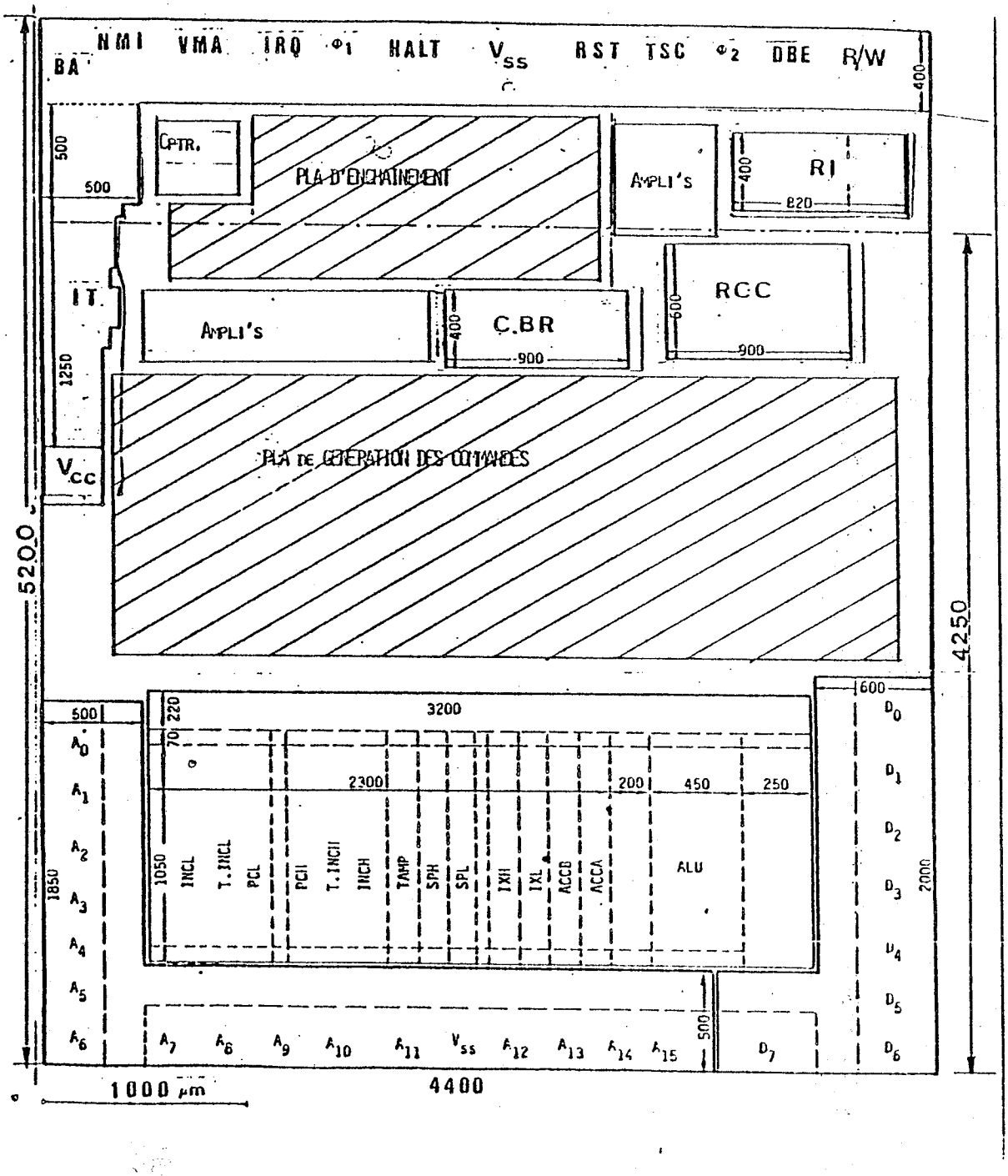


Figure 7-17 topologie du MC6800, réalisation Mealy

****

CHAPITRE VIII

CONCLUSION

Introduction

L'augmentation de la compléxité des microprocesseurs nous a amené à concevoir un outil de C.A.O. pour une génération automatique de parties contrôles. Cet outil réalisé dans l'axe du projet CAPRI est interface à d'autres outils de C.A.O. pour la génération des masques de PLA.

La connexion du système PAMELA aux outils de génération des spécifications de haut niveaux (l'algorithme d'interprétation des instructions) est établie à travers un langage intermédiaire.

L'une des architectures de parties contrôles constituée des PLA (enchainement et génération) est visée par ce système.

L'efficacité de cet outil est testé en regénérant la partie contrôle du microprocesseur MC6800. Les résultats obtenus sont mis en comparaison avec la surface du circuit original implanté en logique anarchique. La réalisation Mealy de la partie contrôle nécessite seulement 22 % d'augmentation de surface par rapport au MC6800 original. Un résultat tout à fait acceptable pour les circuits à la demande qui nécessitent une vitesse et sécurité de conception.

1-Horizons et extensions

Plusieurs extensions sont envisagables sur les divers plans de ce travail.

L'étude consacrés à l'analyse des signaux de contrôle de l'algorithme d'interprétation des instructions du MC68000 (annexes 4 et 5) donne une idée de conception et d'organisation des signaux d'exceptions: initialisation (Reset), arrêt(Halt), interruption et codes invalides. La priorité d'execution de ces signaux étant bien établie dans la conception du MC68000,elle permet la conception anarchique des circuits de la partie contrôle, constituée de PLA.

Extension du langage intermédiaire afin de générer des PLA de priorités implantés dans le circuit anarchique de la partie contrôle permettra de classer la priorité de traitement des

exceptions. La prise en compte des priorités des interruptions en plusieurs niveaux et le classement des exceptions peut être organisé en utilisant les PLA de priorités étudiés dans l'annexe 5-$1-16 et $2-4-3.

La structure du langage intermédiaire peut être équipée d'un système graphique. Ce système permettra de visualiser les états, les conditions, les transitions,....etc. Un tel outil facilitera les manipulation sur les aspects Moore et Mealy des algorithme. Un compromis des deux solutions peut être atteint grâçe aux possibilités de visualisation de l'algorithme sous forme de graphecets.

La conception d'autres outils de génération automatique de partie contrôle sous forme de(s) ROM(s) ou d'un mélange de PLA et de ROM (Cas du MC68000) reste à concevoir.

Les outils d'assemblage automatique existant ne permettent pas une conception automatique du microprocesseur en entier. La génération automatique de la partie opérative à l'aide de l'assembleur des briques LUBRICK est possible mais la conception de la logique anarchique entre la P.C. et la P.O. et les circuits associés à la P.C. (circuits des exceptions) est un travail manuel. Un outil de l'assemblage automatique des blocs (PLA, Partie opérative, circuit d'interruption,.. etc..) doit être envigager.

L'évaluation automatique des surfaces des PLA et éventuellement de la partie contrôle donne une idée assez precise dans les premieres étapes de la conception d'un circuit integré. Un outil plus simple que le générateur de PLA inspiré de sa structure algorithmique peut évaluer la surface des PLA de manière automatique en acceptant son algorithme d'interprétation des instructions.

BIBLIOGRAPHIE

[ANC 83] F.ANCEAU,
CAPRI: A design methodologie and a silicon compiler for VLSI circuits specified by algorithms,
International conference Very Large Scale Integration,
Trondheim, Norway, 16-19 August 1983.

[ANC 80] F.ANCEAU,
Architecture and design of von Neumann microprocessors
NATO advanced Summer institute, juillet 1980

[BOS 80 A.BOSSEBOEUF,
Analyse du fonctionnement interne du MC 68000
Rapport de DEA, ENSIMAG, Juin 1980

[CHU 82] S.CHUQUILLANQUI and T.PEREZ SEGOVIA,
PAOLA: A tool for topological optimization of large PLAs,
IEEE, 19th Design automation conference,
Las Vegas, June 1982, pp. 300-306

[DER 81] H.DERANTONIAN,
Etude de la structure interne et du fonctionnement du microprocesseur INS 807X de National Semiconductor
Rapport de DEA, ENSIMAG, Septembre 1981

[DER 84] D.BASKERA,H.DERANTONIAN, P.MARCHAL, M.NICOLAIDIS,
Fonctionnement et test du microprocesseur MC 68000
Rapport final contrat EDF/ENSIMAG, no.120 IB 1455

[FOR 83] J.FORREST, M.D.EDWARDS,
The automatic generation of programmable logic arrays from algorithmic state machine descriptions
proceedings of IFIP TC 10/WG 10.5 International conference on Very Large Scale Integration '
Trondheim, Norway, 16-19 August 1983

[GUI 78] M.GUITTET,
Microprogrammation du microprocesseur MC 68000,
Rapport d'ingénieur ENSIMAG, Juin 1978.

[LAT 79] B.LATTIN,
THE PROBLEMS OF 80'S FOR MICROPROCESSOR DESIGN
CALTECH CONFERENCE ON VLSI, January 1979

[MAR 80] J.M.C.A. MARQUEZ,
MOSAIC: une méthodologie de conception pour les circuits système VLSI
Thèse de Docteur Ingénieur, INPG, Septembre 1980

[MAR 83] S.MARINE, F.ANCEAU and K.JAHIDI,
IRENE, un langage de descriptions des circuits intégrés logiques",
R.R. No. 356 IMAG,France Mars 1983.

[NEM 79] M.NEMMOUR,
Etude du fonctionnement des microprocesseurs MC6800,
Rapport final de contrat EDF/ENSIMAG,
no. 511 78 1 0, Mai 1979

[OBR 82] M.OBRBSKA,
Etude comparative de differentes méthodes de conception des parties contrôle des microprocesseurs
Thèse de Docteur Ingénieur
INPG, Juin 1982

[OBR 81] M.OBREBSKA,
Comparative survay of different design methodologies for control part of microprocessors,
Carneige-Mellon University
Conference on VLSI systems and computations,
Pittsburgh. Pennsylvania, USA, 19/21 Octobre 1981.

[PER 80] PEREZ SEGOVIA,
Optimisation en surface des PLA's
Rapport de DEA, ENSIMAG, Juin 1980

[REI 80] R.REIS,
Etude du séquencement dans le microprocesseur Z8000
Rapport de DEA, ENSIMAG/ENSERG, Septembre 1980

[REI 81] R.REIS,
Evaluateur topologique pour circuits VLSI
Rapport de recherche IMAG no. 252, Juin 1981.

[SIS 82] J.M.SISKIND,J.R.SOUTHARD and K.W.CROUCH,
"Generating custom high performance VLSI design from succinct algorithmic description "
conference on advanced research in VLSI,
M.I.T , Janv. 1982

[SCH 83] J.P.SCHOELLKOPF
LUBRICK: a silicon assembler and its application to data-path design for FISC".
R.R.No. 363, March 1983 and proc. of
VLSI 1983, Trandheim.August 1983, North Hall.Ed.

[SUZ 81] A.SUZIM,
Etude des parties opérative à éléments modulaires pour processeur monolithique,
Thèse de Docteur Ingénieur, INPG, Novembre 1981.

[THO 81] D.E.THOMAS, D.P.SIEWIOREK,
Measuring design performance to verify
design automation systems
IEEE TRANSACTIONS ON COMPUTERS,
Vol. c-30. no. 1, January 1981

[TOM 80] TOMLINSON G.RAUSCHER, PHILLIP.M.ADAMS,
"Microprogramming: A tutorial and survey of recent
developpements.
IEEE transactions on computers", vol C-29 No.1, January 1980.

ANNEXE I

1-DESCRIPTION ALGORITHMIQUE DE L'AUTOMATE DU MC6800 (MOORE)

```
p1
1 1 -1
p2
1 41 -1
AS-CARRY
3 201 101 2 4 00 01 10 11 -1
AS-UAL
3 202 101 3 4 001 101 000 111 -1
DBS
5 801 1 1 -1
ILB
5 802 1 1 -1
ILP
5 803 1 1 -1
PLL
5 804 1 1 -1
PLD
5 805 1 1 -1
0-DB0
5 806 1 1 -1
0-DB1
5 807 1 1 -1
0-DB2
5 808 1 1 -1
PHD
5 809 1 1 -1
TINO
5 810 1 1 -1
DEC
5 811 1 1 -1
TIM
5 812 1 1 -1
IABH
5 813 1 1 -1
DBI
5 814 1 1 -1
---
```

```
5 815 1 1 -1
TRD
5 816 1 1 -1
DSL
5 817 1 1 -1
ASP
5 818 1 1 -1
SLA
5 819 1 1 -1
SLD
5 820 1 1 -1
DSH
5 821 1 1 -1
SHD
5 822 1 1 -1
DXN
5 823 1 1 -1
BXR
5 824 1 1 -1
XHB
5 825 1 1 -1
XHD
5 826 1 1 -1
XHL
5 827 1 1 -1
BLL
5 828 1 1 -1
DXL
5 829 1 1 -1
XLL
5 830 1 1 -1
XLD
5 831 1 1 -1
DCB
5 832 1 1 -1
CBL
5 833 1 1 -1
```

```
CBD
5 834 1 1 -1
O-BL1
5 835 1 1 -1
DCA
5 836 1 1 -1
CAL
5 837 1 1 -1
CAD
5 838 1 1 -1
PSDbar
5 839 1 1 -1
CPA
5 840 1 1 -1
BAA
5 841 1 1 -1
SR
5 842 1 1 -1
SUD
5 843 1 1 -1
BID
5 844 1 1 -1
WRT8
5 845 1 1 -1
LIR
5 846 1 1 -1
VMA
5 847 1 1 -1
CSHALT
7 1401 1 -1
CSRST
7 1402 -1
RI
7 1403 8 -1
S
7 1404 1 -1
C
```

```
7 1405 1 -1
A
7 1406 1 -1
ET
7 1407 1 -1
IX
7 1408 1 -1
CBR2
7 1409 1 -1
IT
7 1410 1 -1
IRQC
7 1411 1 -1
NMIC
7 1412 1 -1
RTIS
7 1413 1 -1
RST
8 1702 1 1402
2
1
0
-1
HALT
8 1701 1 1401
2
1
0
-1
CND-ECL
8 1703 7 1408 1407 1406 1405 1404 1403 1401
80
XXXXX0010XXXXX
XXXXX0011001XX
XXXXX001110X1X
XXXXX00001000X
XXXXX00001001X
```

XXXXX00000101X
XXXXX00110101X
XXXXX00110001X
XXXXX00110000XX
XXXXX00110100X
XXXXX00110110X
XXXXX00110111X
XXXX11201XXXXX
XXX1X1X01XXXXX
XX1XX1X01XXXXX
X1XXXXXXXXXXX
1XXXXXXXXXXXX
XXXXX10110000X
XXXXX10110010X
XXXXX10110100X
XXXXX10111011X
XXXXX10111000X
XXXXX10111001X
XXXXX10111010X
XXXXX10110110X
XXXXX10110001X
XXXXX10110101X
XXXXX11110000X
XXXXX11110010X
XXXXX11110100X
XXXXX11111011X
XXXXX11111000X
XXXXX11111001X
XXXXX11111010X
XXXXX11110110X
XXXXX11110001X
XXXXX11110101X
XXXXX10001110X
XXXXX10001111X
XXXXX10001100X
XXXXX00011101X
XXXXX00011111X

```
XXXXX00011001X
-1
BR-C
8 1704 1 1409
2
1
0
-1
WAI-SWI
8 1705 1 1403
2
101X1101
0011111X
-1
JSR-IX,ET
8 1706 1 1403
3
00101111
00101101
00101100
-1
CBR2
8 1707 1 1409
2
1
0
-1
CND-SWI,WAI
8 1708 1 1403
2
00111110
01111111
-1
IT
8 1709 1 1410
2
1
```

```
0
-1
CND-SWI,IRQC,NMIC
8 1710 3 1412 1411 1403
3
XX00111111
0XXXXXXXXX
X0XXXXXXXX
-1
PULA,B-RTS,I
8 1711 1 1403
2
001110X1
0011001X
-1
CND-TSX
8 1712 1 1403
2
0000100X
XXXXXXXX
-1
DEλ-TXS
8 1713 1 1403
2
00110101
00001001
-1
TSX-INS
8 1714 1 1403
2
00110001
00110000
-1
STSXAB
8 1715 1 1403
4
10XX0111
```

```
11XX0111
110X1111
10XX1111
-1
LDSX-CPX
8 1716 1 1403
3
10XX1110
11XX1110
10XX1100
-1
TSTFCH
8 1717 3 1413 1408 1403
2
11X01XXXXX
1XXXXXXXXX
-1
CND- SUBF1
8 1718 1 1403
20
10000000
10000010
10110100
10001011
10001000
10001001
10001010
10000110
10000001
10000101
11000000
11000010
11000100
11001011
11001000
11001001
11001010
```

```
11000110
11000001
11000101
-1
CND-JMP1
8 1719 4 1406 1405 1404 1403
4
X1XXXXXXXX
1XXXXXXXXX
XX1XXXXXXX
XXX011X1110
-1
CND-JSR1
8 1720 1 1403
32
10121101
10000000
10000010
10100100
10001011
10001000
10001001
10001010
10000110
10000001
10000101
11000000
11000010
11000100
11001011
11001000
10001001
11001001
11000110
11000001
11000101
01110000
```

```
01110011
01110011
01110110
01110111
01111000
01111000
01111010
01111100
01111111
01111101
-1
CND-JSR2
8 1722 2 1408 1403
2
1XXXXXXXX
X01X11101
-1
CND-JMP2
8 1723 5 1408 1406 1405 1404 1403
4
11XXXXXXXXXX
1X1XXXXXXXXX
1XX1XXXXXXXX
XXXX011X1110
-1
CXA
8 1724 1 1403
31
01110000
01110011
01110100
01110110
01110111
01111000
01111001
01111010
01111100
```

```
01111111
01111101
10000000
10000010
10110100
10001011
10001000
10001001
10001010
10000110
10000001
10000101
11000000
11000010
11000100
11001011
11001000
11001001
11001010
11000110
11000001
11000101
-1
CXC
8 1725 2 1405 1403
3
110XX1110
111XX1110
110XX1100
-1
CXS
8 1726 2 1408 1403
4
110XX0111
111XX0111
111XX1111
111XXXXXX
```

```
-1
RTI-RTS
8 1727 1 1403
2
00111011
00111001
-1
PULA-B
8 1728 1 1403
2
00110010
00110011
-1
RSTO
9 2001 6 801 806 201 1 202 1
       1 1702 2001 2002 -1
RST1
9 2002 7 801 806 847 201 1 202 2
       1 1400 2006 -1
RST1I
9 2003 8 806 807 808 847 201 1 202 2
       1 1400 2006 1 2032 -1
RST1N
9 2004 7 806 807 201 1 202 2 847
       1 1400 2006 -1
RST1S
9 2005 7 806 808 847 201 1 202 2
       1 1400 2006 -1
RST2
9 2006 8 802 814 844 847 201 1 202 1
       1 1400 2007 -1
F3.1
9 2007 9 801 815 844 846 847 201 1 202 1
       1 1701 2013 2010 -1
F2.1
9 2008 8 802 803 846 847 201 1 202 1
       1 1701 2013 2010 -1
```

```
F1.1
9 2009 7 804 846 847 201 1 202 1
       1 1701 2013 2010 -1
F.2
9 2010 7 802 803 847 201 1 202 1
       1 1703 2014 2033 2170 2041 2043 2171 2044 2045 2048 2049 2051 2058
2068 2109 2125 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100
2101 2102 2103 2104 2105 2106 2107 2108 2059 2060 2061 2056 2054 2053 2052
2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144
2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159
2160 2161 2162 2163 2164 2165 2166  2013 -1
F2SA
9 2011 9 802 803 836 843 847 201 1 202 1
       1 1703 2014 2033 2170 2041 2043 2171 2044 2045 2048 2049 2051 2058
2068 2109 2125 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100
2101 2102 2103 2104 2105 2106 2107 2108 2059 2060 2061 2056 2054 2053 2052
2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144
2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159
2160 2161 2162 2163 2164 2165 2166 2013   -1
F2SB
9 2012 9 802 803 832 843 847 201 1 202 1
       1 1703 2014 2033 2170 2041 2043 2171 2044 2045 2048 2049 2051 2058
2068 2109 2125 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100
2101 2102 2103 2104 2105 2106 2107 2108 2059 2060 2061 2056 2054 2053 2052
2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144
2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159
2160 2161 2162 2163 2164 2165 2166 2013   -1
HLT2
9 2013 6 802 803 201 1 202 1
       1 1701 2009 2013  -1
BR1
9 2014 10 802 803 810 814 828 844 201 1 202 1
       1 1704 2015 2020 -1
W1
9 2015 9 805 811 819 845 847 201 1 202 1
       1 1400 2016 -1
W2
```

```
9 2016 9 802 809 811 845 847 201 1 202 1
       1 1705 2017 2024 -1
S3
9 2017 6 802 818 201 1 202 1
       1 1706 2018 2021 2022 -1
JS.ET1
9 2018 6 805 811 201 1 202 1
       1 1400 2019 -1
JS.ET2
9 2019 6 802 847 201 1 202 1
       1 1400 2007 -1
BR2
9 2020 10 801 802 810 811 812 843 201 1 202 1
       1 1707 2008 2009 -1
JS.IX
9 2021 9 810 816 825 828 830 201 1 202 1
       1 1400 2023 -1
BSR1
9 2022 10 802 803 810 814 828 844 201 1 202 1
       1 1400 2023  -1
JS/BS
9 2023 9 801 802 810 812 843 201 1 202 1
       1 1400 2008 -1
W3
9 2024 8 802 811 831 845 201 1 202 1
       1 1400 2025 -1
W4
9 2025 9 802 811 827 845 847 201 1 202 1
       1 1400 2026 -1
W5
9 2026 9 802 811 838 845 847 201 1 202 1
       1 1400 2027 -1
W6
9 2027 9 802 811 834 845 847 201 1 202 1
       1 1400 2028 -1
W7
9 2028 9 802 811 839 845 847 201 1 202 1
```

```
        1 1708 2029 2032 -1
W8
9 2029 7 802 810 818 201 1 202 1
        1 1400 2030 -1
W9
9 2030 7 802 810 818 201 1 202 1
        1 1709 2031 2030 -1
W10
9 2031 7 802 810 818 201 1 202 1
        1 1400 2032 -1
W11
9 2032 6 802 818 201 1 202 1
        1 1710 2005 2004 2003 -1
RT1
9 2033 5 802 201 1 202 2
        1 1711 2167 2034 -1
RT2
9 2034 6 802 847 201 1 202 1
        1 1727 2035 2040 -1
RT3
9 2035 6 802 844 201 1 202 1
        1 1400 2036 -1
RT4
9 2036 7 802 832 844 201 1 202 1
        1 1400 2037 -1
RT5
9 2037 7 802 836 844 201 1 202 1
        1 1400 2038 -1
RT6
9 2038 8 802 823 844 847 201 1 202 1
        1 1400 2039  -1
RT7
9 2039 7 802 829 844 201 1 202 1
        1 1400 2040 -1
RT8
9 2040 9 802 814 818 844 847 201 1 202 1
        1 1400 2007 -1
```

```
INX
9 2041 7 825 828 830 201 1 202 1
       1 1400 2042 -1
IN
9 2042 6 802 824 201 1 202 1
       1 1712 2047 2009 -1
DXTX
9 2043 8 811 825 828 830 201 1 202 1
       1 1713 2046 2042 -1
ISTS
9 2044 5 819 201 1 202 1
       1 1714 2046 2042 -1
DES
9 2045 6 811 819 201 1 202 1
       1 1400 2046  -1
TS
9 2046 6 802 818 201 1 202 1
       1 1400 2009 -1
INX2
9 2047 9 826 828 830 846 847 201 1 202 1
       1 1400 2010 -1
PUSHA
9 2048 9 811 819 838 845 847 201 1 202 1
       1 1400 2050 -1
PUSHB
9 2049 9 811 819 834 845 847 201 1 202 1
       1 1400 2050 -1
PUSH2
9 2050 6 802 818 201 1 202 1
       1 1400 2009 -1
DR1.S
9 2051 10 801 803 810 813 814 844 201 1 202 1
       1 1715 2052 2053 2054 2056 -1
STAB
9 2052 8 802 834 845 847 201 1 202 1
       1 1400 2009 -1
STAA
```

```
9 2053 8 802 838 845 847 201 1 202 2
       1 1400 2009 -1
STX1
9 2054 8 802 826 845 847 201 1 202 2
       1 1400 2055 -1
STX2
9 2055 8 802 831 845 847 201 1 202 1
       1 1400 2009 -1
STS1
9 2056 8 802 822 845 847 201 1 202 2
       1 1400 2057 -1
STS2
9 2057 8 802 820 845 847 201 1 202 2
       1 1400 2009 -1
DR1.C
9 2058 9 801 803 813 814 844 201 1 202 1
       1 1716 2059 2060 2061 -1
LDS1
9 2059 8 802 821 844 847 201 1 202 2
       1 1717 2062 2063 -1
LDX1
9 2060 8 802 803 844 847 201 1 202 2
       1 1717 2064 2065 -1
CPX1
9 2061 9 802 827 841 844 847 201 1 202 2
       1 1717 2066 2067 -1
LDS2F.1
9 2062 9 804 817 844 846 847 201 1 202 2
       1 1400 2010 -1
LDS2F.2
9 2063 10 802 803 817 844 846 847 201 1 202 2
       1 1400 2010 -1
LDX2F.1
9 2064 9 804 829 844 846 847 201 1 202 2
       1 1400 2010 -1
LDX2F.2
9 2065 10 802 803 829 844 846 847 201 1 202 2
```

```
        1 1400 2010 -1
CPX2F.1
9 2066 10 804 830 841 844 846 847 201 1 202 2
        1 1400 2010 -1
CPX2F.2
9 2067 11 802 803 830 841 844 846 847 201 1 202 1
        1 1400 2010 -1
DR1.A
9 2068 11 801 803 810 813 814 844 847 201 1 202 1
        1 1718 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079
2080 2081 2082 2083 2084 2085 2086 2087 2088 -1
SUBAF.1
9 2069 10 804 837 841 844 846 847 201 1 202 2
        1 1400 2011 -1
SBCAF.1
9 2070 10 804 837 841 844 846 847 201 4 202 2
        1 1400 2011 -1
ANDAF.1
9 2071 9 804 837 844 846 847 201 1 202 1
        1 1400 2011 -1
ADDAF.1
9 2072 9 804 837 844 846 847 201 1 202 2
        1 1400 2011 -1
EORAF.1
9 2073  9 804 837 844 846 847 201 1 202 3
        1 1400 2011 -1
ADCAF.1
9 2074 9 804 837 844 846 847 201 3 202 2
        1 1400 2011 -1
ORAAF.1
9 2075 9 804 837 844 846 847 201 1 202 3
        1 1400 2011 -1
LDAAF.1
9 2076 8 804 844 846 847 201 1 202 1
        1 1400 2011 -1
CMPAF.1
9 2077 9 804 837 844 846 847 201 2 202 2
```

```
        1 1400 2010 -1
BITAF.1
9 2078 9 804 837 844 846 847 201 1 202 1
       1 1400 2010 -1
SUBBF.1
9 2079 10 804 833 841 844 846 847 201 2 202 2
       1 1400 2012 -1
SBCBF.1
9 2080 10 804 833 841 844 846 847 201 4 202 2
        1 1400 2012 -1
ANDBF.1
9 2081 9 804 833 844 846 847 201 1 202 1
        1 1400 2012 -1
ADDBF.1
9 2082 9 804 833 844 846 847 201 1 202 2
       1 1400 2012 -1
EORBF.1
9 2083 9 804 833 844 846 847 201 1 202 3
        1 1400 2012 -1
ADCBF.1
9 2084 9 804 833 844 846 847 201 3 202 2
        1 1400 2012 -1
ORABF.1
9 2085 9 804 833 844 846 847 201 1 202 3
        1 1400 2012 -1
LDABF.1
9 2086 8 804 844 846 847 201 1 202 1
       1 1400 2012 -1
CMPBF1
9 2087 10 804 833 841 844 846 847 201 2 202 2
        1 1400 2010 -1
BITBF.1
9 2088 9 804 833 844 846 847 201 1 202 1
       1 1400 2010 -1
SUBAF.2
9 2089 11 802 803 837 841 844 846 847 201 2 202 2
        1 1400 2011 -1
```

```
SBCAF.2
9 2090 11 802 803 837 841 844 846 847 201 4 202 2
        1 1400 2011 -1
ANDAF.2
9 2091 10 802 803 837 844 846 847 201 1 202 1
        1 1400 2011 -1
ADDAF.2
9 2092 10 802 803 837 844 846 847 201 1 202 2
        1 1400 2011 -1
EORAF.2
9 2093 10 802 803 837 844 846 847 201 1 202 3
        1 1400 2011 -1
ADCAF.2
9 2094 10 802 803 837 844 846 847 201 3 202 2
          1 1400 2011 -1
ORAAF.2
9 2095 10 802 803 837 844 846 847 201 1 202 3
        1 1400 2011 -1
LDAAF.2
9 2096  9 802 803 844 846 847 201 1 202 1
        1 1400 2011 -1
CMPAF.2
9 2097 11 802 803 837 841 844 846 847 201 2 202 2
        1 1400 2010 -1
BITAF.2
9 2098 10 802 803 837 844 846 847 201 1 202 1
        1 1400 2010 -1
SUBBF.2
9 2099  11 802 803 833 841 844 846 847 201 2 202 2
          1 1400 2012 -1
SBCBF.2
9 2100  11 802 803 833 841 844 846 847 201 4 202 2
        1 1400 2012 -1
ANBF.2
9 2101  10 802 803 833 844 846 847 201 1 202 1
          1 1400 2012 -1
ADDBF.2
```

```
9 2102  10 802 803 833 844 846 847 201 1 202 2
          1 1400 2012 -1
EORBF.2
9 2103  10 802 803 833 844 846 847 201 1 202 3
        1 1400 2012 -1
ADCBF.2
9 2104  10 802 803 833 844 846 847 201 3 202 2
        1 1400 2012 -1
ORABF.2
9 2105  10 802 803 833 844 846 847 201 1 202 3
        1 1400 2012 -1
LDABF.2
9 2106  9 802 803 844 846 847 201 1 202 1
        1 1400 2012 -1
CMPBF.2
9 2107  11 802 803 833 841 844 846 847 201 2 202 2
          1 1400 2010 -1
BITBF.2
9 2108  10 802 803 833 844 846 847 201 1 202 1
        1 1400 2010 -1
ET1
9 2109 8 802 814 844 847 201 1 202 1
       1 1719 2110 2111 2112 2007 -1
ET2.C
9 2110 9 801 803 815 844 847 201 1 202 1
       1 1716 2059 2060 2061 -1
ET2.A
9 2111 10 801 803 810 815 844 847 201 1 202 1
       1 1720 2015 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078
2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2113 2114 2115 2116 2117
2118 2119 2120 2121 2122 2123 -1
ET2.S
9 2112 9 801 803 810 815 844 201 1 202 1
       1 1715 2052 2053 2054 2056 -1
NEGM
9 2113 10 802 803 810 835 841 844 201 2 202 1
       1 1400 2124 -1
```

```
COMM
2 2114 9 802 803 810 841 844 201 1 202 2
        1 1400 2124 -1
LSRM
9 2115 10 802 803 810 835 842 844 201 1 202 1
       1 1400 2124 -1
RORM
9 2116 10 802 803 810 835 842 844 201 1 202 1
        1 1400 2124 -1
ASRM
9 2117 10 802 803 810 835 842 844 201 1 202 1
        1 1400 2124 -1
ASLM
9 2118 8 802 803 810 844 201 1 202 4
        1 1400 2124 -1
ROLM
9 2119 8 802 803 810 844 201 3 202 4
        1 1400 2124 -1
DECM
9 2120 8 802 803 810 844 201 1 202 1
        1 1400 2124 -1
INCM
9 2121 9 802 803 810 835 844 201 2 202 1
1 1400 2124 -1
CLRM
9 2122 9 802 803 810 835 844 201 1 202 2
        1 1400 2124 -1
TSTM
9 2123 9 802 803 810 835 844 201 1 202 1
        1 1400 2124 -1
NFIN
9 2124 9 802 843 845 846 847 201 1 202 1
        1 1400 2009 -1
IX1
9 2125 11 803 810 814 825 828 830 844 201 1 202 1
       1 1722 2126 2015 -1
IX2
```

```
9 2126 9 801 802 810 812 843 201 1 202 1
        1 1723 2127 2128 2129 2008 -1
IX3.A
9 2127 7 802 810 847 201 1 202 1
       1 1724 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123
2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2
2085 2086 2087 2088
-1
IX3.C
9 2128 6 802 847 201 1 202 1
        1 1725 2059 2060 2061 -1
IX3.S
9 2129 6 802 810 201 1 202 1
        1 1726 2056 2054 2053 2052 -1
NEGA
9 2130 11  803 804 835 838 841 846 847 201 1 202 1
       1 1400 2011 -1
COMA
9 2131 10 803 804 838 841 846 847 201 1 202 2
       1 1400 2011 -1
LSRA
9 2132 11 803 804 835 838 842 846 847 202 1 202 1
       1 1400 2011 -1
RORA
9 2133 11 803 804 835 838 842 846 847 201 1 202 1
        1 1400 2011 -1
ASRA
9 2134 11 803 804 835 838 842 846 847  201 1 202 1
       1 1400 2011 -1
ASLA
9 2135 9 803 804 838 846 847 201 1 202 4
        1 1400 2011 -1
ROLA
9 2136 9 803 804 838 846 847 201 3 202 4
        1 1400 2011 -1
DECA
9 2137 9 803 804 838 846 847 201 1 202 1
```

```
        1 1400 2011 -1
INCA
9 2138 10 803 804 835 838 846 847 201 2 202 1
        1 1400 2011 -1
CLRA
9 2139 10 803 804 835 838 846 847 201 1 202 2
        1 1400 2011 -1
TSTA
9 2140 10 803 804 835 838 846 847 201 1 202 1
        1 1400 2010 -1
NEGB
9 2141 11 803 804 834 835 842 846 847 201 1 202 1
        1 1400 2012 -1
COMB
9 2142 10 803 804 834 841 846 847 201 1 202 2
        1 1400 2012 -1
LSRB
9 2143 11 803 804 834 835 842 846 847 201 1 202 1
        1 1400 2012 -1
RORB
9 2144 11 803 804 834 835 842 846 847 201 1 202 1
        1 1400 2012 -1
ASRB
9 2145 11 803 804 834 835 842 846 847 201 1 202 1
        1 1400 2012 -1
ASLB
9 2146 9 803 804 834 846 847 201 1 202 4
        1 1400 2012 -1
ROLB
9 2147 9 803 804 834 846 847 201 3 202 4
        1 1400 2012 -1
DECB
9 2148 9 803 804 834 846 847 201 1 202 1
        1 1400 2012 -1
INCB
9 2149 10 803 804 834 835 846 847 201 2 202 1
        1 1400 2012 -1
```

```
CLRB
9 2150 10 803 804 834 835 846 847 201 1 202 2
       1 1400 2012 -1
TSTB
9 2151 10 803 804 834 835 846 847 201 1 202 1
        1 1400 2010 -1
NOP
9 2152 7 804 846 847 201 1 202 1
        1 1400 2010 -1
TAP
9 2153 9 804 832 838 846 847 201 1 202 1
        1 1400 2010 -1
TPA
9 2154 9 804 836 839 846 847 201 1 202 1
        1 1400 2010 -1
CLV
9 2155 7 804 846 847 201 1 202 1
        1 1400 2010 -1
SEV
9 2156 7 804 846 847 201 1 202 1
        1 1400 2010 -1
CLC
9 2157 7 804 846 847 201 1 202 1
        1 1400 2010 -1
SEC
9 2158 7 804 846 847 201 1 202 1
        1 1400 2010 -1
CLI
9 2159 7 804 846 847 201 1 202 1
        1 1400 2010 -1
SEI
9 2160 7 804 846 847 201 1 202 1
        1 1400 2010 -1
SBA
9 2161 10 804 834 837 841 846 847 201 2 202 1
        1 1400 2011 -1
CBA
```

```
9 2162 10 804 834 837 841 846 847 201 2 202 1
        1 1400 2010 -1
TAB
9 2163 8 804 838 846 847 201 1 202 2
        1 1400 2012 -1
TBA
9 2164 8 804 834 837 840 201 1 202 2
        1 1400 2011 -1
DAA
9 2165 7 804 846 847 201 1 202 1
        1 1400 2011 -1
ABA
9 2166 9 804 834 837 846 847 201 1 202 1
        1 1400 2011 -1
RT2*
9 2167 7 802 818 847 201 1 202 1
        1 1728 2168 2169 -1
PULA
9 2168 9 804 806 844 846 847 201 1 202 1
        1 1400 2011 -1
PULB
9 2169 9 804 832 844 846 847 201 1 202 1
        1 1400 2012 -1

FIN
```

ANNEXE II

DESCRIPTION ALGORITHMIQUE DE L'AUTOMATE

DU MC6800 (MEALY)

```
p1
1 1 -1
p2
1 41 -1
AS-CARRY
3 201 101 2 4 00 01 10 11 -1
AS-UAL
3 202 101 3 4 001 101 000 111 -1
AC601-RST
4 601 3 801 807 808
1729
   1 0 0
   0 1 1
   0 1 0
   0 0 1
 -1
AC602-F.1
4 602 6 801 802 803 804 815 844 1730
 0 0 0 1 0 0
 0 1 1 0 0 0
 1 0 0 0 1 1
 -1
AC603-F.2
4 603 3 832 836 843 1731
  0 0 0
  0 1 1
  1 0 1
 -1
AC604-PULAB
```

```
 4 604 2 832 836 1740
 0 1
 1 0
  -1
 AC608-PSHAB
 4 608 2 834 838 1710
 0 1
 1 0
  -1
 AC613
 4 613 2 834 838 1732
 0 1
 1 0
  -1
 AC615
 4 615 2 822 826 1733
 1 0
 0 1
  -1
 AC616
 4 616 2 820 831 1733
 1 0
 0 1
  -1
 AC617
 4 617 2 821 823 1734
 1 0
 0 1
  -1
AC607
4 607 5 802 803 804 817 829 1717
0 0 1 1 0
1 1 0 1 0
0 0 1 0 1
1 1 0 0 1
 -1
AC 605
```

```
4 605 3 802 803 804 1735
0 0 1
1 1 0
-1
AC609-SUB
4 609 8 802 803 804 833 837 841 201 202
1736
1 1 0 0 1 1 2 1
1 1 0 0 1 1 4 1
1 1 0 0 1 0 1 2
1 1 0 0 1 0 1 1
1 1 0 0 1 0 1 3
1 1 0 0 1 0 3 1
1 1 0 0 1 0 1 3
1 1 0 0 0 0 1 2
1 1 0 0 1 1 2 1
1 1 0 0 1 0 1 2
1 1 0 1 0 1 2 1
1 1 0 1 0 1 4 1
1 1 0 1 0 0 1 2
1 1 0 1 0 0 1 1
1 1 0 1 0 0 1 3
1 1 0 1 0 0 3 1
1 1 0 1 0 0 1 3
1 1 0 0 0 0 1 2
1 1 0 1 0 1 2 1
1 1 0 1 0 0 1 2
0 0 1 1 0 1 2 1
0 0 1 1 0 1 4 1
0 0 1 1 0 0 1 2
0 0 1 1 0 0 1 1
0 0 1 1 0 0 1 3
0 0 1 1 0 0 3 1
0 0 1 1 0 0 1 3
0 0 1 0 0 0 1 2
0 0 1 1 0 1 2 1
0 0 1 1 0 0 1 2
```

```
0 0 1 0 1 1 2 1
0 0 1 0 1 1 4 1
0 0 1 0 1 0 1 2
0 0 1 0 1 0 1 1
0 0 1 0 1 0 1 3
0 0 1 0 1 0 3 1
0 0 1 0 1 0 1 3
0 0 1 0 0 0 1 2
0 0 1 0 1 0 2 1
0 0 1 0 1 0 1 2
-1
AC611
4 611 5 835 841 842 201 202 1737
1 1 0 2 1
0 1 0 1 2
1 0 1 1 1
1 0 1 1 1
1 0 1 1 1
0 0 0 1 4
0 0 0 3 4
0 0 0 1 1
1 0 0 2 1
1 0 0 1 2
1 0 0 1 1
-1
AC612
4 612 7 834 835 838 841 842 201 202 1738
0 1 1 1 0 2 1
0 0 1 1 0 1 2
0 1 1 0 1 1 1
0 1 1 0 1 1 1
0 1 1 0 1 1 1
0 0 1 0 0 1 4
0 0 1 0 0 3 4
0 0 1 0 0 1 1
0 1 1 0 0 2 1
0 1 1 0 0 1 2
```

```
0 1 1 0 0 1 1
1 1 0 0 1 1 1
1 0 0 1 0 1 2
1 1 0 0 1 1 1
1 1 0 0 1 1 1
1 1 0 0 1 1 1
1 0 0 0 0 1 4
1 0 0 0 0 3 4
1 0 0 0 0 1 1
1 1 0 0 0 2 1
1 1 0 0 0 1 2
1 1 0 0 0 1 1
-1
AC614
4 614 10 832 834 836 837 838 839 840 841 201 202 1739
0 0 0 0 0 0 0 0 1 1
1 0 0 0 1 0 0 0 1 1
0 0 1 0 0 1 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1
0 1 0 1 0 0 0 1 1 1
0 1 0 1 0 0 0 1 1 1
0 0 0 0 1 0 0 0 1 2
0 1 0 1 0 0 1 0 1 2
0 0 0 0 0 0 0 0 1 1
-1
DBS
5 801 1 1 -1
ILB
5 802 1 1 -1
ILP
5 803 1 1 -1
PLL
```

```
5 804 1 1 -1
PLD
5 805 1 1 -1
O-DBO
5 806 1 1 -1
O-DB1
5 807 1 1 -1
O-DB2
5 808 1 1 -1
PHD
5 809 1 1 -1
TINO
5 810 1 1 -1
DEC
5 811 1 1 -1
TIM
5 812 1 1 -1
IABH
5 813 1 1 -1
DBT
5 814 1 1 -1
TRB
5 815 1 1 -1
TRD
5 816 1 1 -1
DSL
5 817 1 1 -1
ASP
5 818 1 1 -1
SLA
5 819 1 1 -1
SLD
5 820 1 1 -1
DSH
5 821 1 1 -1
SHD
5 822 1 1 -1
```

```
DXN
5 823 1 1 -1
BXR
5 824 1 1 -1
XHB
5 825 1 1 -1
XHD
5 826 1 1 -1
XHL
5 827 1 1 -1
BLL
5 828 1 1 -1
DXL
5 829 1 1 -1
XLL
5 830 1 1 -1
XLD
5 831 1 1 -1
DCB
5 832 1 1 -1
CBL
5 833 1 1 -1
CBD
5 834 1 1 -1
O-BL1
5 835 1 1 -1
DCA
5 836 1 1 -1
CAL
5 837 1 1 -1
CAD
5 838 1 1 -1
PSDbar
5 839 1 1 -1
CPA
5 840 1 1 -1
BAA
```

```
5 841 1 1 -1
SR
5 842 1 1 -1
SUD
5 843 1 1 -1
BID
5 844 1 1 -1
WRT8
5 845 1 1 -1
LIR
5 846 1 1 -1
VMA
5 847 1 1 -1
CSHALT
7 1401 1 -1
CSRST
7 1402 1 -1
RI
7 1403 8 -1
SCAET
7 1404 1 -1
ET
7 1405 1 -1
IX
7 1406 1 -1
CBR2
7 1407 1 -1
IT
7 1408 1 -1
IRQC
7 1409 1 -1
NMIC
7 1410 1 -1
RTIS
7 1411 1 -1
TXS
7 1412 1 -1
```

```
IM
7 1413 1 -1
P1,P2
7 1414 2 -1
P2,P3
7 1415 2 -1
CAC605
7 1416 2 -1
HALTC
8 1701 1
  1401
  2
  1
  0
-1
RSTC
8 1702 1 1402
2
1
0
-1
CND-ECL
8 1703 9 1411 1412 1413 1408 1406 1405 1404 1403 1401
18
        XXXXXXX0010XXXX1
        X1XXXXX0011001XX
        XXXXXXX000010001
        1XXXXXX000010011
        XXXXXXX0011000X1
        XXXXXXX001101001
        XXXXXXX0011011X1
        XXXX1XXXX01XXXX0
        XXXXX1XXX01XXXX1
        XXXXXX1XX01XXXX1
        XXX1XXXXXXXXXX1
        XX1XXXXXXXXXXX1
        XXXX1XX1X00X0XX1
```

```
        XXXXXXX1X0011101
        XXXXX1XXX00XX0X1
        XXXXXXX010XXXXX1
        XXXXXXX0000XXXX1
        XXXXXXXXXXXXXXX0
-1
BR-C
8 1704 1 1409
2
1
0
-1
WAI-SWI
8 1705 1 1403
2
101X1101
0011111X
-1
JSR-IX,ET
8 1706 1 1403
3
10111101
10101101
10001101
-1
CBR2
8 1707 1 1409
2
1
0
-1
CND-SWI,WAI
8 1708 1 1403
2
00111110
01111111
-1
```

```
IT
8 1709 1 1410
2
1
0
-1
CAC608-PSHAB
8 1710 1 1403
2
00110110
00110111
-1
PULA,B-RTS,I
8 1711 2 1413 1403
2
X0011001X
1XXXXXXXX
-1
CND-TSX
8 1712 1 1403
2
0000100X
00110000
-1
DEX-TXS
8 1713 1 1403
2
00110101
00001001
-1
TSX-INS
8 1714 1 1403
2
00110001
00110000
-1
STSXAB
```

```
8 1715 1 1403
2
1XXX0111
1XXX1111
-1
LDSX-CPX
8 1716 1 1403
2
1XXX1110
10XX1100
-1
CAC607-LDSX-F.1,F.2
8 1717 1 1414
4
00
01
10
11
-1
CND- SUBF1
8 1718 3 1412 1413 1403
2
11XXXXXXXX
XX0111XXXX
-1
CND-JMP1
8 1719 4 1406 1405 1404 1403
4
X1XXXXXXXXX
1XXXXXXXXXX
XX1XXXXXXXX
XXX01121110
-1
CND-JSR1
8 1720 2 1406 1403
3
X101X1101
```

```
11XXXXXXX
10XXXXXXX
-1
CND-JSR2
8 1722 2 1408 1403
2
1XXXXXXXX
X01X11101
-1
CND-JMP2
8 1723 5 1408 1406 1405 1404 1403
4
11XXXXXXXXX
1X1XXXXXXXX
1XX1XXXXXXX
XXXX011X1110
-1
CXA
8 1724 1 1406
2
0
1
-1
CXC
8 1725 1 1403
2
1XXX1110
10XX1100
-1
CXS
8 1726 1 1403
2
1XXX0111
11XX1111
-1
RTI-RTS
8 1727 1 1403
```

```
2
00111011
00111001
-1
CAC601-RST
8 1729 4 1402 1403 1411 1412
4
0XXXXXXXXXXX
XXXXXXXXXX0X
XXXXXXXXXXX0
X111111100XXX
-1
CAC602-F1
8 1730 1 1414
3
00
10
01
-1
CAC603-F2
8 1731 1 1415
3
00
10
X1
-1
CAC613
8 1732 1 1403
2
10XX0111
11XX0111
-1
CAC615-STSX
8 1733 1 1403
2
10XX1111
11XX1110
```

```
-1
CAC616-LDSX
8 1734 1 1403
2
10XX1110
11XX1110
-1
CAC605
8 1735 1 1416
2
00
11
-1
CAC609-SUB-AB
8 1736 1 1403
40
10110000
10110010
10110100
10111011
10111000
10111001
10111010
10110110
10110001
10110101
11110000
11110010
11110100
11111011
11111000
10111001
11111010
11110110
11110001
11110101
11000000
```

```
11000010
11000100
11001011
11001000
11001001
11001010
11000110
11000001
11000101
10000000
10000010
10110100
10001011
10001000
10001001
10001010
10000110
10000001
10000101
-1
CAC612-NEGAB
8 1738 1 1403
22
01000000
01000011
01000100
01000110
01000111
01001000
01001001
01001010
01001100
01001111
01001101
01011101
01010011
01010100
```

```
01010110
01010111
01011000
01011001
01011010
01011100
01011111
01011101
-1
CAC611
8 1737 1 1403
11
01111101
01110011
01110100
01110110
01110111
01111000
01111001
01111010
01111100
01111111
01111101
-1
CAC614
8 1739 1 1403
15
00000001
00000110
00000111
00001010
00001011
00001100
00001101
00001110
00001111
00010000
```

```
00010001
00010110
00010111
00011001
00011011
-1
CAC604-PULAB
8 1740 1 1403
2
00110010
00110011
-1
RSTO
9 2001 6 801  806  201 1 202 1
                              1 1702 2001 2002  -1
RST1
9 2002 7 806  847  201 1 202 1 601
       1 1400 2006  -1
RST2
9 2006  8 802  814  844  847  201 1 202 1
        1 1400 2007 -1
F3.1
9 2007 7 846  847  201 1 202 1 602 1 1701 2013 2010 -1
F.2
9 2010 7 802  803  201 1 202 1 603
       1 1703 2014 2033 2041 2043 2044 2045 2048 2068 2058 2051 2109 2125
         2069 2059 2061 2130 2152 2013  -1
HLT2
9 2013  7 802  803  810  201 1 202 1
        1 1701 2007 2013  -1
BR1
9 2014 10 802  803  810  814  828  844  201 1 202 1
       1 1704 2015 2020  -1
W1
9 2015  9 805   811  819  845  847  201 1 202 1
        1 1400 2016  -1
W2
```

```
9 2016  9 802  809  811  845  847  201 1 202 1
        1 1705 2017 2024 -1
S3
9 2017 6 802  818  201 1 202 1
       1 1706 2018 2021 2022  -1
JS.ET1
9 2018 6 805  811  201 1 202 1
       1 1400 2019  -1
JS.ET2
9 2019 6 802  847  201 1 202 1
       1 1400 2007  -1
BR2
9 2020 10 801  802  810  811  812  843  201 1 202 1
        1 1400 2007 -1
JS.IX
9 2021  9 810  816  826  828  830  201 1 202 1
        1 1400 2023 -1
BSR1
9 2022  8 804  810  816  828  201 1 202 1
        1 1400 2023 -1
JS/BS
9 2023  9 801  802  810  812  843  201 1 202 1
        1 1400 2007 -1
W3
9 2024  9 802  811  831  845  847  201 1 202 1
        1 1400 2025 -1
W4
9 2025  9 802  811  827  845  847  201 1 202 1
       1 1400 2026 -1
W5
9 2026  9 802  811  838  845  847  201 1 202 1
        1 1400 2027 -1
W6
9 2027  9 802  811  834  845  847  201 1 202 1
        1 1400 2028 -1
W7
9 2028  9 802  811  839  845  847  201 1 202 1
```

```
          1 1708 2029 2032 -1
 W8
 9 2029  8 802  810  818  847  201 1 202 1
          1 1400 2030 -1
 W9
 9 2030  8 802  810  818  847  201 1 202 1
          1 1709 2031 2030 -1
 W10
 9 2031  8 802  810  818  847  201 1 202 1
          1 1400 2032 -1
 W11
 9 2032  7 802  818  847  201 1 202 1
          1 1400 2002 -1
 RT1
 9 2033 6 802  847  201 1 202 1
         1 1711 2167 2034 -1
 RT2
 9 2034 6 802  818  201 1 202 1
         1 1727 2035 2040 -1
 RT3
 9 2035 6 802  844  201 1 202 1
         1 1400 2036 -1
 RT4
 9 2036  7 802  832  844  201 1 202 1
         1 1400 2037 -1
 RT5
 9 2037 6 802  844  201 1 202 1
         1 1400 2038 -1
 RT6
 9 2038  7 802  823  844  201 1 202 1
         1 1400 2039 -1
 RT7
 9 2039  7 802  829  844  201 1 202 1
         1 1400 2040 -1
 RT8
 9 2040  8 802  814  818  844  201 1 202 1
         1 1400 2007 -1
```

```
INX
9 2041  8 825  828  830  847  201 1 202 1
        1 1400 2042 -1
IN
9 2042  7 802  824  847  201 1 202 1
        1 1712 2047 2007 -1
DXTX
9 2043  9 811  825  828  830  847  201 1 202 1
        1 1713 2046 2042 -1
ISTS
9 2044 6 819  847  201 1 202 1
       1 1714 2046 2042 -1
DES
9 2045  7 811  819  847  201 1 202 1
        1 1400 2046 -1
TS
9 2046  7 802  818  847  201 1 202 1
        1 1400 2007 -1
INX2
9 2047  9 826  828  830  846  847  201 1 202 1
        1 1400 2010 -1
PUSHAB
9 2048  9 811  819  845  847  201 1 202 1 608
        1 1400 2050 -1
PUSH2
9 2050 6 802  818  201 1 202 1
       1 1400 2010 -1
DR1.S
9 2051 10 801  803  810  813  814  844  201 1 202 1
        1  1715 2052 2054 -1
STAB
9 2052 7 802  847  201 1 202 2 613
       1 1400 2007 -1
STXS1
9 2054 7 802  847  201 1 202 2 615
       1 1400 2055 -1
STXS2
```

```
9 2055 7 802  847  201 1 202 2 616
       1 1400 2007  -1
DR1.C
9 2058 10 801  803  813  814  844  847  201 1 202 1
       1 1716 2059 2061 -1
LDS1
9 2059  8 802  844  847  201 1 202 2 617
       1 1400 2062 -1
CPX1
9 2061  9 802  827  841  844  847  201 2 202 1
       1 1400 2066 -1
LDSX
9 2062 8 844 846 847 201 1 202 2 607
       1 1400 2010 -1
CPX2F.1
9 2066 10 830 841 844 846 847 201 2 202 1 605
       1 1400 2010 -1
DR1.A
9 2068 11 801  803  810  813  814  844  847  201 1 202 1
       1  1400 2069 -1
SUBAF.1
9 2069 4 844 846 847 609 1 1400 2010 3 2068 2127 2111 -1
ET1
9 2109  8 802  814  844  847  201 1 202 1
       1 1719 2110 2111 2112 2007 -1
ET2.C
9 2110 9 801  803  815  844  847  201 1 202 1
       1 1716 2059 2061 -1
ET2.A
9 2111 10 801 803 810 815 844 847 201 1 202 1
       1 1720 2015 2069 2113 -1
ET2.S
9 2112 9 801 803 810 815 844 201 1 202 1
       1 1715 2052 2054 -1
NEGM
9 2113 5 802 803 810 844 611 1 1400 2124 -1
NFIN
```

```
9 2124 8 802 843 845 847 201 1 202 1
       1 1400 2007 -1
IX1
9 2125 11 803 810 815 825 828 830 844 201 1 202 1
       1 1722 2126 2015 -1
IX2
9 2126 9 801 802 810 812 843 201 1 202 1
       1 1723 2127 2128 2129 2007 -1
IX3.A
9 2127 7 802 810 847 201 1 202 1
       1 1724 2113 2069 -1
.IX3.C
9 2128 6 802 847 201 1 202 1
       1 1716 2059 2061 -1
IX3.S
9 2129 6 802 810 201 1 202 1
       1 1715 2052 2054 -1
NEG
9 2130 5 803 804 846 847 612 1 1400 2010 -1
NOP
9 2152 4 804 846 847 614 1 1400 2010 -1
RT2*
9 2167 7 802 808 847 201 1 202 1
       1 1400 2168 -1
PUL
9 2168 4 804 844 846 604 1 1400 2010 -1
FIN
```

ANNEXE III

Tableau des branches de l'algorithme d'interpretation
du MC 6800 realisation Mealy

No.	longueur de séquence	état éclaté de début de séq.	condition de l'éclatement	1-er état de séquence	2-ème état	3-eme état	4-eme état	5-eme état	6-eme état	7-eme état
0	7	2034	1727	2035	2036	2037	2038	2039	2040	2007
1	5	2016	1705	2024	2025	2026	2027	2028		
2	3	2010	1703	2045	2046	2007				
3	3	2010	1703	2048	2050	2010				
4	3	2010	1703	2068	2069	2010				
5	3	2010	1703	2059	2062	2010				
6	3	2010	1703	2061	2066	2010				
7	3	2001	1702	2002	2006	2007				
8	3	2017	1706	2018	2019	2007				
9	3	2017	1706	2021	2023	2007				
10	3	2033	1711	2167	2168	2010				
11	3	2051	1715	2054	2055	2007				
12	3	2111	1720	2113	2124	2007				
13	2	2010	1703	2041	2042					
14	2	2010	1703	2130	2010					
15	2	2010	1703	2152	2010					
16	2	2014	1704	2015	2016					
17	2	2014	1704	2020	2007					
18	2	2017	1706	2022	2023					
19	2	2028	1708	2029	2030					
20	2	2028	1708	2032	2002					
21	2	2030	1709	2031	2032					
22	2	2042	1712	2047	2010					
23	2	2051	1715	2052	2007					
24	1	2010	1703	2069						
25	1	2010	1703	2033						
26	1	2016	1705	2017						
27	1	2010	1703	2043						
28	1	2010	1703	2044						
29	1	2007	1701	2013						
30	1	2010	1703	2051						
31	1	2010	1703	2109						
32	1	2010	1703	2125						
33	1	2010	1703	2013						
34	1	2030	1709	2030						
35	1	2007	1701	2010						
36	1	2033	1711	2034						

37	1	1	2001	1702	2001
38	2	1	2034	1727	2040
39	1	1	2013	1701	2007
40	2	1	2042	1712	2007
41	1	1	2043	1713	2046
42	2	1	2043	1713	2042
43	1	1	2044	1714	2046
44	2	1	2044	1714	2042
45	2	1	2013	1701	2013
46	1	1	2010	1703	2014
47	1	1	2058	1716	2059
48	2	1	2058	1716	2061
49	1	1	2109	1719	2110
50	2	1	2109	1719	2111
51	3	1	2109	1719	2112
52	4	1	2109	1719	2007
53	1	1	2110	1716	2059
54	2	1	2110	1716	2061
55	1	1	2111	1720	2015
56	2	1	2111	1720	2069
57	9	1	2010	1703	2058
58	1	1	2112	1715	2052
59	2	1	2112	1715	2054
60	1	1	2125	1722	2126
61	2	1	2125	1722	2015
62	1	1	2126	1723	2127
63	2	1	2126	1723	2128
64	3	1	2126	1723	2129
65	4	1	2126	1723	2007
66	1	1	2127	1724	2113
67	2	1	2127	1724	2069
68	1	1	2128	1716	2059
69	2	1	2128	1716	2061
70	1	1	2129	1715	2052
71	2	1	2129	1715	2054

Annexe IV

ARCHITECTURE DE LA PARTIE CONTROLE DU MC 68000

L'analyse des circuits presentés dans les deux chapitres suivants, est basée sur le décodage du masque T6E du microprocesseur MC68000.

1-Introduction:

L'architecture de la partie contrôle du MC68000 est présentée par la figure 4-1. Le coeur du système de contrôle est une rom de microprogrammes. La rom de contrôle d'une capacité de 31 K bits contient toutes les routines de micro programmes, pour interpreter les macro-instructions. Celle-ci est divisée en deux parties: la micro rom qui s'occupe de l'aspect de séquencement et contrôle des éléments de la partie contrôle, et la nano rom qui contient les profils des nano commandes pour la partie operative. La longueur d'une micro-instruction est de 17 bits et celle de la nano-instruction est de 68 bits. L'adresse de chaque micro instruction est constituée de 10 bits. Cette adresse peut être définie par les PLA de décodage des macro instructions ou la micro-instruction elle-même. En cas de séquencement 10 bits, parmi les 17 bits de chaque micro-instruction définissent, l'adresse de la micro-instruction suivante. Pendant le traitement des routines de micro-programmes, chacun des PLA: PLA-A1, PLA-A2 et PLA-A3, peut etre choisi par les champs de micro instruction. Dans ce cas là, le PLA fournit l'adresse de micro-programme suivant. Le PLA-A1 calcule l'adresse de la première micro-instruction au début de chaque macro-intruction. Le séquencement se déroule à partir de cette adresse, par chargement de l'adresse de micro-instruction suivante dans le registre de micro-adresses.

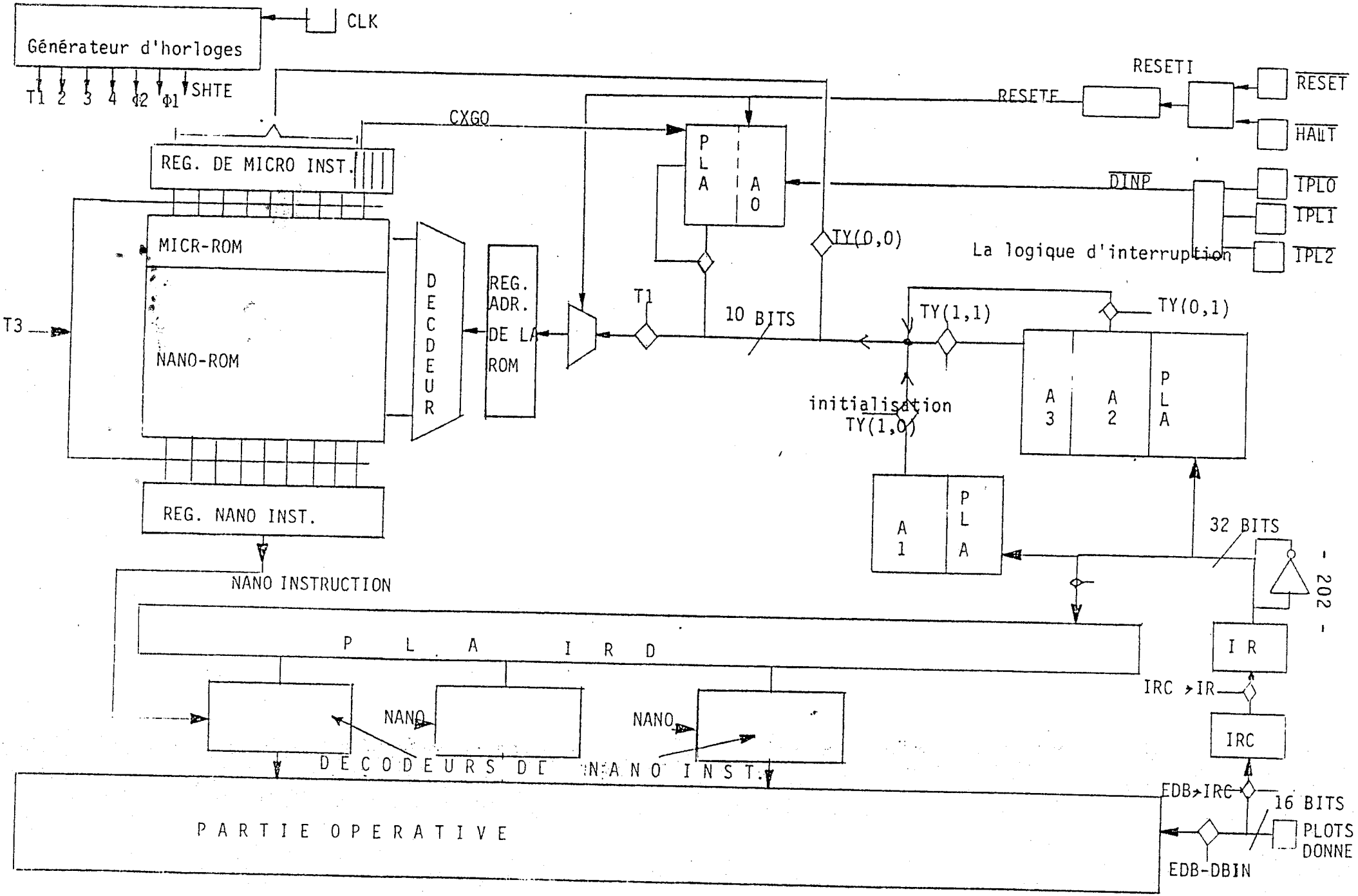

Générateur d'horloges
CLK
T1 2 3 4 Φ2 Φ1 SHTE
RESETI
RESETE
RESET
HALT
CXGO
REG. DE MICRO INST.
P L A A 0
DINP
IPL0
IPL1
IPL2
La logique d'interruption
MICR-ROM
NANO-ROM
DECODEUR
REG. ADR. DE LA ROM
T3
T1
10 BITS
TY(0,0)
TY(1,1)
TY(0,1)
initialisation TY(1,0)
A 3
A 2
P L A
A 1
P L A
32 BITS
REG. NANO INST.
NANO INSTRUCTION
P L A I R D
I R
IRC >IR
IRC
NANO
NANO
DECODEURS DE NANO INST.
EDB>IRC
16 BITS
PLOTS DONNE
EDB-DBIN
PARTIE OPERATIVE
- 202 -

Figure - 4-1

Plusieurs PLA sont employés, pour fournir l'adresse des micro-programmes. Le PLA-A0 s'occupe de la prise en compte des exceptions et fournit l'adresse des routines de traitement des excepti en cas de branchements.

La logique de contrôle de bus s'occupe d'une façon autonome, du contrôle des entrées/sorties du systéme.

Les générateurs d'horloges contrôlent le déroulement des séquences. La logique d'interruption traite les demandes d'interruptions, pour générer le signal de reconaissance d'interruption.

2-Traitement d'une macro-instruction

Le code opération est chargé dans le tampon d'entrée (IRC) à partir des plots de données. La commande de chargement du registre d'instruction (IR), par le bit 0 de la micro-instruction, envoie 16 bits directs et 16 bits complémentés aux PLA. L'adresse du micro-programme de traitement de cette instruction est calculée par les PLA. Cette adresse est envoyée au décodeur de la micro-rom et la séquence de traitement se déroule par la suite. Selon le type d'instruction, chacun des PLA de A1,A2 ou A3 peut être choisi pour fournir l'adresse suivante. A la fin de chaque traitement, le PLA-A1 est adressé par la micro-instruction de la fin de séquence en cours pour charger le nouveau code-operation.

L'organigramme d'execution des routines de contrôles par la micro-rom est présenté par figure 4-2.

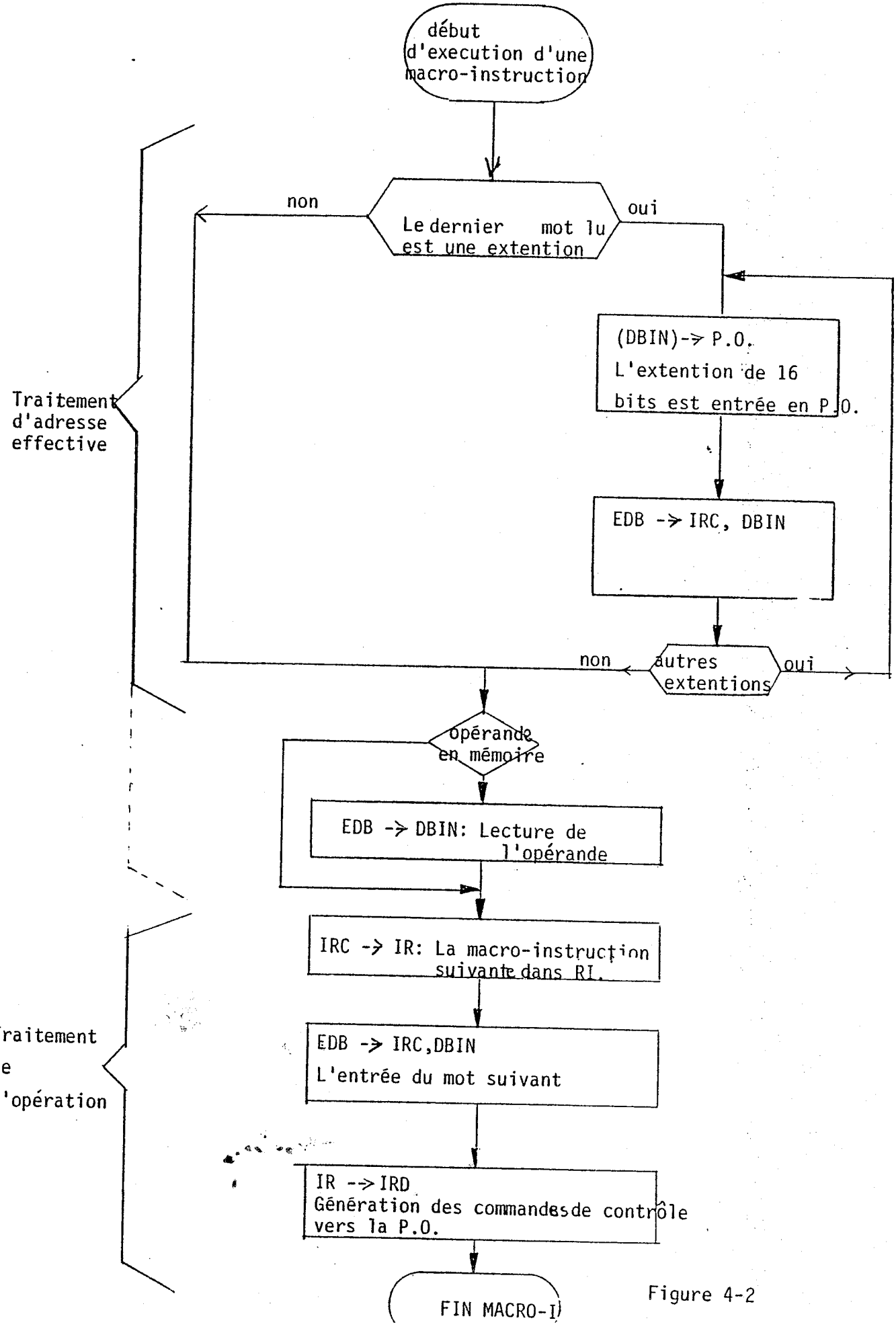
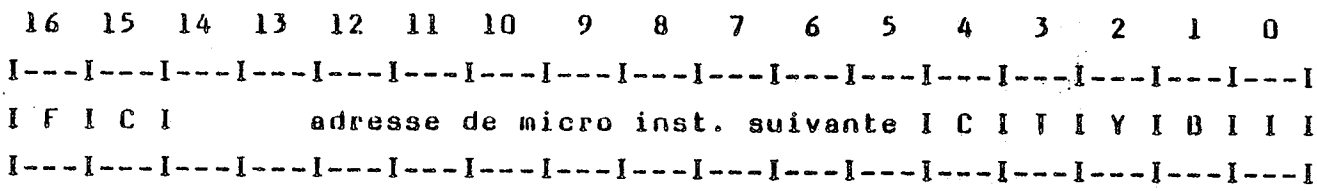

début
d'execution d'une
macro-instruction
non
Le dernier mot lu
est une extention
oui
(DBIN)-> P.O.
L'extention de 16
bits est entrée en P.O.
EDB -> IRC, DBIN
non
autres
extentions
oui
Traitement
d'adresse
effective
opérande
en mémoire
EDB -> DBIN: Lecture de
l'opérande
IRC -> IR: La macro-instruction
suivante dans RI.
EDB -> IRC,DBIN
L'entrée du mot suivant
IR -> IRD
Génération des commandes de contrôle
vers la P.O.
Traitement
de
l'opération
FIN MACRO-I

Figure 4-2

3-Structure des micro-instructions

Chaque micro-instruction comprend les commandes nécessaires pour contrôler les éléments de la partie contrôle. Ces commandes sous forme codée, constituent les différents champs de chaque micro-instruction. les bits des micro-instructions effectuent les opérations suivantes:



Bits 15,16

0 , 1 Processeur en mode donnée

1 , 0 Processeur en mode programme

1 , 1 Acquittement d'interruption

Bits 5 à 14

adresse de la micro-instruction suivante

Bit 4 (C)

Commande de chargement du PLA AO
permet de charger les procedures d'exceptions

Bits 2,3 (T,Y)

Choix de la source de l'adresse du micro-programme à partir du tableau suivant:

bit 2	3	source de la micro-adresse
0	0	l'adresse contenue dans les bits 5 à 14
0	1	PLA A1
1	0	PLA A2
1	1	PLA A3

Bit 1 (B)

Choix de branchement (B=1) ou non (B=0)

Bit 0 (I)

Commande de chargement du registre de micro-instruction (IR) si si (I=1)

Remarque:

En cas de bit B=1 l'intérprétation des bits 2 à 14 est modifiée pour les traitement des branchements, de la façon suivante:

suivante. le complément d'adresse est fourni dans ce cas par le PLA de branchement.

4-Le générateur d'horloges interne

Le système de contrôle est synchronisé par les impulsions des horloges. Ces horloges sont modifiées par la reconnaissance du signal reset. Cette modification est en effet, le moyen de priorité du signal reset sur le fonctionnement de la partie contrôle. Pour cette raison, l'horlogerie de MC68000 sera etudiée dans cette partie.

L'application du signal d'horloge (CLK), à la broche 15 du processeur commande le générateur d'horloge interne.

L'analyse du fonctionnement des générateurs d'horloge est présentée dans cette partie et le système de contrôle synchronisé par les horloge est étudié par la suite.

5-Les signaux d'horloges

Deux niveaux de générations d'horloges, peuvent être distingués en cas de fonctionnement normal. Le premier, commandé par signal CLK, génére deux signaux non recouvrants PHI-1 et PHI-2. Ces deux signaux ne sont pas affectés par le signal reset ils synchronisent constamment les differents blocs de la partie contrôle. Un deuxième niveau de signaux d'horloge, génèré à partir de PHI-1 et PHI-2, constitue quatre instants conséqutifs T1-->T4. La génération de ces horloges est affectée par lesignal reset.

La reconaissance du signal reset démarre l'hologerie d'exceptions. Celle-ci xprovoque l'arrêt de la génération des horloges secondaires (T1, T2, T3) et la modification de la génération de T4 (T4 identique à PHI-2). Un signal d'horloge SHTE est généré, au moment du bloquage des horloges.

Donc six signaux synchronisent le déroulement du processeur en cas de fonctionnement normal. Quatre horloges PHI-1 et PHI-2, T4 et SHTE sont utilisés en cas d'arrêt du fonctionnement normal et pendant les traitement d'exceptions.

5-1-Générateurs de PHI-1 et PHI-2

La génération des signaux PHI-1 et PHI-2 est obtenue à partir de deux signaux intermédiaires H1 et H2. Le circuit de la figure 4-3 presente les générateurs de H1 et H2 commandés par le signal CLK. On remarque deux inverseurs 2 et 3 qui tansmettent et amplifient le signal CLK pour les portes de puissance en sortie. Les inverseurs 5 et 6 constituent les push pulls de sorties. Le transistor T1 sert à la protection du plot, CLK.

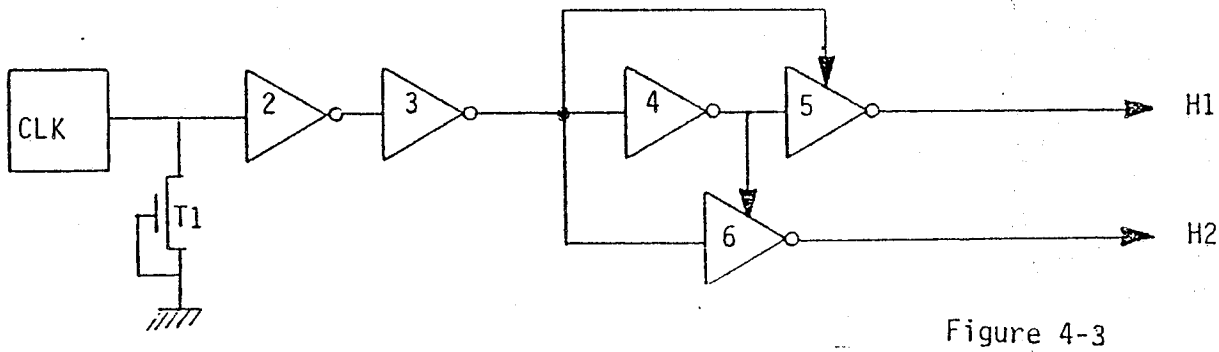


Figure 4-3

Figure 4-3, les générateurs de H1 et H2

H1 et H2 commandent les blocs de génération de PHI-1 et PHI-2. Quatre blocs identiques sont employés pour la génération de ces dernières. Le circuit de figure 4-4 present l'assemblage du bloc de générateur de PHI-1.

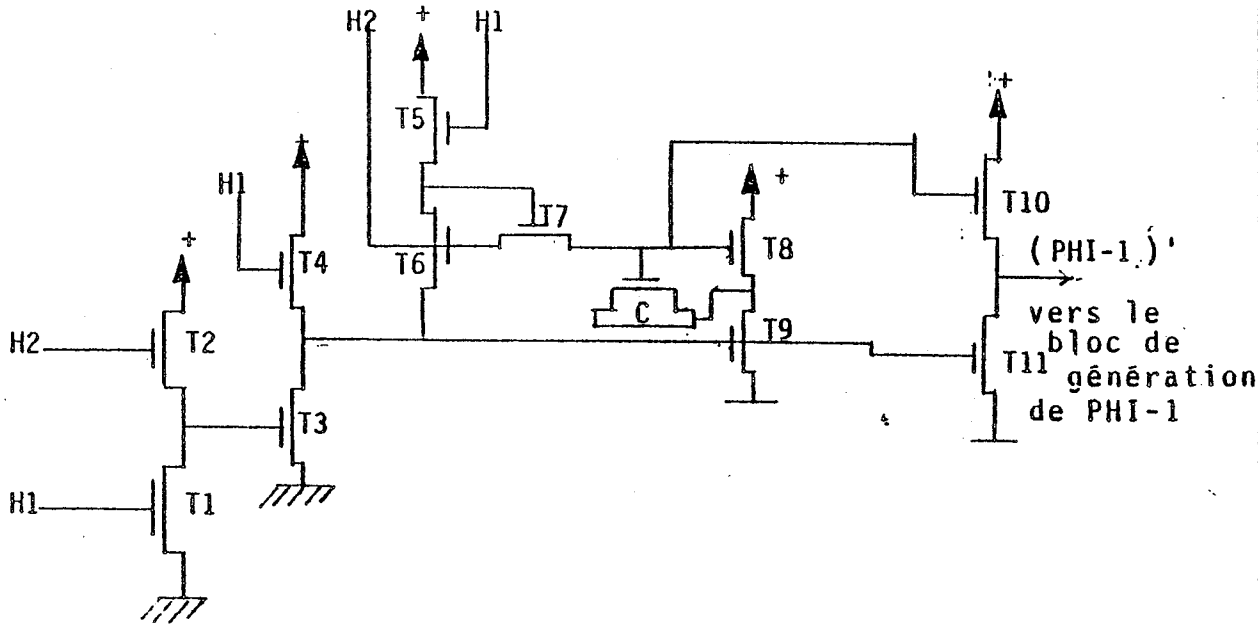


Figure 4-4, Le bloc de génération de (PHI-1)'

Ce circuit, impose le temps de non-recouvrement entre (PHI-1)' et (PHI-2)'. Les trransistors T-1 et T-2, commandés par H1 et H2 constituent la première porte de puissance. Les transistors T3 et T4 sont les buffers d'entrée pour le reste du circuit. On trouve le complément du signal H1 appliqué au transistor T5 sur sa source. Celui-ci commande le transistor T7 qui comme une resistance variable contrôle la charge et la décharge de la capacité C. Donc la resistance T7 est modulé par les signaux H1 et H2. Pendant le chargement de C, cette resistance a une impédance trés élevée, ce qui est validée pendant H1. Lorsque H1 est à zero l'impédance faible de T7 décharge la capacité C. Les deux transistors T10 et T11 génèrent le PHI-1 par la porte d'entrée (transistors:T3 et T4).

Le pull-up T10, est commandé par la capacité C. Par conséquent la monté de (PHI-1)' est retardée par commande de la grille de T10. Le chronogramme suivant montre le rapport entre H1 et les signaux de sorties (PHI-1)' et (PHI-2)'. (PHI-2)' est généré à partir d'un bloc identique où seule les entrées de H1 et H2 sont inversées.

L'ensemble des circuits de génération de PHI-1 et PHI-2 ainsi que leurs chronogrammes à partir du plot CLK est presenté dans la figure 4-5.

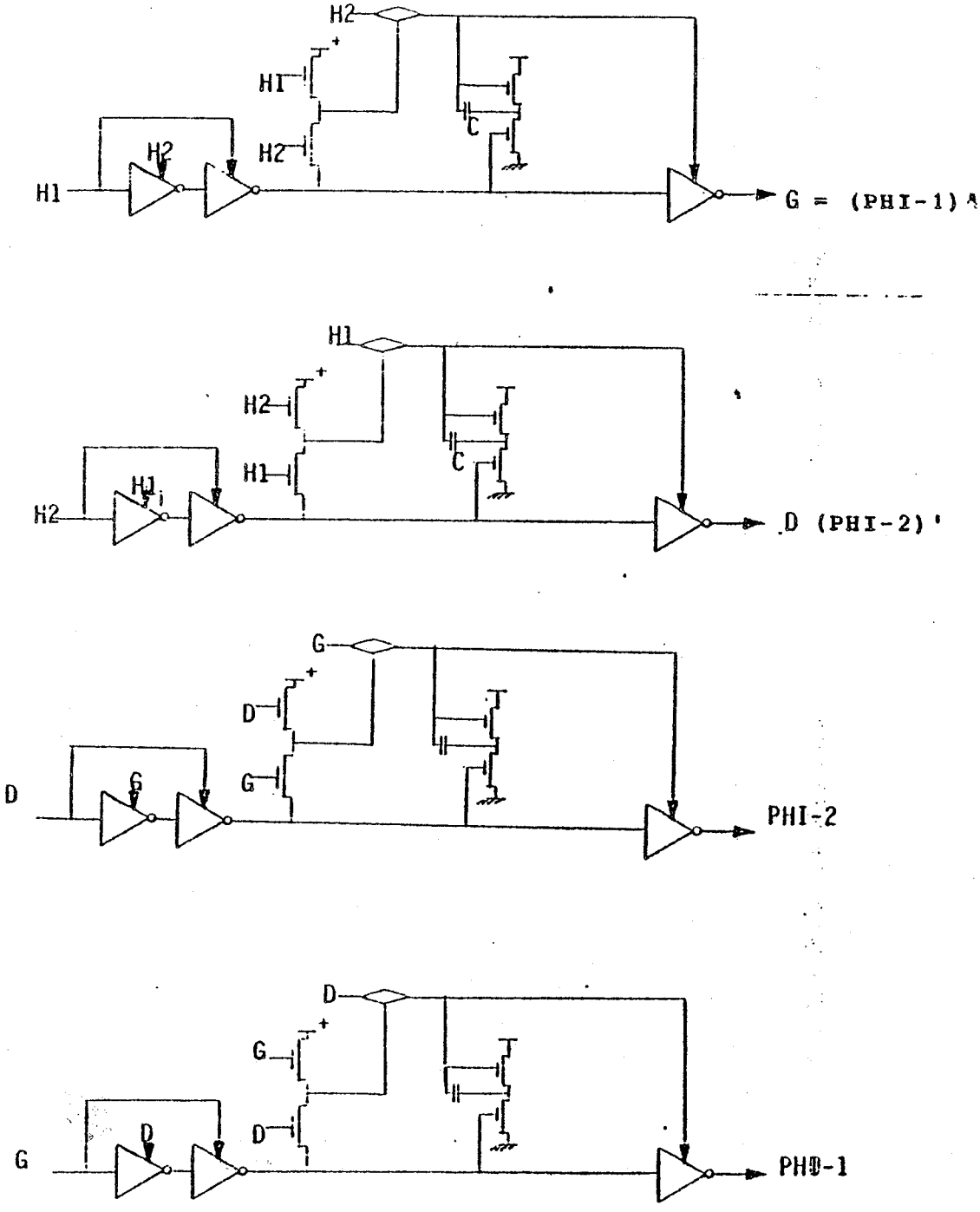


Figure-4-5 -A Génération de PHI-1 et PHI-2 par H1 et H2

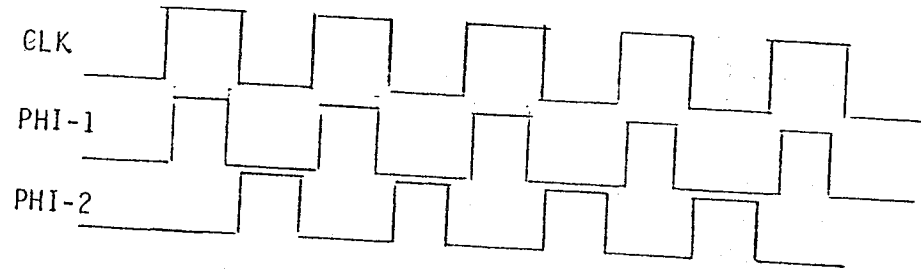


Figure 4-5, Générateur d'horloge primaire

5-2- La génération des horloges secondaires

Le fonctionnement du système de contrôle, et le séquencement de la Rom de microprogrammes est synchronisé sur les cadences des quatre horloges secondaires, T1,T2,T3,T4. Ces horloges sont issues de deux signaux de bases PHI-1 et PHI-2. L'étudedétailée de ces dernières aunen importance particulière, car l'intervention de ces horloges, est affectée par le reset. Cet effet est réalisé par l'arrêt de génération de T1,T2 et T3 et la modification de la génération de T4. Pour détailler un tel fonctionnement, le circuit de génération des horloges secondaires et sa logique de contrôle, est donné à la figure 4-6. Ce circuit est composé de trois compteurs CPTR1, CPTR2 et CPTR3. Chaque compteur est constitué de deux cellules de mémorisation, commandées par PHI-1 et PHI-2. La sortie de CPTR1 (S1) active l'entrée de CPTR2 (E2). Par conséquenct le troisième bloc CPTR3, est commandé par la sortie S2 de CPTR2. Les quatre signaux de bases sont générés à partir des sorties de ces compteurs. La sortie S1 commande le bloc de génération de T1. CPTR2 avec deux sorties: directe et compléméntée, active les

générateurs de T2 et T4. Eventuellement T3 est générée à partir de sortie S3 de CPTR3. Les éléments de mémorisation des compteurs sont mis à zero par reset. Le bloquage de ces compteurs est faite par la mise à zero des NOR 2 et 4 de chaque compteur.
Dans ce cas là, tant que reset est actif, les sources S1,S2,S3 de T1, T2 et T3 sont mises à zero. Pendant le reset S2 à zero, S2' est mis à "1" ce qui active constamment le bloc de génération de T4. Une copie de PHI-2 est générée sur la sortie de ce bloc.

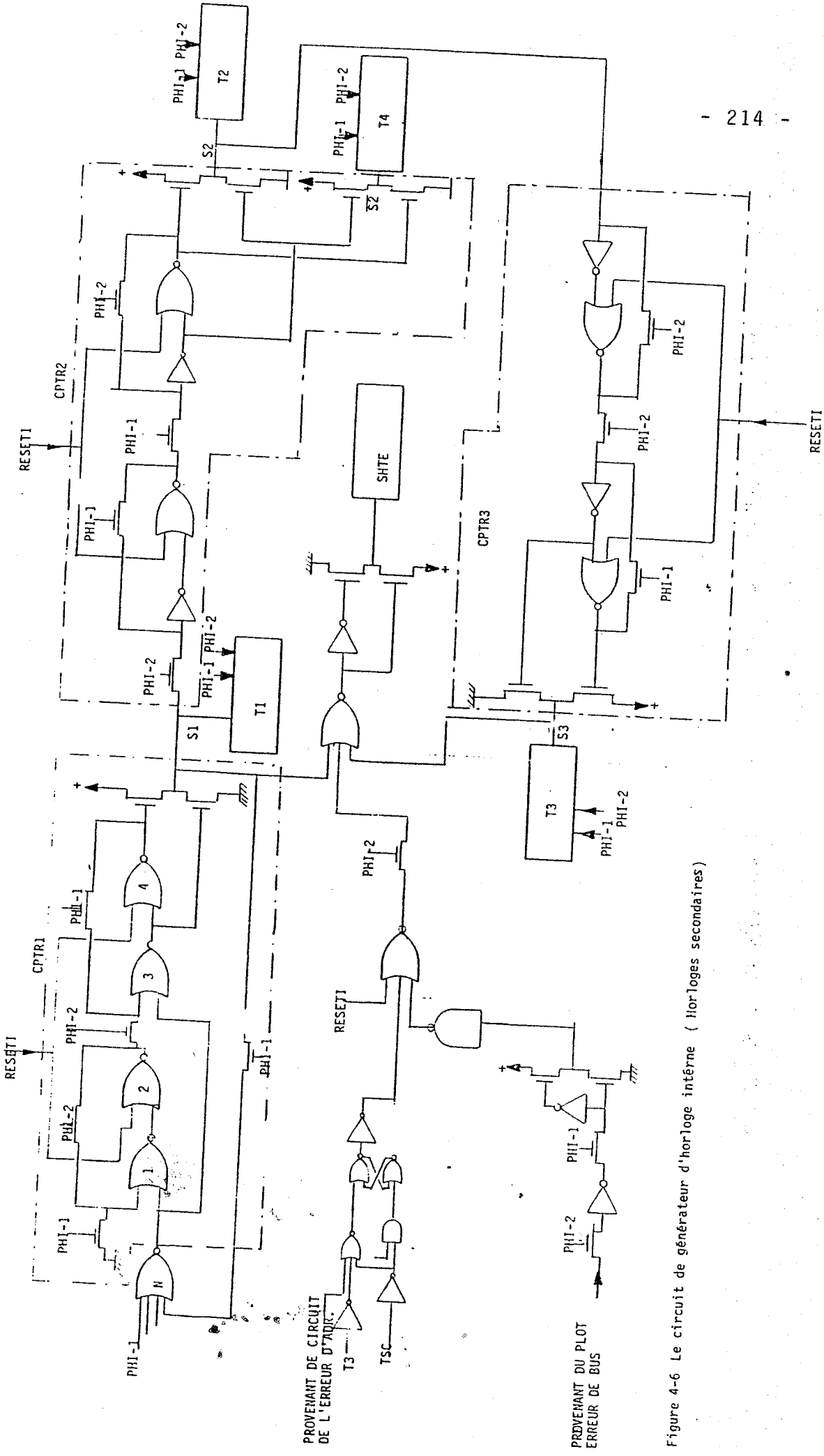
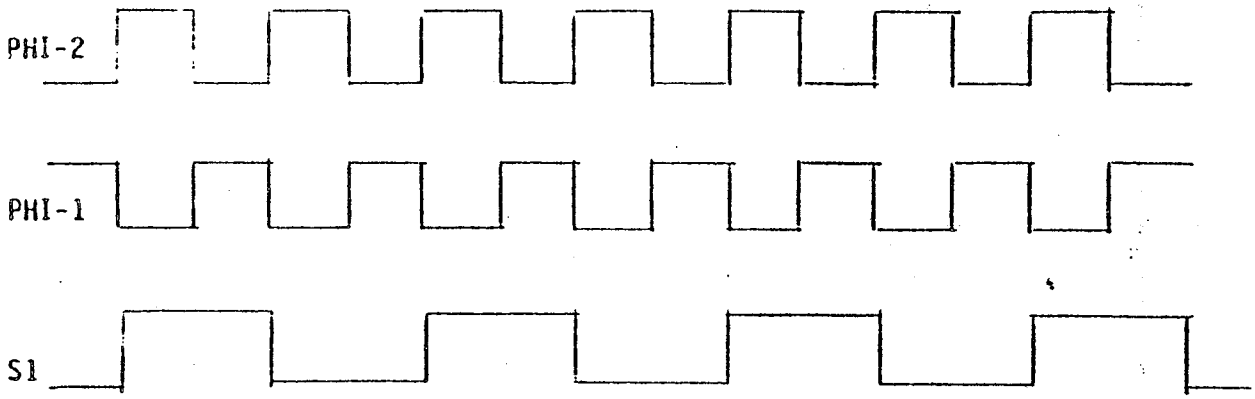


Figure 4-6 Le circuit de générateur d'horloge intérne (Horloges secondaires)

Le compteur CPTR1 génére le signal S1 à partir du NOR multi-entrées (N). PHI-2 est la commande d'entrée principale, et en cas d'absence de reset interne (signal RESETI à zero), CPTR1 génére le chronogramme de sortie suivant:



Chronogramme de sortie S1

Ce compteur est composé de deux étages. Le premier mémorise l'entrée sur PHI-2 et le deuxième, prolonge son entrée par la phase PHI suivante, ce qui explique le chronogramme de S1.

Les circuits de figure 4-7 presentent les générateurs de puissance T1 -> T4. La même capacité de retard, utilisée dans les blocs de génération de PHI-1 et PHI-2 est employée pour le non recouvrement des instants T1,T2,T3 et T4.

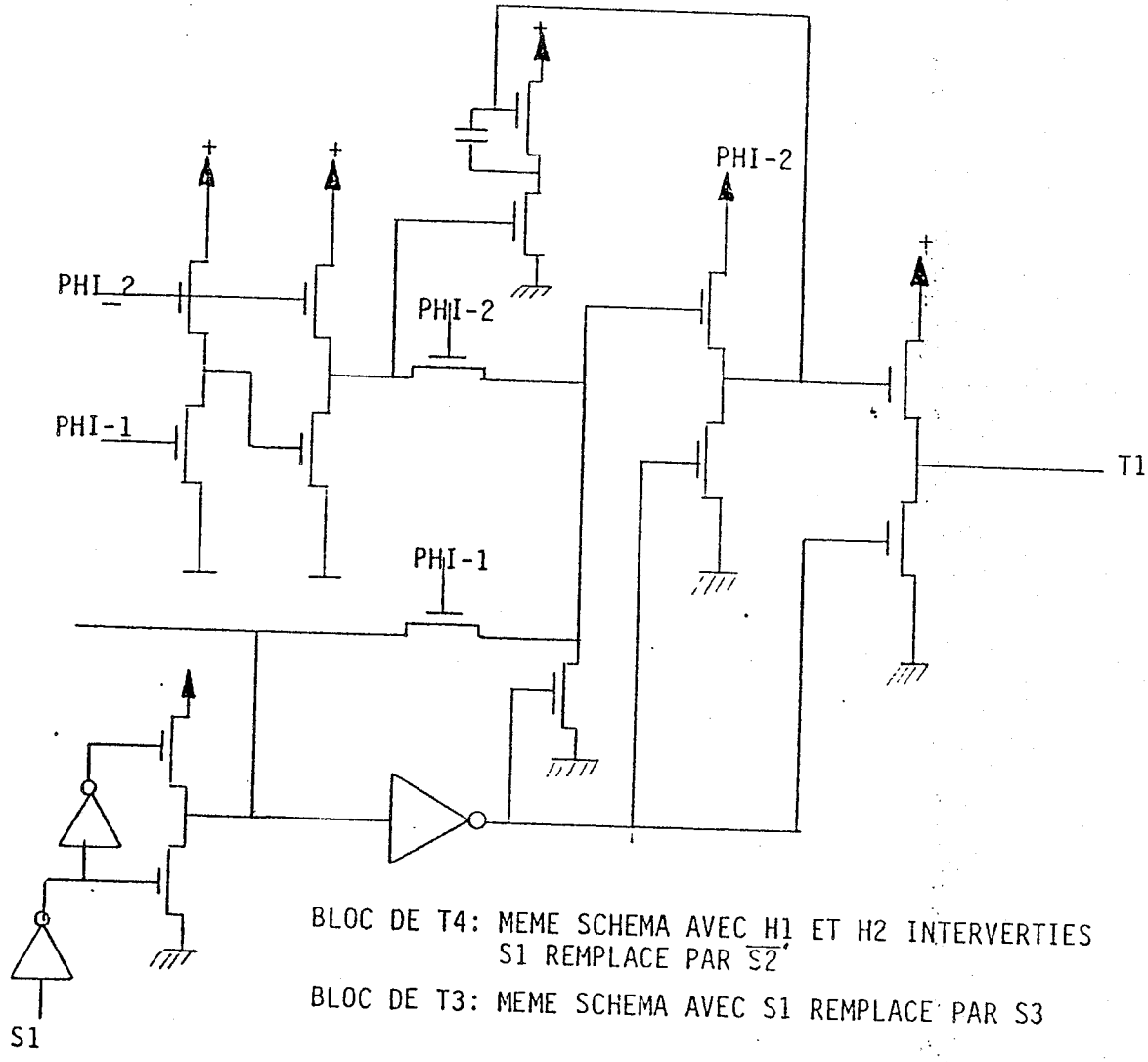


Figure 4-7- Le circuit de génération des horloges secondaire

L'analyse du fonctionnement des blocs de génération de T1->T4 donne les chronogrammes de figure 4-8.

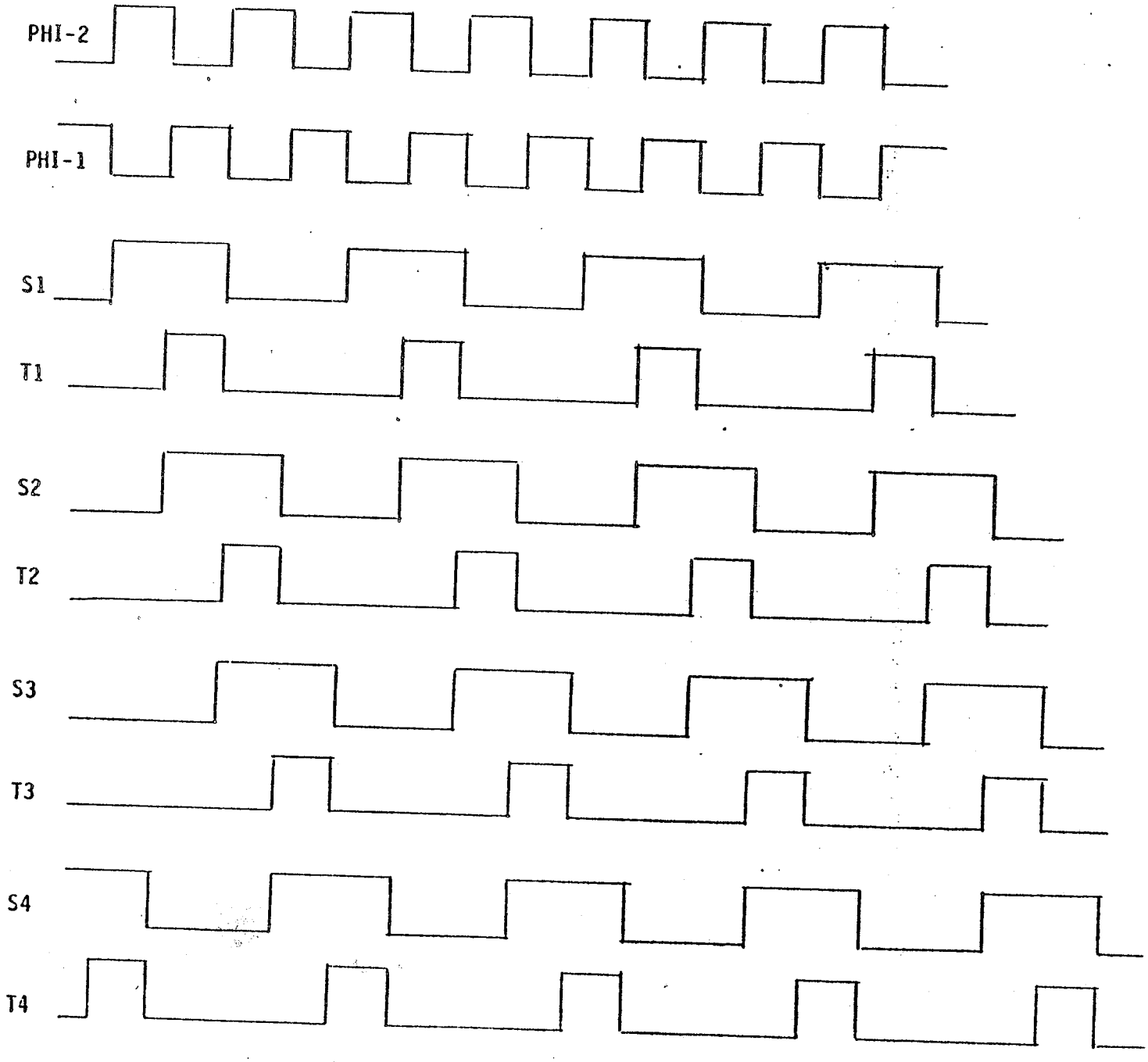


Figure 4-8-chronogrammes des horloges secondaires

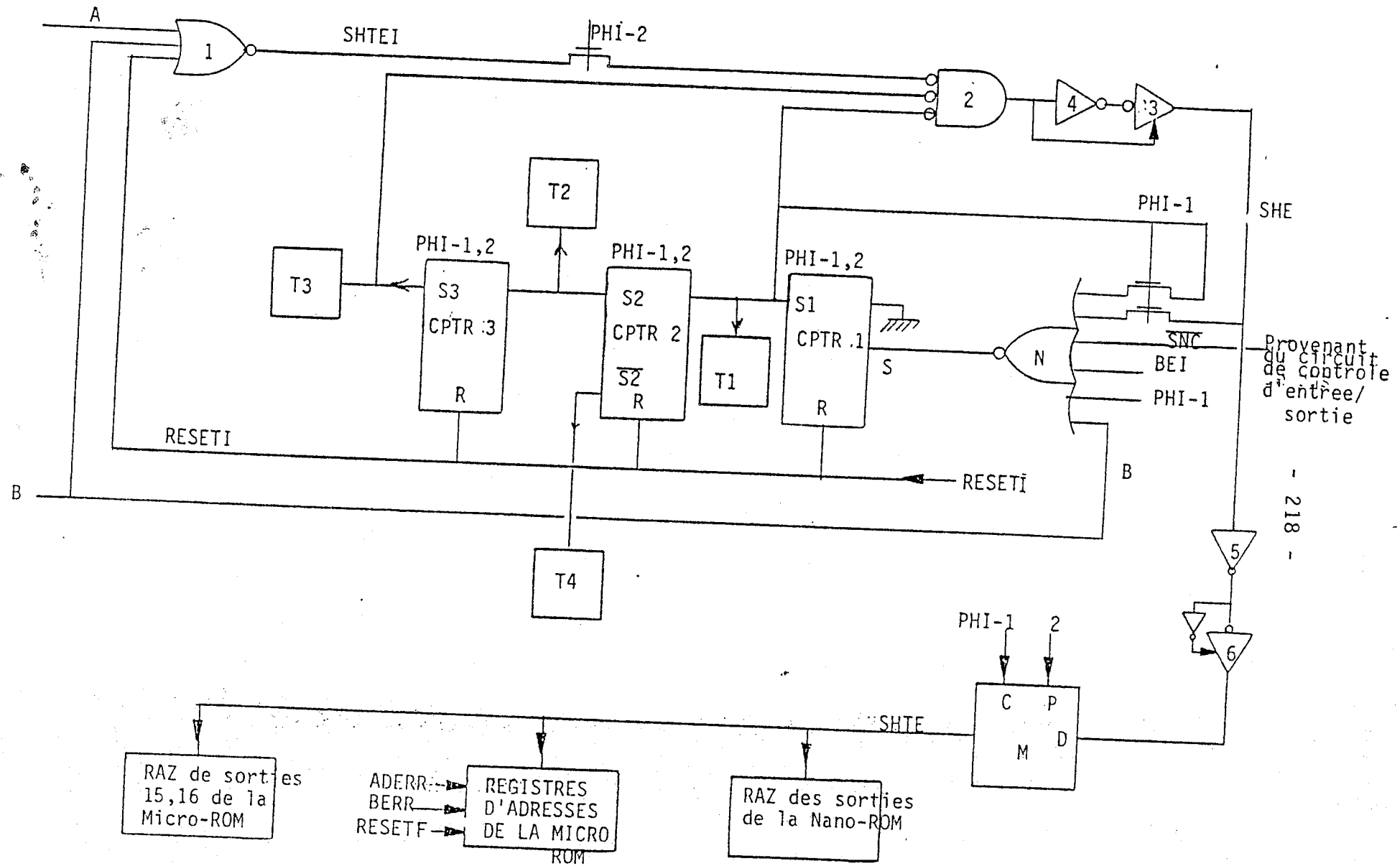

A
1
SHTEI
PHI-2
2
4
3
T2
T3
PHI-1,2
S3
CPTR :3
R
PHI-1,2
S2
CPTR 2
S2
R
T1
PHI-1,2
S1
CPTR .1
R
S
PHI-1
SHE
N
SNC
BEI
PHI-1
Provenant du circuit de controle d'entrée/ sortie
RESETI
RESETI
B
B
T4
5
6
PHI-1
2
C
P
M
D
SHTE
RAZ de sorties 15,16 de la Micro-ROM
ADERR
BERR
RESETF
REGISTRES D'ADRESSES DE LA MICRO ROM
RAZ des sorties de la Nano-ROM

Figure 4-9

6-L'horlogerie en cas de reset

Les circuits des figures 4-6 et 4-7 sont présentés par le schéma de figure 4-9. Le signal RESETI appliqué aux compteurs génère un signal ternaire par la porte NOR-1. La sortie de cette porte est mise à "0" si l'une des exceptions du groupe zero est signalée à l'entrée. Ces exceptions correspondent au reset, à l'erreur de bus pour l'entrée A et l'erreur d'adresse pour l'entrée B du NOR-1. Ces commandes permettent de générer la sortie SHTEI. Les signaux ADERR et BERR sont conditionés avant l'activation de A et de B. Seul RESETI peut activer SHTEI sans aucune contrainte. La porte AND-2 transmet la commande SHTEI aux portes de puissance 3 et 4. Ces dernières sont affectées si les sorties S1 et S3 des compteurs sont au repos. Autrement dit si les horloges sont arrêtées la porte AND-2 génére le signal SHE à partir de SHTEI. SHE commande le bloc "H", qui génére le signal de l'horlogerie d'exception, "SHTE". Ce bloc est constitué du même circuit que le bloc de génération de T1. Seules les commandes d'entrées sont changées. Celui-ci est presenté par la figure 4-10-A. Le chronogramme correspondant est presenté à la figure 4-10-B. Le signal SHTE, deuxième signal d'horloge d'exceptions prend en charge les affectations qui imposent une autre priorité de fonctionnement que le groupe d'exception "0".

Les affectations de SHTE se composent de:

1- Mise à zero des sorties 15 et 16 de la micro rom de contrôle.

2- Chargement de l'adresse d'exceptions (groupe 0) dans la rom de micro-programmes.

3- Remise à zero des sorties de la nano-rom.

Les détails de ces affectations seront analysés dans les autres parties.

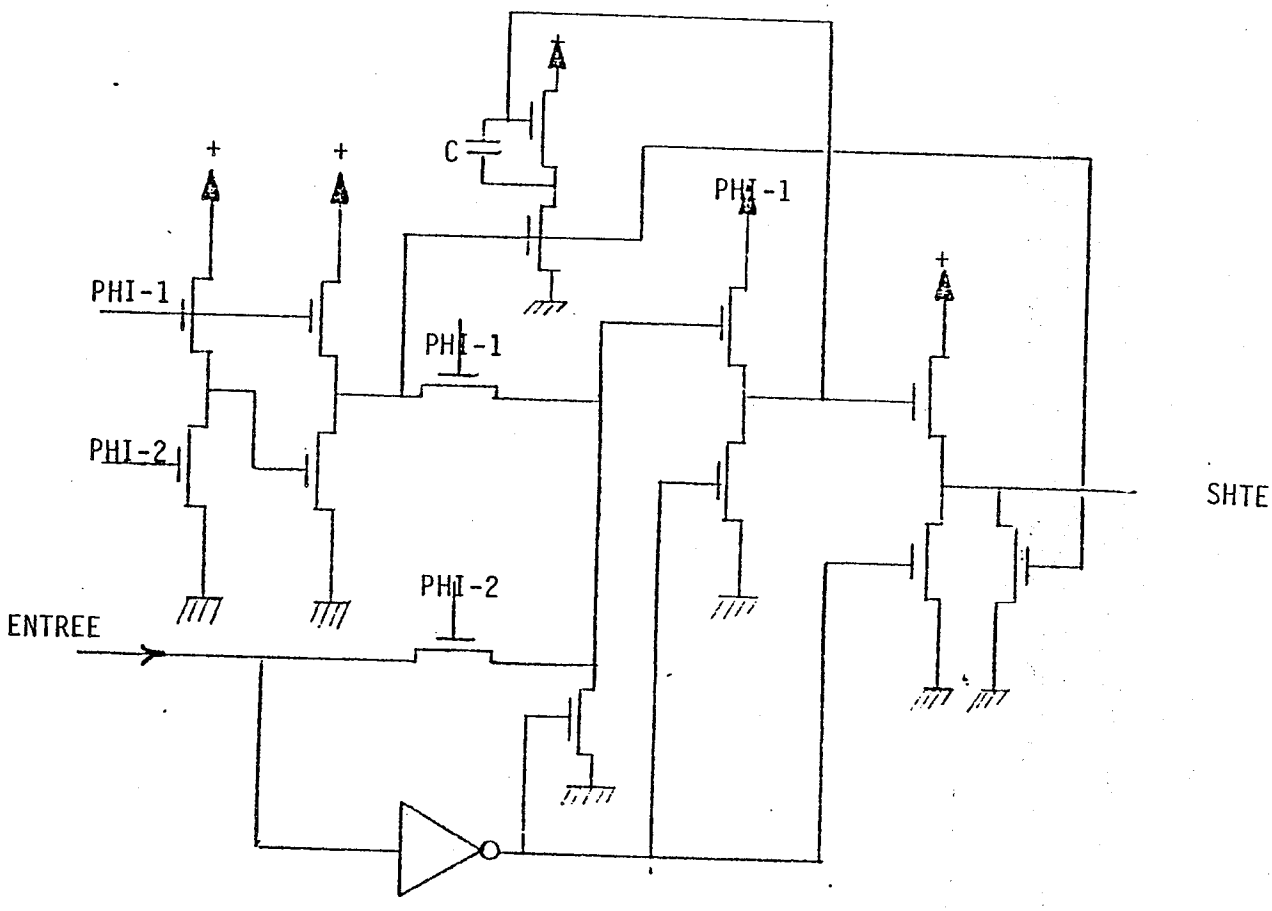


Figure 4-10-A Circuit du bloc de génération de SHTE

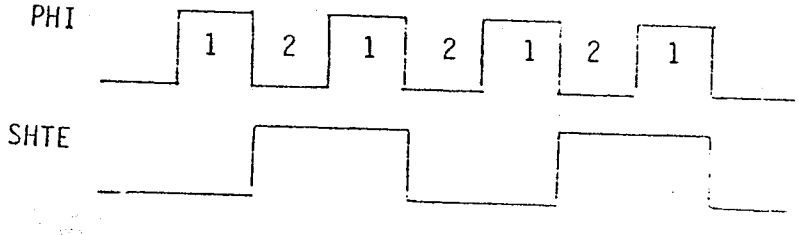


Figure 4-10-B-Chronogramme de SHTE

7- Fonctionnement des horloges secondaires en cas de reset.

En cas de reset les sorties des compteurs CPTR1 -> 3 sont bloquées, seule la sortie S2' de CPTR2 est mise à "1" par RESETI qui génére le T4 d'une manière modifée. Dans ce cas le bloc de génération de T4, génére une copie de PHI-2 sur la sortie T4, qui remplace le signal T4.

La mise à zero de la broche reset bloque le processeur par la modification des horloges. Seuls les circuits associés avec instants T4 et SHTE fonctionnent. Le MC68000 reste en état bloqué pendant toute la durée de l'excitation de la broche reset.

Le signal SHTE, chargé d'imposer l'état d'exception du groupe "0", permet l'initialisation des séquences de reset, l'erreur de bus et d'adresse. La disparition de signal RESETI libére les générateurs d'horloges pour reprendre le fonctionement normal. Les chronogrammes d'arrêt, de modification et de redémarrage des circuits de l'horloge, sont présentés par figure 4-11. On constate que la durée d'arrêt de l'horlogerie interne dépend de la durée de l'excitation de la broche reset.

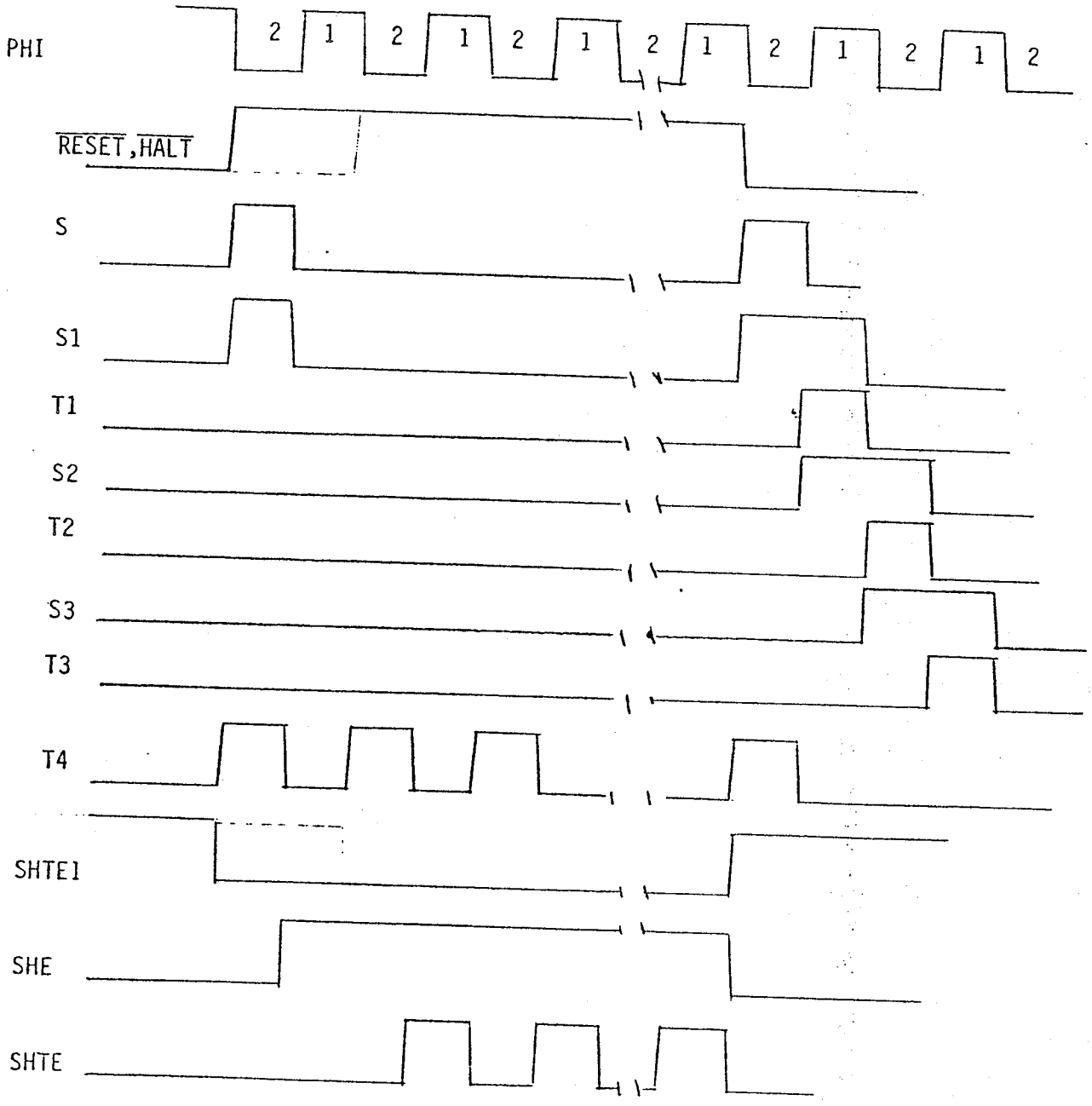


Figure- 4-11
Modification de génération des instants, T1→ T4 et horlogerie du commande SHTE

8-Conclusion sur l'horlogerie du MC68000

Le fonctionnement des générateurs d'horloges peut être classé en deux modes. Le fonctionnement normal tel qui est decrit dans la première partie et le fonctionnement en cas d'exceptions. Dans ce cas-là SHTC et T4-modifié, remplacent le système d'horloges secondaires. Ces signaux traitent la propagation des exceptions du groupe "0", de plus un forçage à l'état halt est autorisé par ces horloges.

9-Synchronisation de la rom de contrôle par les horloges secondaires

Les instants T1 --> T4 synchronisent le séquencement. T1 charge l'adresse du micro-programme dans le registre d'adresse de la micro-rom, et T3 valide la sortie des 17 bits sur le registre de micro-instructions. La durée de traitement d'une micro-instruction est de deux cycles d'horloge. L'adresse de la micro-instruction suivante provenant des PLA-A1 --> PLA-A3, ou du champ de l'adresse suivante de la micro-instruction, est chargée sur T1. La période T1-T2 est nécessaire pour la préchage de la rom et le décodage de l'adresse. Les sorties validées sur T3 débutent un autre cycle de recherche de micro-instructions. Le chronogramme de la figure 4-12 représente la synchronisation de la lecture de la rom de contrôle par les impulsions des horloges secondaires.

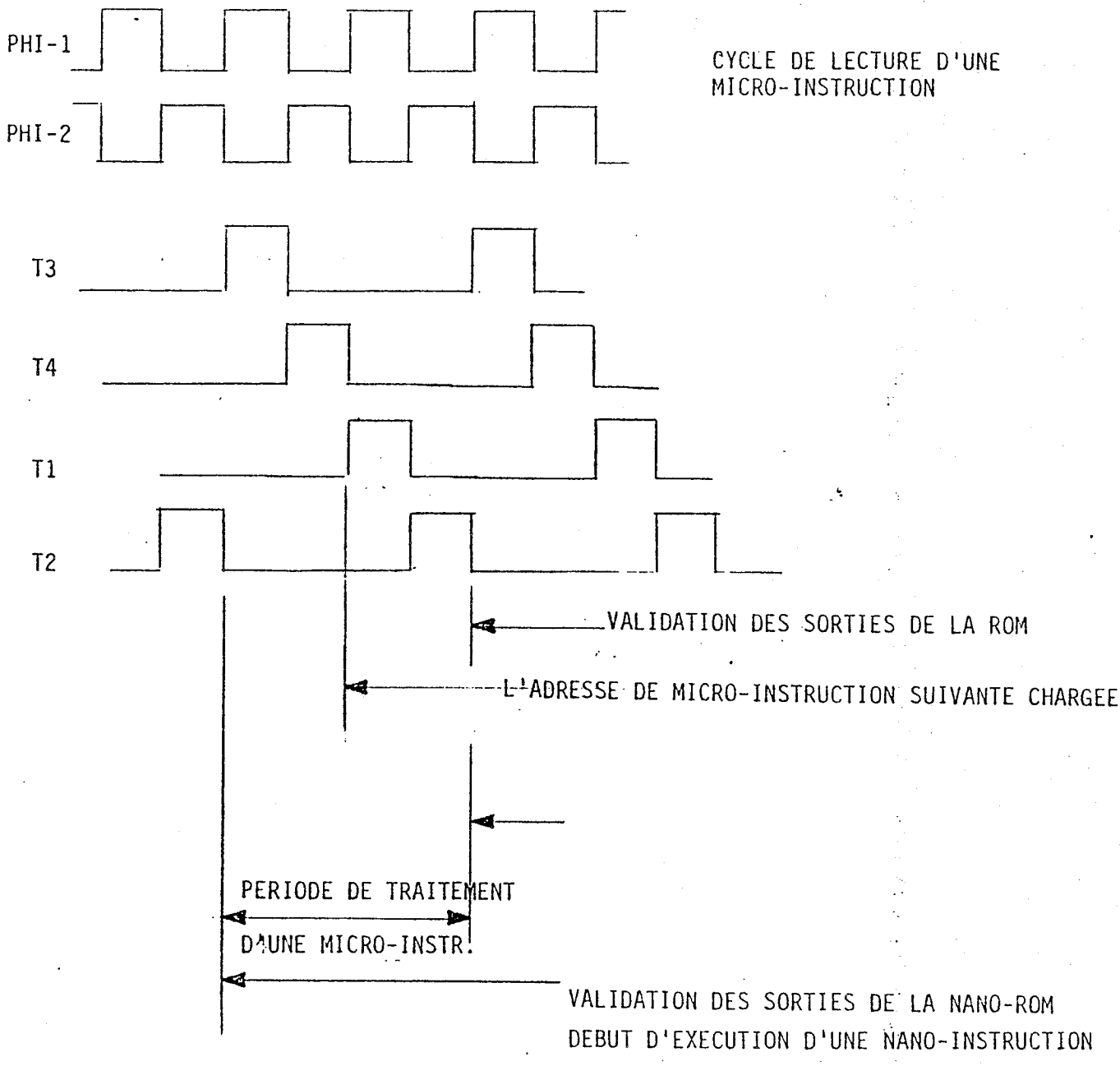
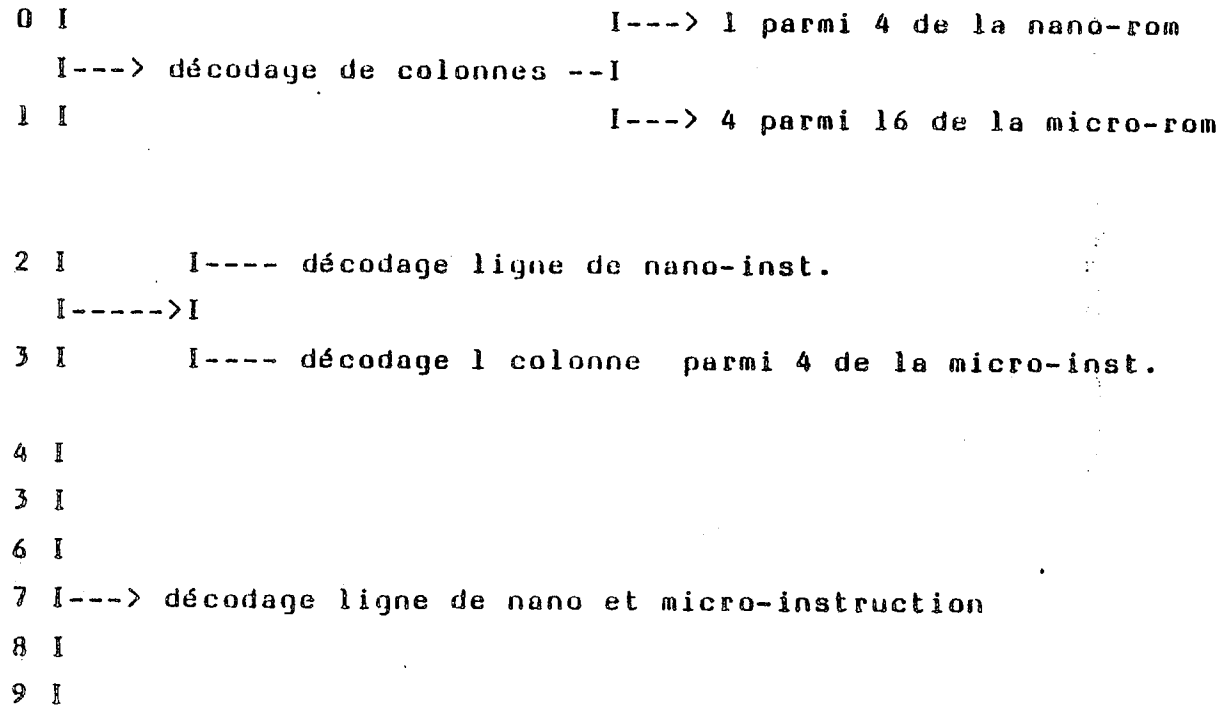


Figure 4-12 Synchronisation de la rom par l'horlogerie secondaires

L'intervention des horloges secondaires, dans le mécanisme de séquencement nécessite l'étude detaillée du décodage des lignes et des colonnes des roms.

10-Les décodeurs de la rom

Dix bits d'adresse de la rom sont regroupés pour la sélection des lignes et des colonnes des micro et nano rom. Ce regroupement est presenté par la suite:



10-1 Décodage des lignes de la micro-rom

La séléction d'une micro-instruction de 17 bits stockée dans une ligne de 272 bits est effectuée dans l'ordre suivant:

1- séléction d'une ligne de 272 bits.
2- décodage de 4 colonnes parmi 16.
3- décodage de 1 colonne parmi 4.

De cette façon on, décode 272 bits en 68 bits et eventuellement en 17 bits.

10-2-Décodage des lignes de la nano-rom

Quatre nano-instructions de 68 bits sont décodées à partir du décodage d'une ligne d'une colonne parmi 4 .

Chaque partie de la rom (la nono et micro rom) posséde un décodeur de colonnes. Les deux premiers bits d'adresses (A0 et A1) éffectuent une séléction de 4 colonnes parmi 16 pour la micro-rom et 1 parmi 4 pour la nano-rom.

11-Mécanisme d'adressage des rom

L'état "1" ou "0" de chaque bit de rom défini une micro-commande. La mise à 1 de chaque bit est réalisée par l'implantation d'un transistor. L'absence de transistors provoque le mise à zero du bit de sortie.

Les micro et nano-instructions sont adressées simulanement à partir des 10 bits d'adresse du micro programme.

La figure 4-13 montre la topologie de la rom qui est adaptée à la manière d'adressage et à la longueur des champs des micro et nano-instructions. La rom est divisée en deux parties, droite et gauche avec les décodeurs de lignes au millieu. Le masque de la rom contient 34 lignes pour la micro rom et 84 lignes pour la nano rom.

La capacité de 31 K bits pêut être calculée à partir des longueurs des champs de micro et nano instructions.

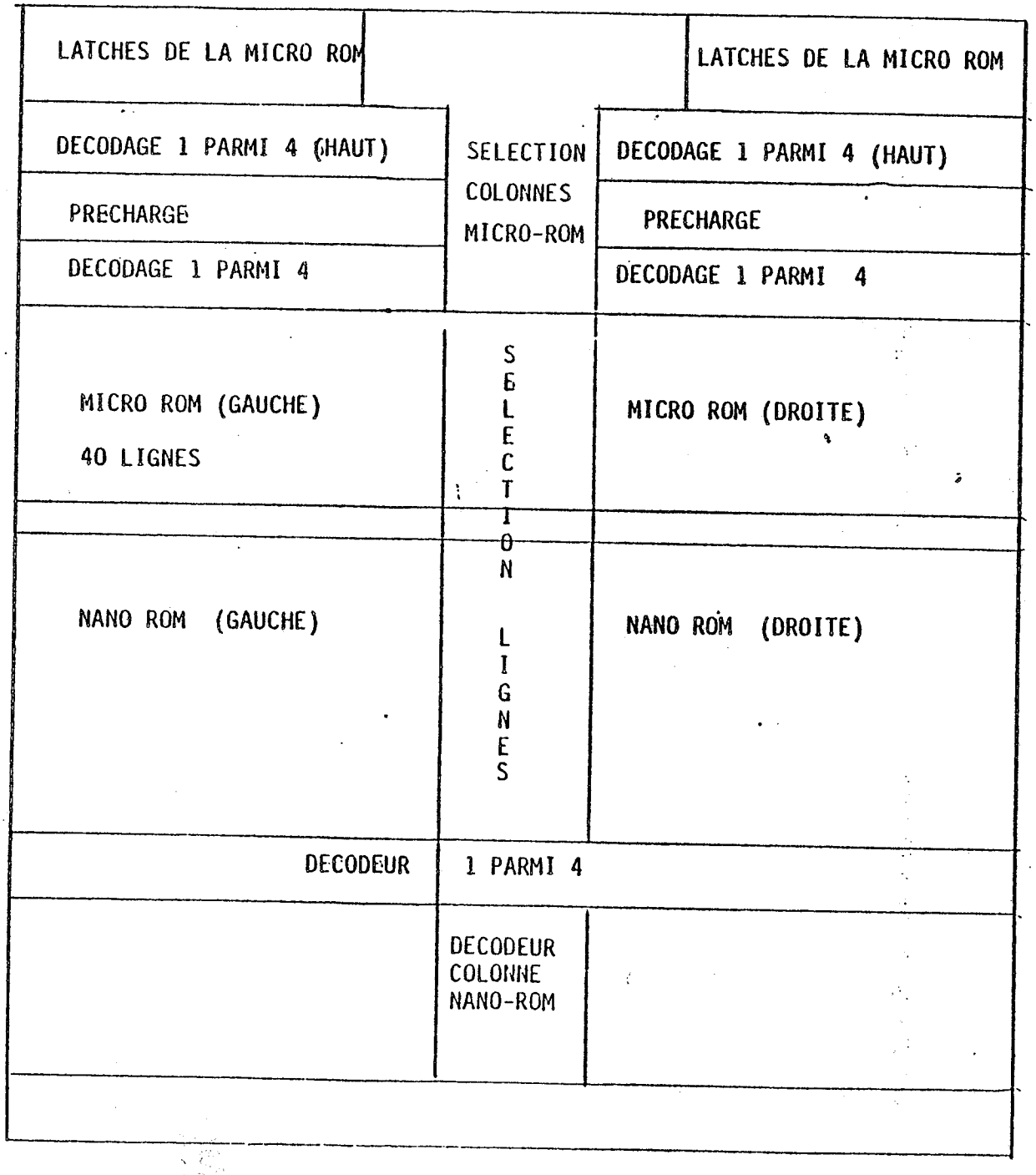


Figure 4-13 Topologie de rom

La figure 4-14 montre les décodeurs de bits d'adresses et le circuit de décodage de colonne.

Lès bits d'adresses A0,A1,A2 et A3 selectent les colonnes à partir de T1. La précharge des transisitors de commande des colonnes s'effectue sur T2. L'implantation d'un transistor sur le point mémoire séléctionné par les décodeurs, valide une commande "1" à la sortie de l'inverseur 1. Cette validation est active sur le temps T3. Donc pendant T2, on précharge seulement les colonnes séléctées.

12-Mécanisme de décodage des lignes

Les bits d'adresses A4,A5,A6,A7,A8 et A9 selectent les lignes de la rom.

Le système de décodage est basé sur les impulsions d'horloge et un système de précharge est employé.

Deux transitors ayant leurs sources communes, reçoivent une commande de sélection à partir du décodage de deux bits d'adresse A6 et A7 (T1 et T2). La validation de la source de ces transistors est synchronosée par T3. Les grilles des transistors 1 et 2 sont activées par deux lignes de commande, LC1 et LC2. Deux transistors (3 et 4) commandés par les bits A5 et A5' vont mettre à zero l'une de ces lignes LC1 et LC2 . La période de validation de A5 est de T2 à T3. Selon l'implantation des transistors de connexion entre LC1 et LC2, l'une de ces dernières sera reliée à la ligne de rom. La sélection est réalisée par les bits A2,A3,A4,A8 et A9. La période de validation de cette commande est du T1 courant jusqu'au T1 prochain. L'une des deux lignes LC1 ou LC2 qui ne sera pas reliée à l'autre prendra la valeur logique "1" sur T1. Celle-ci commandera l'un des deux transistor 1 ou 2.

La validation du décodage des bits A6,A7 sur T3 et sa coincidence avec l'activation de la grille de l'un de deux transistors 1 ou 2 validera une ligne de la rom. Cette activation commande les transistors connectée à cette ligne.

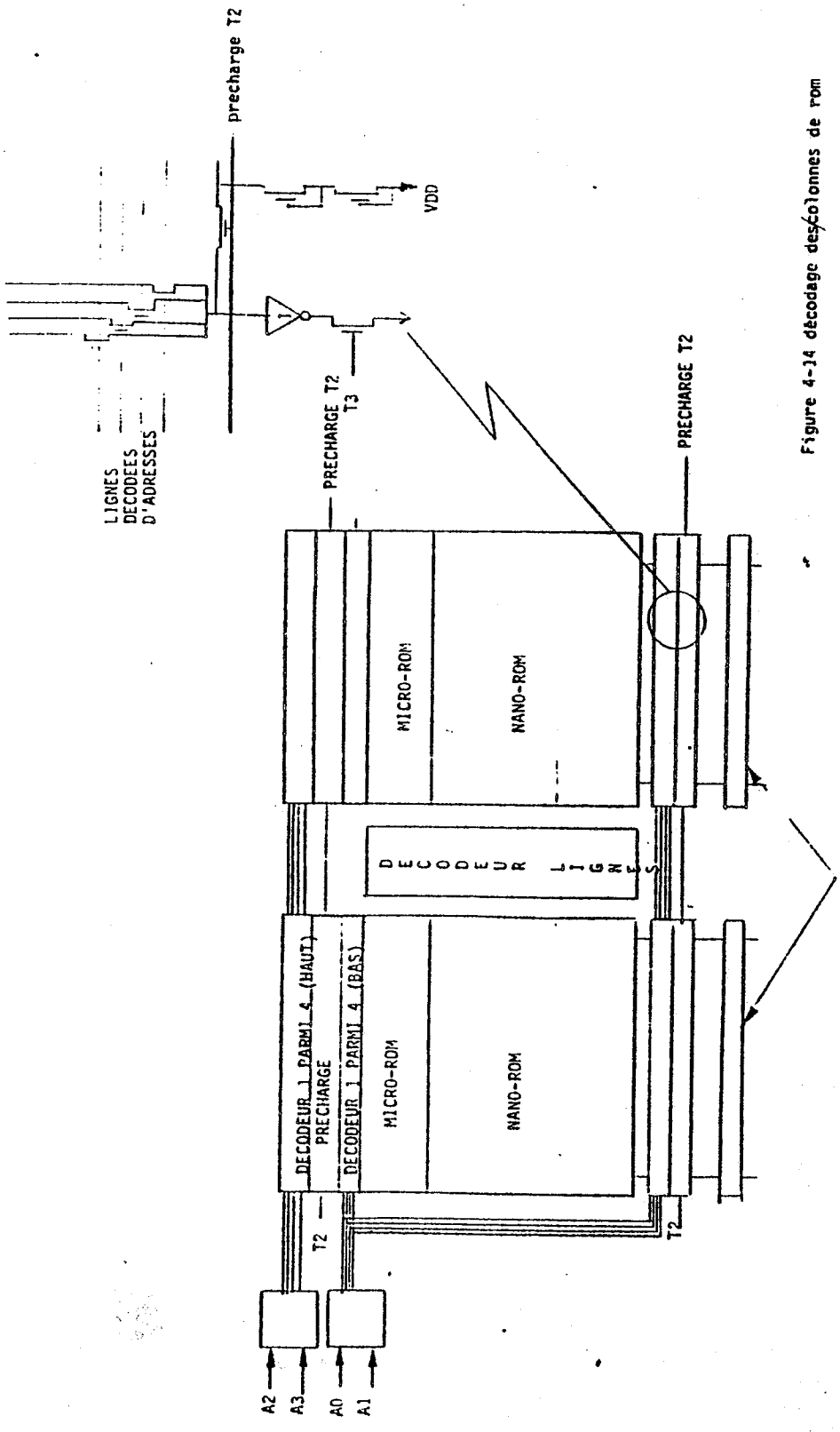
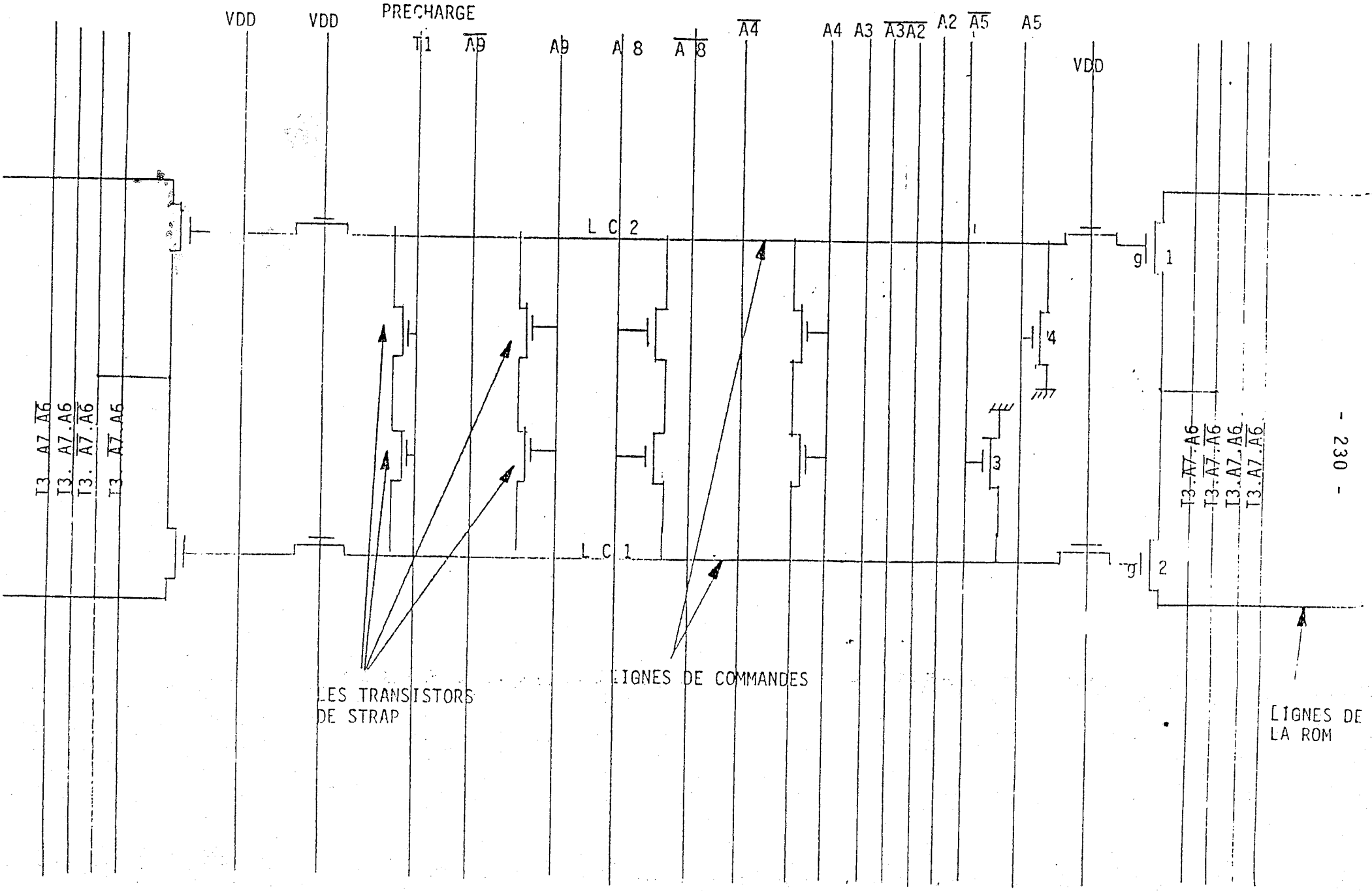


Figure 4-14 décodage des colonnes de rom


VDD
VDD
PRECHARGE
T1
A9
A9
A 8
A 8
A4
A4
A3
A3A2
A2
A5
A5
VDD
L C 2
L C 1
T3. A7 A6
T3. A7.A6
T3. A7.A6
T3. A7 A6
T3-A7-A6
T3-A7.A6
T3.A7_A6
T3.A7.A6
g
1
2
3
4
LES TRANSISTORS DE STRAP
LIGNES DE COMMANDES
LIGNES DE LA ROM

Le chronogramme suivant montre les étapes d'activation des éléments de ce décodeur, selon les impulsions T1 --> T4.

```
      I Les adresses A0 -> A9 sont memorisées
T1    I Les colonnes sont séléctées
      I Les lignes LC1 ou LC2 sont séléctées et préchargées

T2    I Les colonnes séléctées sont préchargées
      I Les adresses A6 et A7  sont latchées

      I Activation des lignes decodées
T3    I Validation de l'instruction de micro-programme en
      I sortie des décodeurs de colonnes


T4    I Décharge de lignes de la rom
```

Tableau de synchronisation de décodages par T1--->T4

Annexe V

TRAITEMENT DES EXCEPTIONS DU MC 68000

CAS PARTICULIERES

1- TRAITEMENT DU SIGNAL RESET

1-1 Introduction

Un point très important dans la conception des micro-processeurs est le comportement du processeur en cas de fonctionnements particuliers. Ces cas de fonctionnements sont par exemple les interruptions, où lorsque le processeur confie la gestion de mémoire à un périphérique. Les séquences d'initialisation ou la prise en compte d'erreurs de bus ou d'adresses sont d'autres exemples. Durant ces traitements d'exceptions, le processeur doit toujours rester maître du système. Ces types d'opérations permettent de classer les processeurs du point de vue de leur capacité à s'intégrer dans les systèmes complexes.

Ce chapitre presente les specifications particulières du MC68000 en cas d'exceptions. L'analyse du comportement de ce processeur en cas des d'exceptions: reset, interruptions, reconnaissance de codes operations invalides est approfondie dans cette partie.

1-2-L'analyse du signal reset

La broche bidirectionnelle RESET' (prime indique un signal actif à l'état bas), est utilisée pour initialiser le processeur, en cas d'activation de la broche RESET' par l'extérieur. La même broche peut aussi être utilisée par le processeur pour initialiser les périphériques en exécutant l'instruction RESET.

L'étude des séquences d'initialisation, est divisée en deux parties. La première partie considère l'initialisation du processeur lui-même (RESET hard), la deuxième, prend en compte les conséquences de l'instruction RESET (RESET soft) .

1-3-Specification d'initialisation de MC68000

D'aprés les spécifications prevues par le constructeur: L'initialisation du microprocesseur par le monde extérieur, nécessite l'activation simultanée des signaux RESET' et HALT', pour assurer l'initialisation correcte du processeur.

L'activation de RESET' et HALT', correspond à l'initialisation complète du systéme. Les opérations suivantes s'effectuent dans le processeur:

1- Lecture du vecteur de reset dans le table de vecteurs (Vecteur d'adresse $000000), et chargement de celui-ci dans le pointeur de pile superviseur(SSP).

2- Lecture et chargement du compteur ordinal avec le contenu de l'adresse $000004.

3- Initialisation de registre d'état avec un niveau d'interruption sept (niveau le plus élevé). Les contenus des autres registres ne sont pas affectés par la séquence de reset.

Comme l'exécution de certaines opérations nécessite des mesures de sécurité, le processeur êut être mis en mode supérviseur par activation du bit S (mise à 1) du registre d'état. C'est le cas des exceptions RESET, erreurs de bus et d'adresse, les interruptions,....etc.

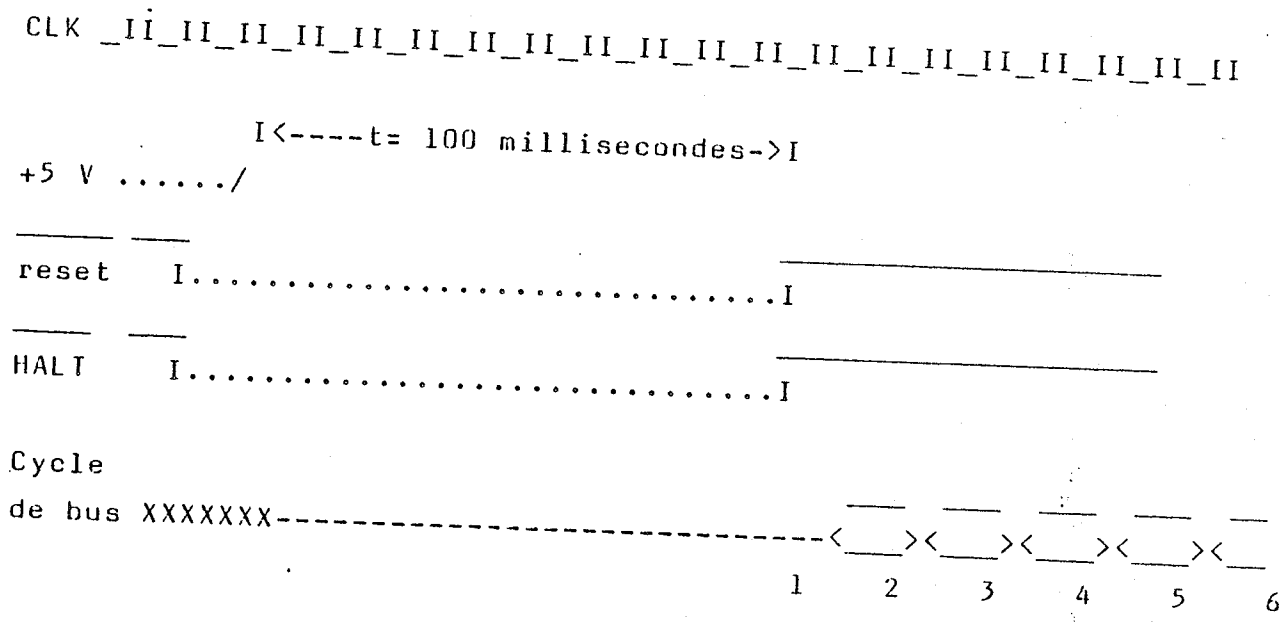
Le tableau suivant indique la priorité des exceptions provoquant un état previlégié ou superviseur du processeur.

TABLE-1 REGROUPEMENT DES EXCEPTIONS ET LEURS PRIORITES

GROUPE	Exception	Début d'éxecution
Zero	Reset Erreur de Bus Erreur d'Addr.	Début de l'éxécution après le premier cycle mineur.
Un	Trace Interruptions codes op.'s: invalides et privilégiées	Début de l'éxécution avant l'instruction suivante.
Deux	TRAP, TRAPV CHK, Division Zero	Début de l'éxécution durant l'éxecution normale de l'instruc.

Dans le groupe zero, nous trouvons l'exception de Reset avec la priorité la plus élévée. Nous verrons ultérieurement la réalisation de la séquence de reset basée sur la priorité du reset dans le groupe "0", puis la priorité du groupe zero par rapport aux autres groupes.

Dans la conception du MC68000, le Reset est considéré comme la récuperation du systéme aprés une opération catastrophique. Ces opérations correspondent à la répétition d'erreurs de bus et d'adresses ,....etc; qui mettent le processeur en état d'arrêt "HALT". En cas d'initialisation, toutes les opérations en cours, sont perdues. Le processeur se trouvant en état superviseur, met les masques d'interruption au niveau sept, pour bloquer la prise en compte d'autres interruptions, et éxécute le programme d'initialisation. En cas de mise sous tension le diagramme suivant doit être respecté.



1- temps de démarrage
2- Lecture de SSP H.
3- Lecture de SSP L.
4- lecture de PC H.
5- Lecture de PC L.
6- L'exécution de première instruction.

XXXXXXX: état de bus indeterminé.
>-----< bus de données mode de lecture

Figure-5-1 Diagramme du temp d'opération de RESET

****

Les étapes d'analyse du reset dans ce rapport, détaillent le chemin du signal reset à partir de la broche reset, jusqu'à la prise en compte et éxécution de sa séquence.

1-4-Présentation de reseau de reset

Le schéma de figure 5-2, montre le circuit du signal reset indiquant le chemin de propagation de ce signal. Le fonctionnement des blocs d'horloge et de la rom de contrôle est décrit dans le chapitre précédent. Les autres organes non-etudiés, en liaison directe avec reset, seront presenté par la suite de ce texte.

directe avec reset, seront presenté par la suite de ce texte.

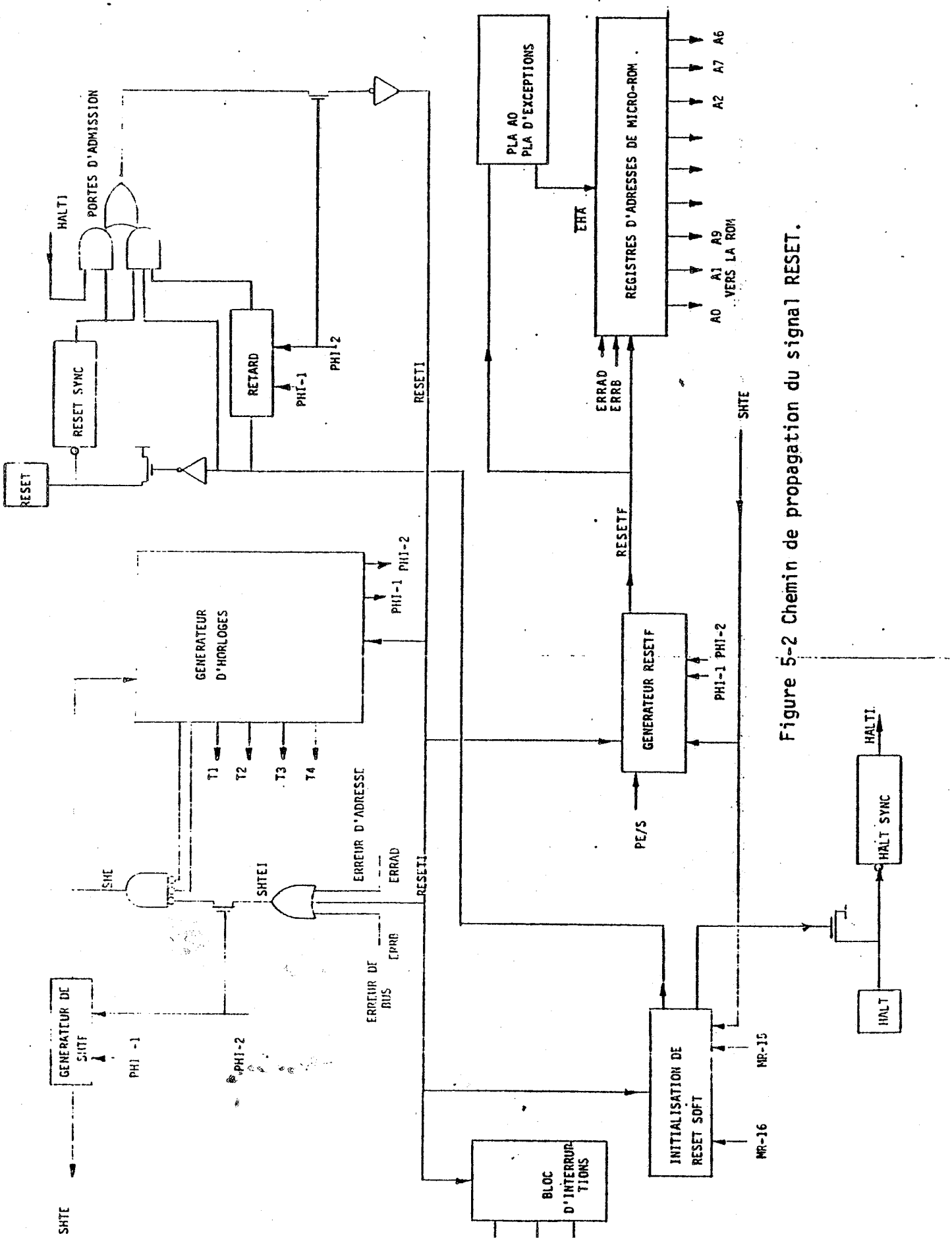


Figure 5-2 Chemin de propagation du signal RESET.

1-5-Les organes influencés par signal RESET interne

La génération du signal RESETI, par portes d'admission modifie la génération des horloges internes.
Les sorties 1,15 et 16 de la micro-rom sont mises à zero, ce qui bloque la génération du RESET soft (Instruction RESET).
Les entrées des bits IPL0->2 sont bloquées par forçage à 7 du niveau des interruptions.
Le signal RESETI, est mémorisé par une bascule, générant un signal secondaire RESETF. Celui-ci est dirigé vers le registre de micro-adresses.
Les sorties de la nano-rom sont mises à zero par le signal reset interne, provoquant un état inactif de la partie opérative.
Ce signal modifie l'adresse du micro-programme en cours. Cette modification permet le forçage direct de l'adresse de rom à zero, cette adresse correspond le début de la séquence d'initialisation.

Le PLA d'exception est mis à zéro par RESETF pour masquer les autres exceptions.

L'arrêt du fonctionnement des générateurs d'horloges par reset est utilisé pour imposer la priorité d'exécution absolue de la séquence d'initialisation sur les traitements en cours. Les impulsions de l'horlogerie d'excéptions, sont générés par l'arrêt des horloges du système. Ces horloges génèrent des signaux SHTE et T4'. Ces dérniers, chargés du traitement d'exceptions, permettent la propagation du signal RESETF vers la micro-rom.

L'architecture du reseaux d'initialisation du signal reset (groupe "0") est presenté dans la première partie de ce chapitre. L'effet des erreurs de bus ou d'adresses sur le fonctionnement du processeur est presenté. L'étude des exceptions sans une description détaillée du fonctionnement de la partie contrôle est difficile. D'autre part l'analyse de la partie contrôle nécessite une étude vaste.

Le chemin du reset est suivi dans ce texte. Il impose l'analyse préalable des blocs affectés par ce signal (ceux-ci ont partiellement presentés dans l'annexe IV).

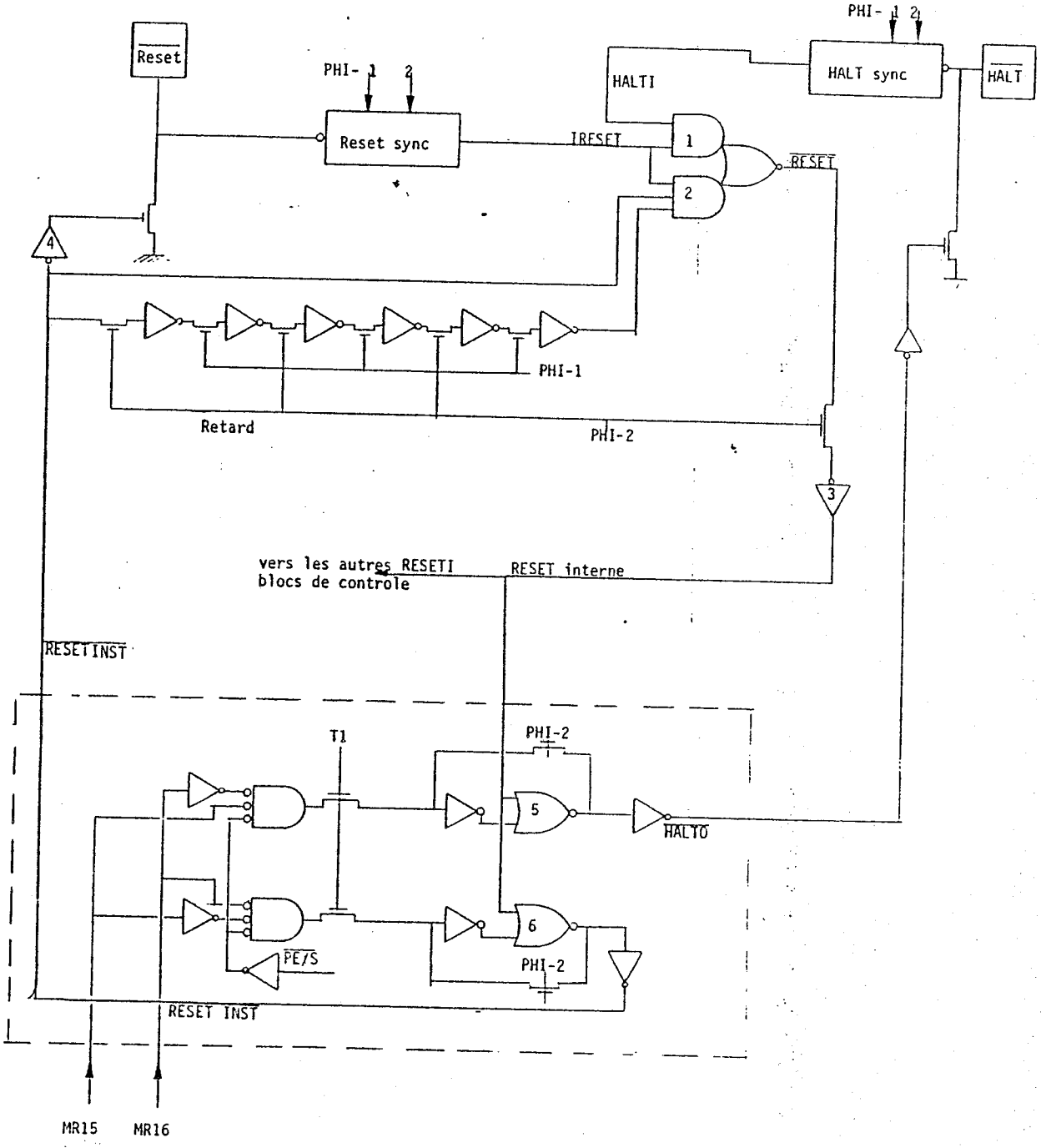
Le deuxième groupe d'exceptions comprenant les interruptions

et le traitement des codes invalides, sera traité dans les deuxièmes et troisièmes parties de ce chapitre.

I-6-Le bloc d'entrée-sortie du reset

Le bloc de génération du signal RESETI, est détaillé dans cette partie. La figure 5-3, présente ce circuit avec les signaux communiqués entre le processeur et les plots: reset et halt. Les conditions de prise en compte d'initialisation sont expliquées. Le signal RESETI sert à modifier le déroulement du processeur et à imposer la priorité du reset par blocage des horloges secondaires.

Le signal RESETI, sera transmis aux autres blocs de contrôle du MC68000. Ces blocs comme le générateur d'horloge interne, les circuits d'admission des interruptions, la ROM de micro-programme,etc..... sont affectés.



Génération de RESETI

Figure 5-3 Génération de RESETI

****

1-7-Les signaux d'entrée et sortie:

Signal RESET' , bidirectionnel chargé de la transmission de la commande d'initialisation, depuis ou vers le monde exterièur. Ce signal est appliqué au plot d'entrée-sortie de reset. L'état actif est l'état bas qui sera noté reset'.

Signal RESET INST , provenant de reconnaissance de l'instruction RESET.

Signal d'entrée HALTI , activé par plot HALT'.

Signal de sortie de ce bloc RESETI pour activer les séquences d'initialisation interne du microprocesseur.

Signal HALTO , pour activer le plot HALT par le processeur._

Les éléments constituants ce bloc:

1-8-Le circuit de reset sync.

La figure 5-4 presente le circuit de Reset sync, La validation du signal reset, active les éléments de mémorisation de ce registre à décalages. Il est constitué de trois éléments de mémorisation (les inverseurs 3,4,5,6,7,8), validant le signal IRESET pour trois periodes de PHI, aprés la fin d'application du signal reset. Le même circuit de synchronisation est utilisé pour le plot HALT, générant le signal HALTI. Le figure 5-5, montre la génération de RESETI à partir de deux signal HALTI et IRESET.

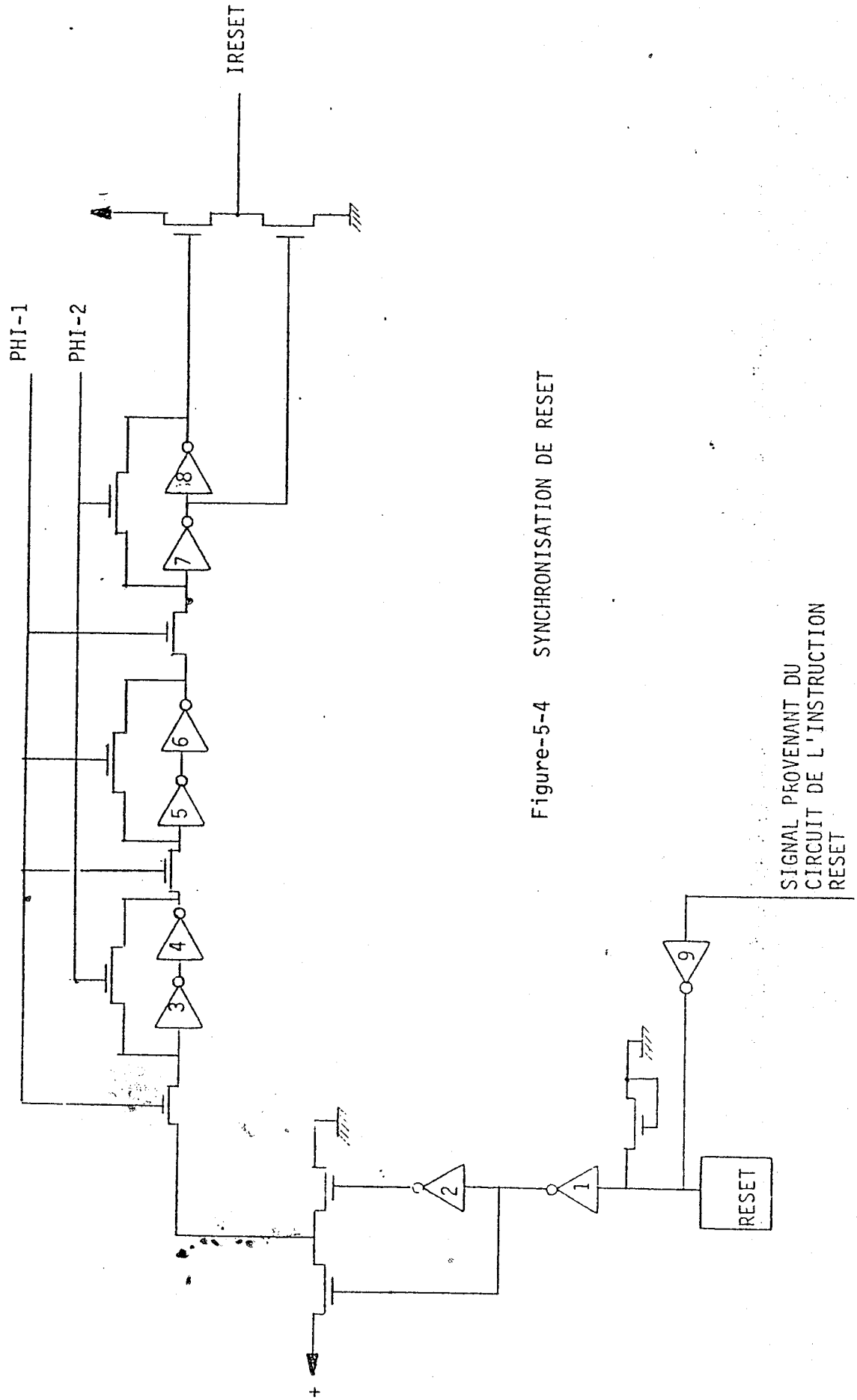


Figure-5-4 SYNCHRONISATION DE RESET

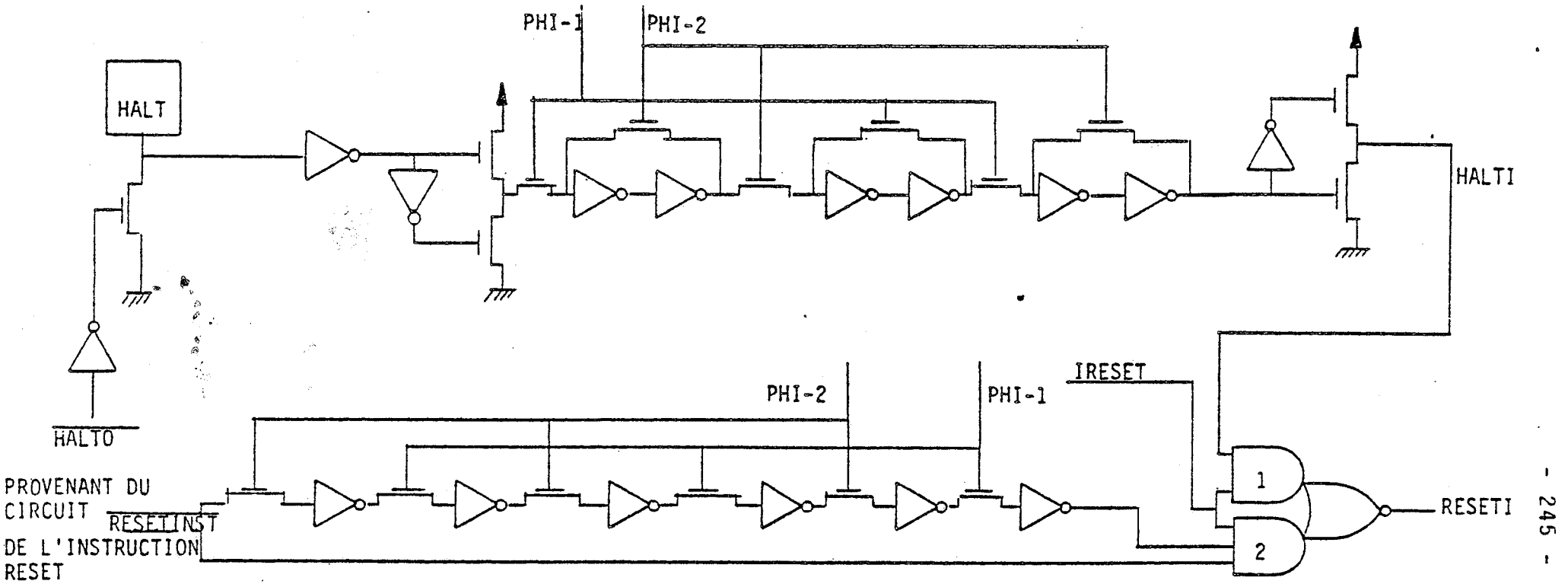


Figure 5-5 - L'ADMISSION DE RESET

Portes d'admission de reset (portes ET-1,2. Fig-5-2).

Les portes ET 1 et ET2, imposent les conditions de presence de halt et absence de l'instruction RESET pour l'admission de la commande de reset.

Retard 3T (PHI-1 et PHI-2).

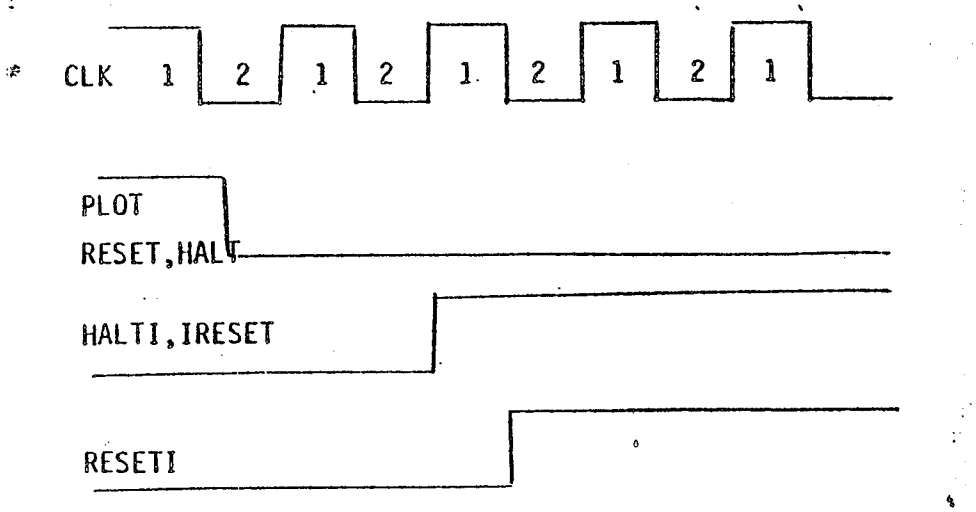
Ce retard est destiné à bloquer la porte ET-2 , durant 3 cycles d'horloge (PHI1, PHI-2) aprés la fin de l'exécution de l'instruction RESET.

1-9-Le circuit de génération de RESET INST et HALTO.

Ces deux signaux sont issus des NOR's 5 et 6 de la figure 5-2. MR15 et MR16 sont les commandes des sources provenant du registre de micro-instruction. Ces deux bits sont positionnés par l'éxécution de l'instruction RESET. Le signal PE/S', commande des opérations d'entrées-sorties du processeur. Celui-ci est issue de deux bits de la nano-instruction , les bits 4 et 5. Le fonctionnement du bloc de génération du signal RESET INST', sera analysé dans une autre partie où les séquences de l'instruction de RESET sont detaillées.

Le fonctionnement de ce bloc -

Les deux portes ET1 et ET2 sont chargées de la prise en compte du signal reset (actif à l'état haut) provenant du plot. L'élément de synchronisation "Reset sync", valide ce signal sur la phase PHI-2. La porte ET-1, génére alors le signal RESETI, à partir du signal reset en coincidence avec l'activation de Halt. Le HALTI est activé si la broche Halt est mise à zero. Le chronogramme suivant montre le timing de génération de RESETI.



5-6 Timing de reconnaissance du signal RESET'

****

La porte ET-2, valide RESETI, si le signal RESET INST' est désactivé. L'élément de retard, distingue les deux états de reset (soft et hard) durant 3 periodes d'horloge. Les sources de la génération de RESET sont: l'execution de l'instruction RESET et l'initialisation du système par mise à zero de la broche RESET'. Cette porte (ET-2) admet les conditions d'entrées, si broche reset' est activée et l'instruction RESET est terminée il y a 3 periodes d'horloge. RESET INST', provient du circuit de génération de reset soft. Ce circuit active les commandes RESET INST' et HALTO' à partir de l'instruction RESET. Les chronogrammes de figure 5-7 montrent la génération de RESETI pour ces conditions.

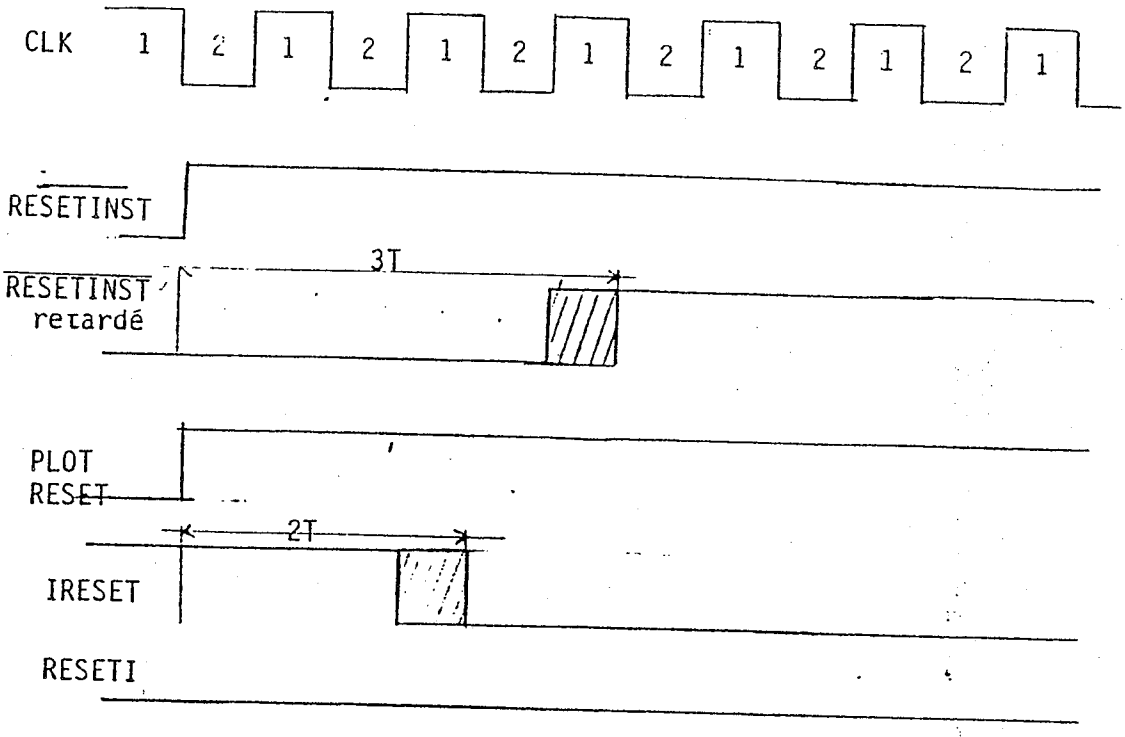


figure 5-7 L'horlogerie de l'admission de RESET'
Cas de coincidence avec l'instruction RESET

****

Un troisième cas peut être distingué si le signal halt n'est pas actif et que l'on n'éxécute pas l'instruction RESET. Dans ce cas-là, RESETI est généré par la porte ET-2 sur la phase PHI-2.

En cas d'activation de broche HALT' en coincidence avec le début de l'exécution de l'instruction RESET la porte ET-1, génére le signal RESETI. Dans ce cas là, le processeur va reconnaitre la mise à zero du plot reset' comme une commande d'initialisation appliquée à cette broche. La présence de RESETI, a cause de la priorité du reset, bloque les portes NOR 5 et 6 sur Figure 5-3. Autrement dit, en cas d'arrivée du signal HALT, le processeur met à zero la broche pour le temps de delai du bloc reset sync. La génération artificielle de RESETI et sa durée de vie dépend du temps d'attente de l'élément de reset sync pour 2 periodes d'horloge. La désactivation de RESETI, à cause de bloquage de sa source "RESETO", établit une durée de mise à zero de 2 cycles d'horloge sur le plot reset. Dans le cas où une telle periode est suffisante pour déclancher les séquences d'initialisation, le

processeur ne répondant pas à la demande de HALT (plus prioritaire que l'instruction RESET) va se réinitialiser.

1-10-Génération de RESETF

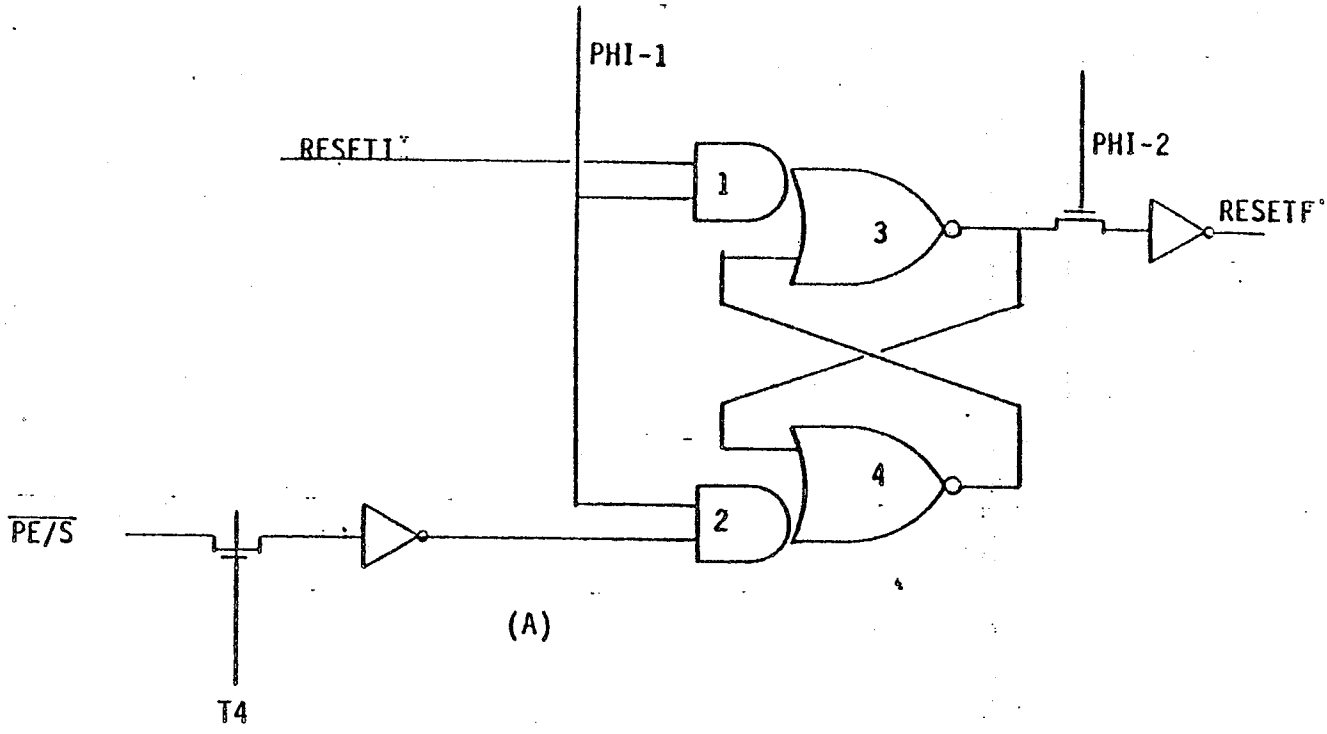
La figure 5-8-A montre le circuit de génération du signal RESETF. Ce circuit est constitué de deux portes NOR(3,4) rebouclées, constituant une bascule. Les signaux d'activation et désactivation de cette bascule, sont synchronisés par PHI-1, par deux portes ET (1,2). Le RESETI active cette bascule afin de générer le signal RESETF. Autrement dit RESETF mémorise l'état reset. Le signal PE/S, désactive le RESETF.

Le chronogramme correspondant à cette partie est donné par la figure 5-8-B.

Le signal PE/S' est issu de deux bits de la nano-rom qui contrôlent les opérations d'entrées/sorties du processeur. Les differentes configurations de BIT0 et BIT1 et les commandes correspondantes sont présentées par le tableau suivant:

bit0	bit1	command générées
0	0	aucune operation
0	1	bus adresses ---> buffer adresses
1	0	bus de donnée ---> buffer données
1	1	sortie l'UAL ---> buffer d'ALU

L'exécution de la séquence d'initialisation débute les opérations d'entrées/sorties. Dans ce cas-là, le PE/S est activé au cours de la procédure d'excéption. Cette activation remet à zéro le signal RESETF. Autrement dit la durée d'activité de RESETF correspond à l'intervalle de temps entre la présence de reset sur la broche et le premier échange de donnée avec le monde extérieur. Figure 5-8-B, montre l'activation et la désactivation de la bascule de mémorisation du reset par les signaux concernés.



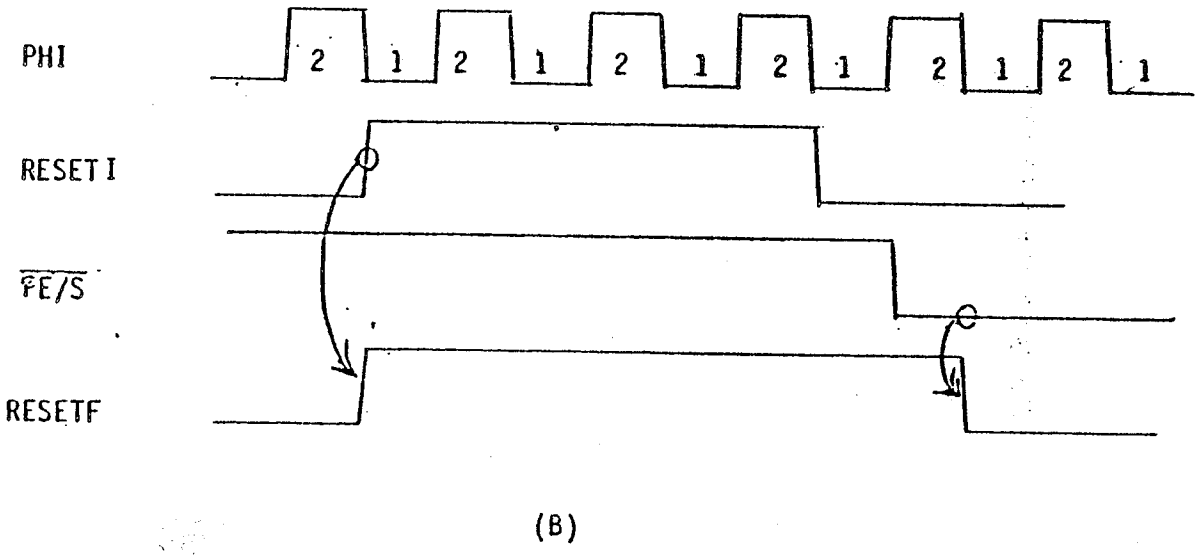


Figure 5-8 Génération de RESETF à partir de RESET

1-11-L'ADRESSAGE DES SEQUENCES D'EXCEPTIONS (GROUPE ZERO)

Le déroulement normal du traitement des instructions est affecté par les signaux d'exceptions du groupe 0, ceci est présenté par figure 5-9. Ces exceptions se manifestent par trois signaux actifs à l'état haut: RESETF, ERRB et ERRAD. Les ERRB et ERRAD, sont issus de la logique de l'erreur d'adresse et l'erreur de bus. A cause de la priorité autonome des exceptions du groupe "0", ces signaux seront chargés dans les registres R1,R2 et R3 afin de modifier l'adresse du micro-programme. La prise en compte du signal d'initialisation RESETF par R1, force directement l'adresse du micro-programme. La porte NOR 10, valide cet effet pour les exceptions de ERRB et ERRAD. La mise à zero du NOR 10, bloque les fils d'adresses par les portes NAND 1,2 et les NOR 3,4,5,6,...etc. Le signal FAD' est à zero et FAD est à 1 ce qui permet ainsi d'annuler l'adresse presente sur A0-->A10. Toutes les sorties des portes NOR 3 --> 6 sont mises à zero et celle des NAND 1 et 2 sont à 1. En cas de validation d'exceptions l'adresse forçée à la valeur "1100000000" permet ainsi le décodage de l'adresse de l'exceptions sur les deux bits poids faibles de cette adresse. Le PLAF situé sur le passage de deux bits A0 et A1 code les adresses d'exceptions.

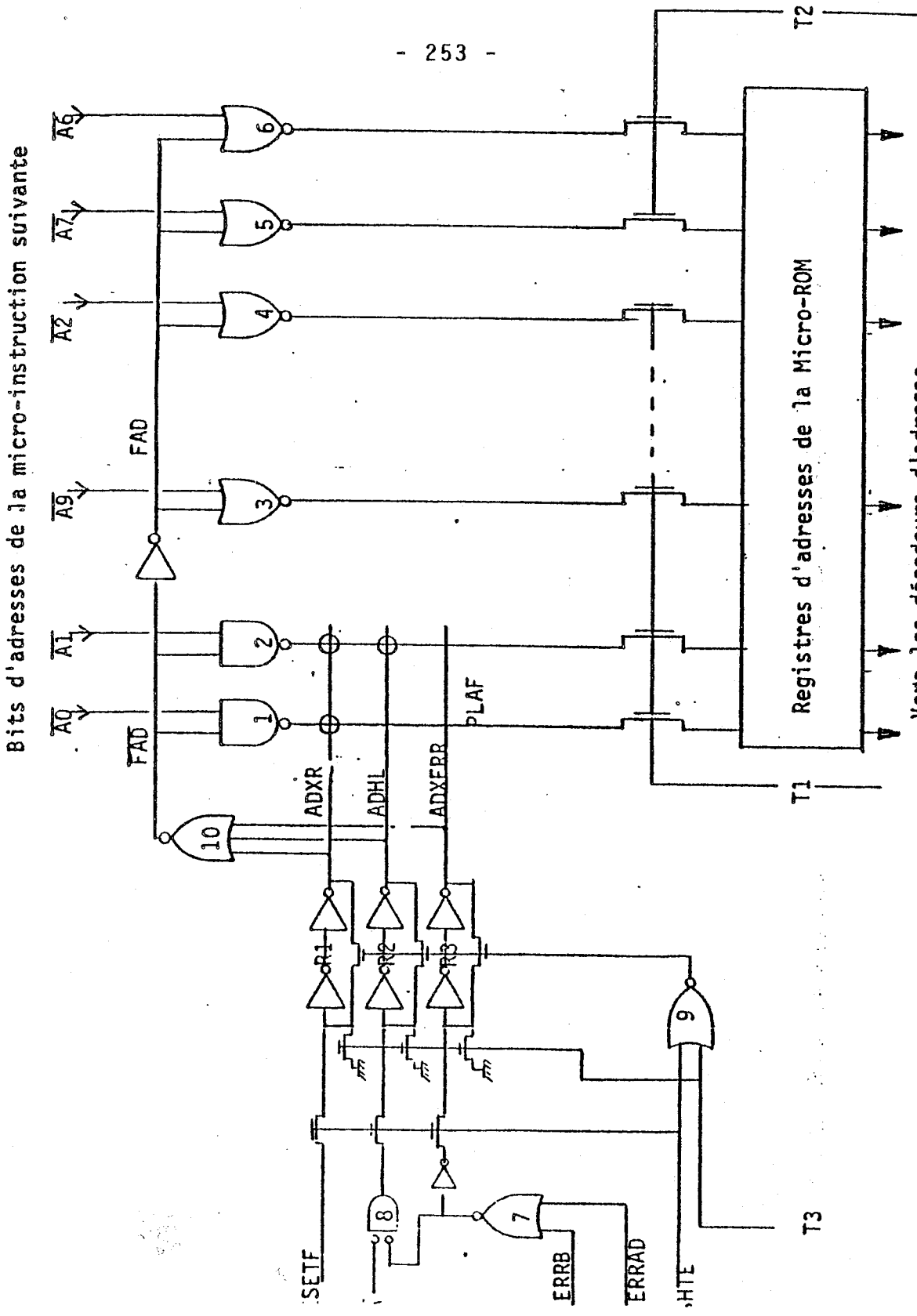


Figure 5-9 Forçage de l'adresses de l'exception dans la ROM de micro-programme

Figure 5-9 Forçage de l'adresses d'éxcéptions dans la rom de micro programmes

****

1-12-Codage des adresses d'exceptions

Le PLAF avec 3 entrées et deux sorties peut modifier les deux bits de sorties de AND 1 et 2, selon les lignes d'entrées:

La registre R1 qui mémorise le signal de RESETF choisi une adresse du micro programme 000(hexa).Cette adresses est le début du traitement d'initialisation du reset.

La deuxième entrée de ce PLA, ADHL, est choisie si le registre R2 est activé, ce qui donne l'adresse du micro-programme 001(hexa). Cette adresse provoque l'état HALT du processeur. Il est activé si deux erreurs de bus et ou d'adresses se manifestent consécutivement.

La troisième ligne donne l'adresse 003(hexa) qui traite les séquences d'erreurs du bus ou d'adresses. La priorité de la séquence de RESET par rapporrt aux autres séquences d'exceptions est respectée par l'implantation du PLAF. En cas de coïncidence d'apparition du reset avec l'erreur de bus et d'adresses, ou du halt; l'adresse du reset masque le codage des autres adresses par mise à zero des deux bits A0 et A1. La même priorité existe pour l'état HALT par rapport aux erreurs de bus ou d'adresses.

La porte NOR 7, en cas d'erreur de bus active le registre R3. En cas d'arrivées consécutive de deux erreurs, reconnues par le signal EHA', l'état halt est choisi par R2. La génération du signal EHA' est précisée dans la partie consacrée au traitement des exceptions. L'adresse de halt est forçée, en cas de coïncidence de l'erreur de bus et d'adresses.

Le fonctionnement et la synchronisation des registres R1 --> R3 est basé sur l'horlogerie d'exception. Le signal SHTE charge, ces registres. Ceux-ci sont remis à zero sur chaque impulsion de T3 (en cas de fonctionnement normal des horloges). Les contraintes de SHTE et mise à zero des registres assurent le forçage de l'adresse d'exceptions si les horloges fonctionnent en mode modifié. Le NOR 9 permet la mémorisation des registres lorsque T3 et SHTE inactifs(T3' et SHTE').

L'imposition de ces contraintes se fait en liaison avec le fonctionnement il est modifié par le reset desgénérateurs d'horloges. La

logique suivante peut être tirée du circuit de registres d'adresses d'exception:

Le forçage eventuel des registres n'est affecté qu'aprés la génération de SHTE.

On charge donc les adresses de traitement d'exceptions durant la période d'arrêt de l'horloge. Ce forçage est éfféctué pendant toute la durée d'activation du signal reset. Le redémarrage des générateurs d'horloges de base (T1->T4), recommence le séquencement normal de la rom. L'adresse de traitement de l'exception forçée, sera prise comme l'adresse de la routine à éxécuter.

Les chronogrammes de la figure 5-10 montrent les quatre étapes successives. La première présente le fonctionnement normal. La deuxième correspond à la période d'activation du signal reset. La troisième explique l'intervalle de chargement de la première micro- instruction de traitement d'initialisation en fin la quatrième correspond à l'éxecution de la première instruction micro-programmée.

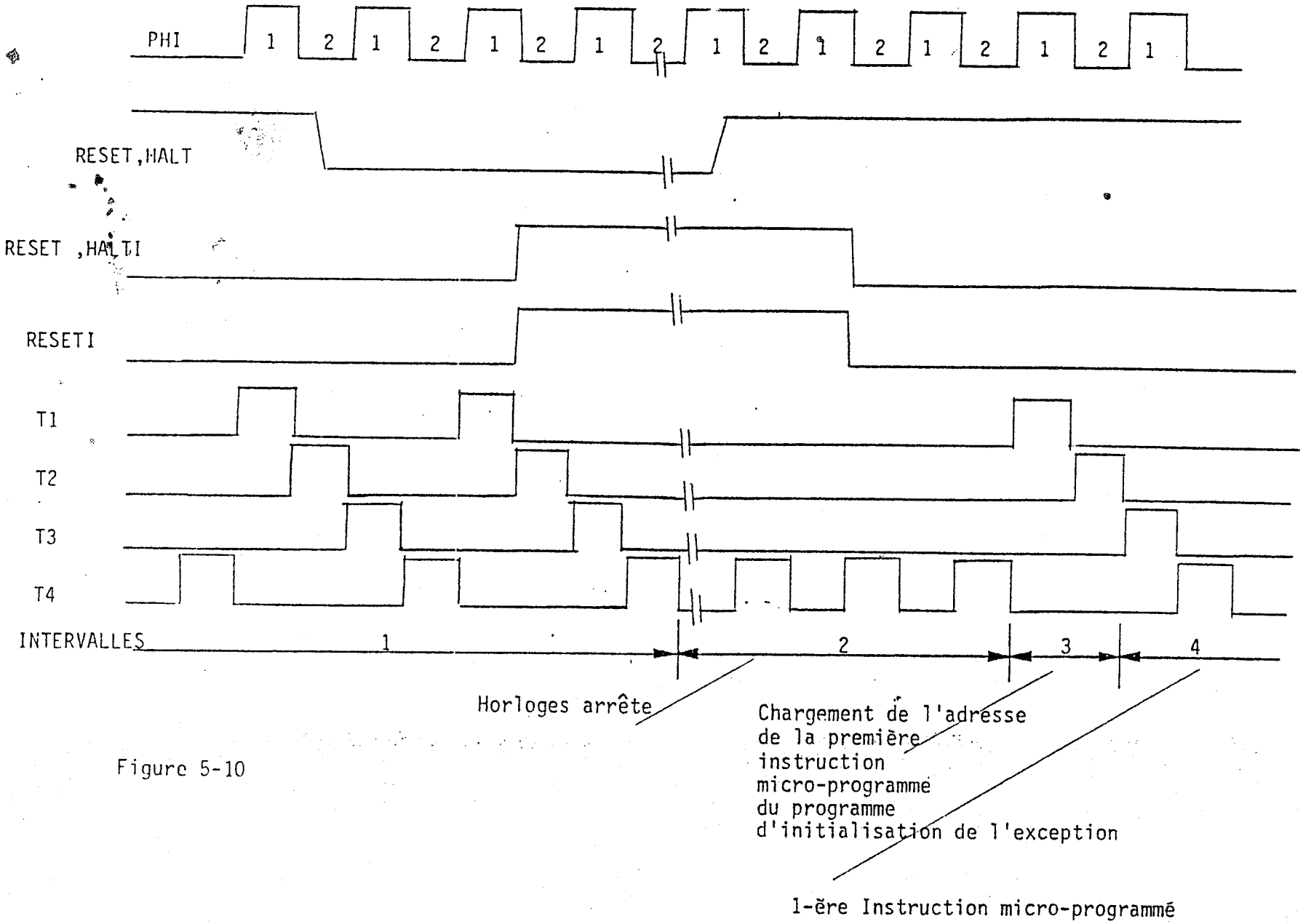

PHI
1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
RESET,HALT
RESET ,HALTI
RESETI
T1
T2
T3
T4
INTERVALLES
1
2
3
4
Horloges arrête
Chargement de l'adresse de la première instruction micro-programme du programme d'initialisation de l'exception
1-ère Instruction micro-programmé

Figure 5-10

1-13-La séquence de traitement d'initialisation

L'adresse du micro-programme d'initialisation est forçée par la broche ' reset'. La routine de traitement du signal reset est démarrée par la suite. Cette séquence est composée de 12 états, se trouvant aux adresses de micro-programmes suivantes:

	ADRESSE PRESENT	NOM ETAT	MICRO INSTRUCTION 15,16	adresse future	C X Y 0	B I
*	0000000000	bnra2	0 0	1010010110	1 0 0	0 0
	1010010110	oplw2	0 0	1010111101	1 0 0	0 0
	1001111110	ffte1	0 0	1011111101	0 0 0	0 0
	1001111111	ffte2	0 0	1011111011	0 0 0	0 0
	1011111011	xerc2	1 0	1011111111	0 0 0	0 0
	1011111111	xerc6	1 0	1100000000	0 0 0	0 0
	1100000000	xerc7	1 0	0110011100	0 0 0	0 0
	0100011110	htyyu	1 0	1011110110	0 0 0	0 0
	1010110111	vvrw1	1 0	0100011100	0 0 0	0 0
	0100011100	cfrty	0 0	1111100001	1 0 0	0 0
	1101100011	rtta2	1 0	1100001101	0 0 0	0 1
	1101001100	vccx3	0 0	0000000000	0 0 1	0 0

Tableau des micro-instructions executées par reset

Dans ce tableau l'adresse de chaque micro-instruction et le nom de l'état correspondant sont présentés. Les 17 bits de micro-instruction sont divisés en différents champs. On distingue les deux premiers bits (bits 15 et 16) de la micro-instruction, puis un un groupe de 10 bits (l'adresse suivante). Enfin les 5 dernières bits, commandent les différents blocs de la partie contrôle. Le champ adresse de chaque micro-istruction ne contient pas les bits d'adresse, dans un ordre correct pour l'adressage. Ces bits sont rangés dans l'ordre suivant:

Bits d'adresse	A9		A8		A1		A0		A5		A4		A3		A2		A7		A	
Bits micro instruction	I	14	I	13	I	12	I	11	I	10	I	9	I	8	I	7	I	6	I	5

Cette cofiguration des bits, facilite l'accès des décodeurs de lignes et colonnes de rom.

Pour arriver à tirer "l'adresse suivante", dans l'ordre corect il faut intervertir les deux premières bits A6 et A7 (bits 5 et 6 de la micro-instruction par A1 et A0, se trouvant dans les bits 11 et 12 de la micro-instruction. Les autres bits ne sont pas changés.

Le detail des opérations effectuées dans la partie contrôle et la partie opérative est classé par états. La séquence suivante est parcourue pour initialiser le processeur:

bnra2

I

oplw2

I

ffte1

I

ffte2

I

xerc2

I

xerc6

I

xerc7

I

htyyu

I

vvrw1

I

cfrty

I

rtta2

I

vccx3

I

Sélection du PLA A1

Séquence d'initialisation du reset

1-14-1-L'instruction RESET

Nous avons déjà constaté, que la séquence de reset interne à une priorité autonome. Le cas particulier de l'éxécution de l'instruction "RESET", est bloqué par signal SHTE. Le circuit de figure 5-11-A, montre les générateurs des sigaux HALTO et RESET INST à partir dir deux bits de la micro-instruction. Les bits 15 et 16 commandent la bascule de RESET INST'. En cas d'éxécution de l'instruction RESET, les deux bits 15 et 16, sont positionés pour activer RESET INST'. Les différentes configurations de ces deux bits sont données dans le tableau suivant.

Bit 16	Bit 15	Mode de génération
0	0	aucun effet
1	0	initialisation de HALTO'
0	1	initialisation de RESET'
1	1	aucun effet

La génération de ces deux signaux est bloquée par la remise à zéro des deux bits 15 et 16 du registre de micro-instructions par SHTE. Cet effet élimine l'instruction reset, en cas de coincidence avec des exceptions du groupe 0.

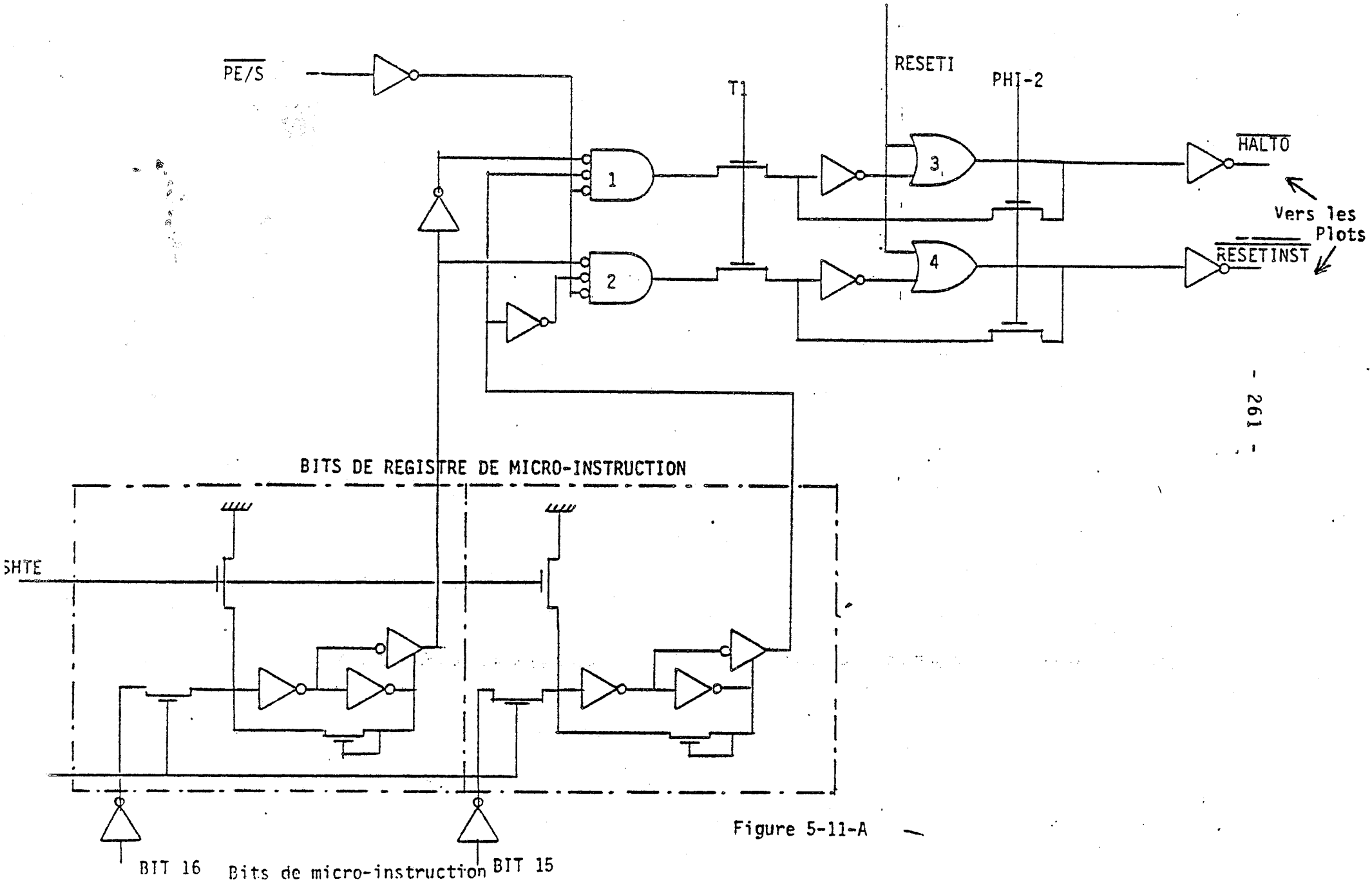

PE/S
T1
RESETI
PHI-2
1
2
3
4
HALTO
Vers les Plots
RESETINST
BITS DE REGISTRE DE MICRO-INSTRUCTION
SHTE
BIT 16
Bits de micro-instruction
BIT 15

Figure 5-11-A

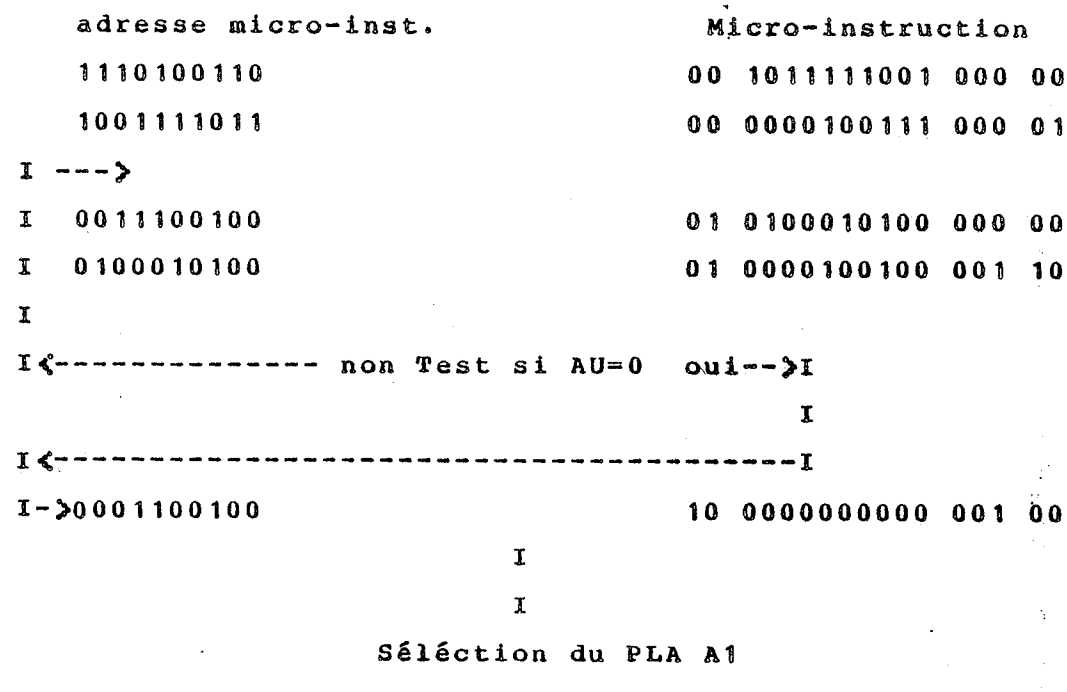
Deux portes AND 1,2 et le décodeur d'entrée, initialisent les sorties finales RESET INST' et HALTO'. Le signal PE/S' est employé pour bloquer les portes ET 1,2. Le signal PE/S' provient de deux bits de la nano rom (bits 0 et 1) qui commandent les buffers d'entrées/sorties. A l'état de repos, PE/S'=1 indique qu'aucune entrées/sorties n'est commandée par le processeur. Lors de l'initialisation d'échanges de données PE/S' met à zéro les sorties des ET 1 et 2. Ces deux sorties, synchronisées sur T1, commandent les éléments de génération de RESET INST' et HALTO'. Les deux NOR 3,4 mémorisent ces deux signaux sur PHI-2.

Le signal RESETI, remet à zéro les portes NOR 3 et 4. La priorité du reset interne par rapport au reset soft, est assurée par ce circuit. Les sorties de ce bloc à travers un inverseur tirent les plots HALT et RESET à la masse.

En cas de reconaissance de l'instruction de RESET, la configuration des bits 15 et 16 de la micro-instruction est gardée à la valeur (1,0) durant 124 périodes d'horloge, ce qui met à zero le plot reset durant tout cet inervalle. La partie opérative est chargée de compter les 124 impulsions d'horloge. Dans ce cas-là, L'AU en mode décrementation est chargée de la valeur 32. Ceci est réalisé par décomptage de -32 à 0 , le signal de compte de fin d'opération ne se faisant que sur 5 bits du résultat de l'AU (bit 0 à bit4). Le micro-programme de l'instruction RESET, pour 32 cycles machines, décremente et test l'arrivée de cette valeur,à zéro. Cette période est équivalente à 124 pas d'horloge, cèci assure le mise à zéro des éléments du monde exterieur par le processeur.

1-14-2-La séquence de l'instruction RESET

Cette séquence est constituée de 5 étapes et d'un test sur le résultat de AU. L'organigramme de l'exécution de l'instruction RESET est presenté par la suite:



Séléction du PLA A1

Les chronogrammes des signaux activés par l'éxecution de l'instruction RESET sont presentés par la figure 5-11-B. La désactivation du signal RESETI à la fin de mise à zero du plot reset, est assurée par l'élément de retard. Celui-ci prolonge le signal RESET-INST', pour 6 périodes de T. Par conséquent le signal RESETI reste désactivé, pendant toute la durée d'initialisation par l'instruction RESET.

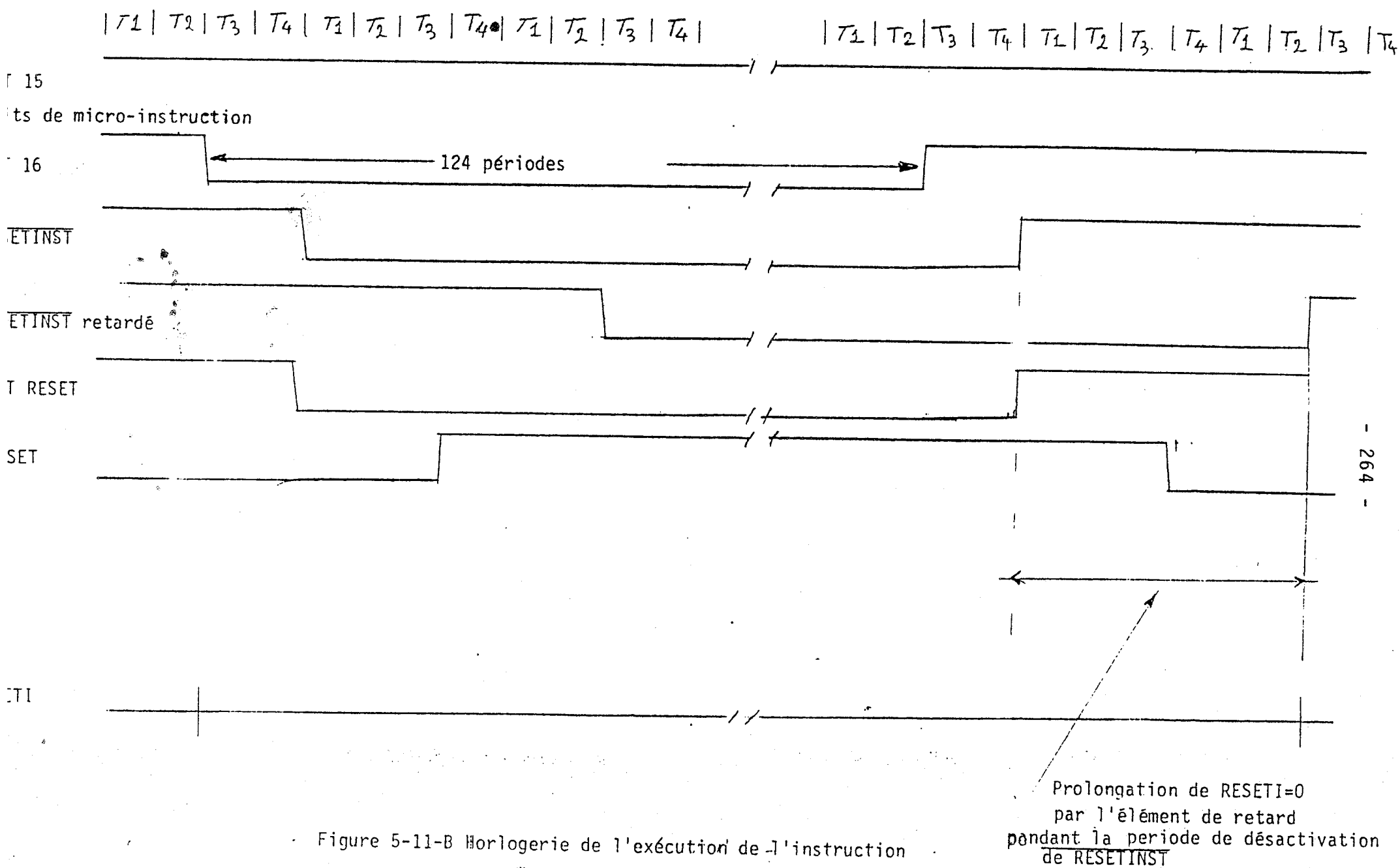


Figure 5-11-B Horlogerie de l'exécution de l'instruction "RESET "

1-15- L'état Halt du processeur

Le PLA d'exception PLA-A0, génère le signal EHA' destiné au circuit de forçage d'adresse du micro-programme d'exception (groupe 0). Dans le cas où une erreur de bus ou d'adresse se produit pendant le traitement des exceptions du groupe zéro, le signal EHA' force le processeur à l'état Halt. L'adresse du micro-programme correspondant à cet état est 001. Le contenu de la micro-instruction est le suivant:

adresse (001)---> 00 0100000000 000 00

I I

I<----------I

Cette micro-instruction reboucle sur elle-même. En effet l'adresse de la micro-instruction suivante fournie par cette micro-instruction est son adresse propre.

Ceci bloque le processeur en état Halt. La seule possibilité de sortir de cet état, est l'application d'un reset externe pour réinitialiser le processeur.

1-16-La logique d'exception

Le signal RESETF est appliqué au PLA-A0, ceci pernet d'effectuer la comparaison entre les signaux d'exceptions. L'adresse du micro-programme d'exception le plus prioritaire est générée pour les groupes d'exceptions sauf pour le groupe "0". Le niveau du vecteur d'exception correspondant est aussi généré par ce PLA. Nous allons étudier ce PLA et son fonctionnement en cas d'exception.

Le déroulement de la séquence de traitement d'initialisation appelle le PLA A0. Cette requète est commandée par le bit C de la micro-instruction.

Plusieurs commandes entrent dans ce PLA, chacune correspondant à une connaissance d'état d'exception. Le détail de ces commandes est donné par la suite. Le PLA-A0 est divisé en trois parties, chacune de ces parties ayant un rôle particulier.

La figure 5-12, représente ce découpage; les signaux d'exceptions sont entrés dans le registre d'éxception et le chargement des exceptions est commandés par deux signaux CXG0 et CXG1.

Le signal EHA' est issu de cette partie par la porte NOR-1. Ce signal permet le forçage à l'état HALT. Les entrées sont les signaux ERRAD, ERRB et RESETF.

Deux groupes de signaux sortent des deuxième et troisième parties de ce PLA.

4 fils véhiculent le codage du numéro de vecteurs d'exceptions. Ils sont générés par la deuxième partie. Ces fils génèrent à travers les deux PLA IRD et TRAP, le vecteur d'exception correspondant.

La troisième partie génére les 10 bits d'adresse du micro-programme d'exception. Ce bus est connecté par un double réseau de 10 interrupteurs aux autres bus d'adresses du micro-programme . Le signal correspondant au choix de la source de l'adresse du micro-programme, parmi les séquences normales et les exceptions, est généré par le PLA-A0. C'est un signal de sortie qui commande les deux portes AND 2 et 3 de figure 5-12.

1-17-Chargement des signaux d'exception

Deux groupes de signaux d'exception sont chargés dans le registre; Les premiers sont:

RESETF

erreur de bus ERRB

erreur d'adresse ERRAD

INTPAR'

AUTVC'

Ces signaux sont échantillonés par le signal CXG0.

Un deuxième groupe correspond aux signaux:

DINP'

PREVIL

ILLEG'

CD1010, CD1111

TCTS

Ces signaux sont pris en compte par la commande CXG1, issue du bit I de micro-instruction. Parmi les signaux d'exceptions, DINP' provient de la logique d'interruptions qui sera détaillée dans une autre partie.

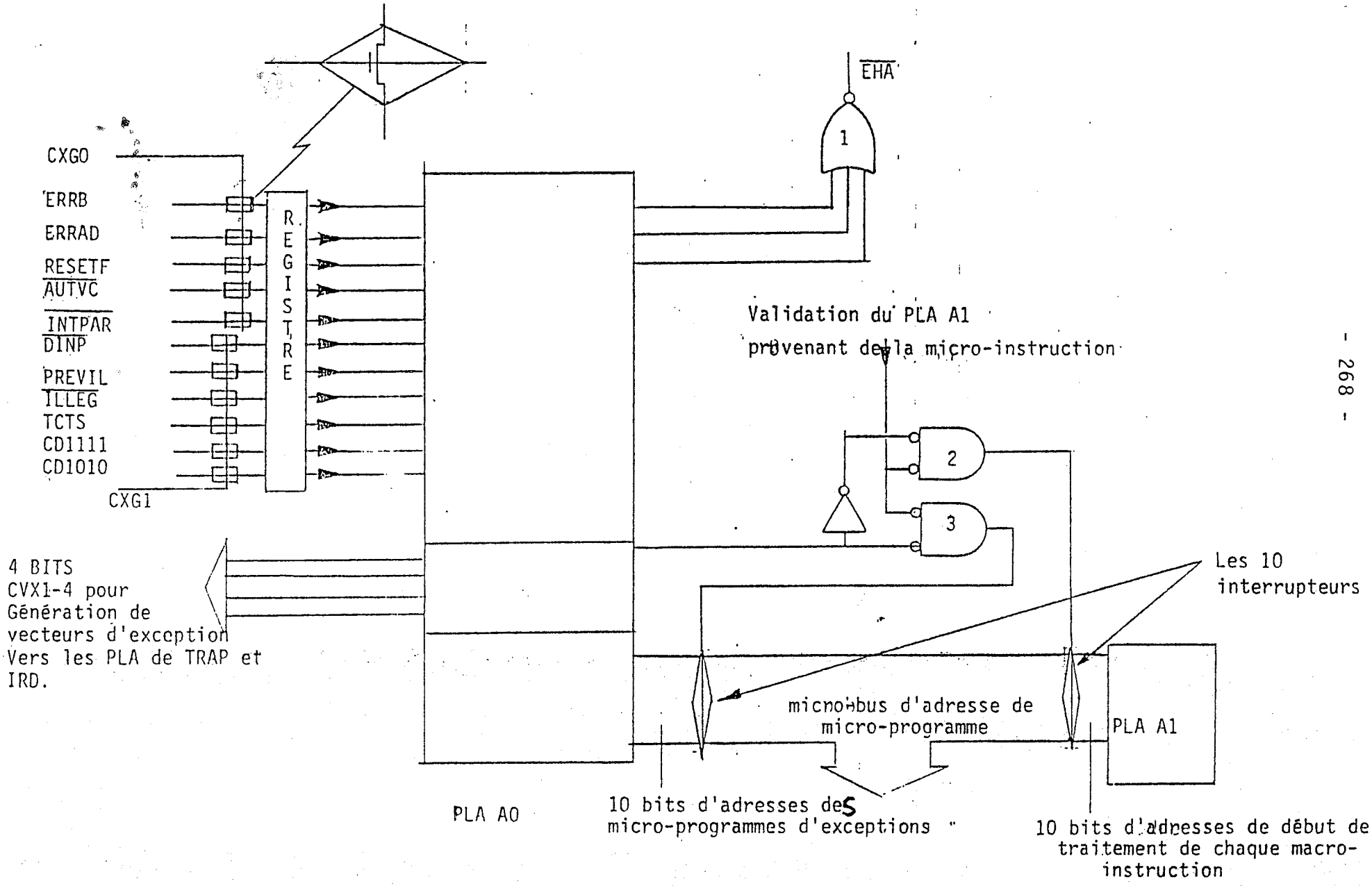


Figure 5.18 La logique d'exceptions

Comme on l'a vu précedam ent ce PLA est constitué de trois parties (sous-PLA). La première partie correspond à l'évaluation de la priorité, la deuxième partie à la génération du numéro de vecteur d'exception et la troisième partie à la génération de l'adresse du micro-programme d'exception. L'implantation des transistors du PLA A0 est précisée dans la figure 5-13.

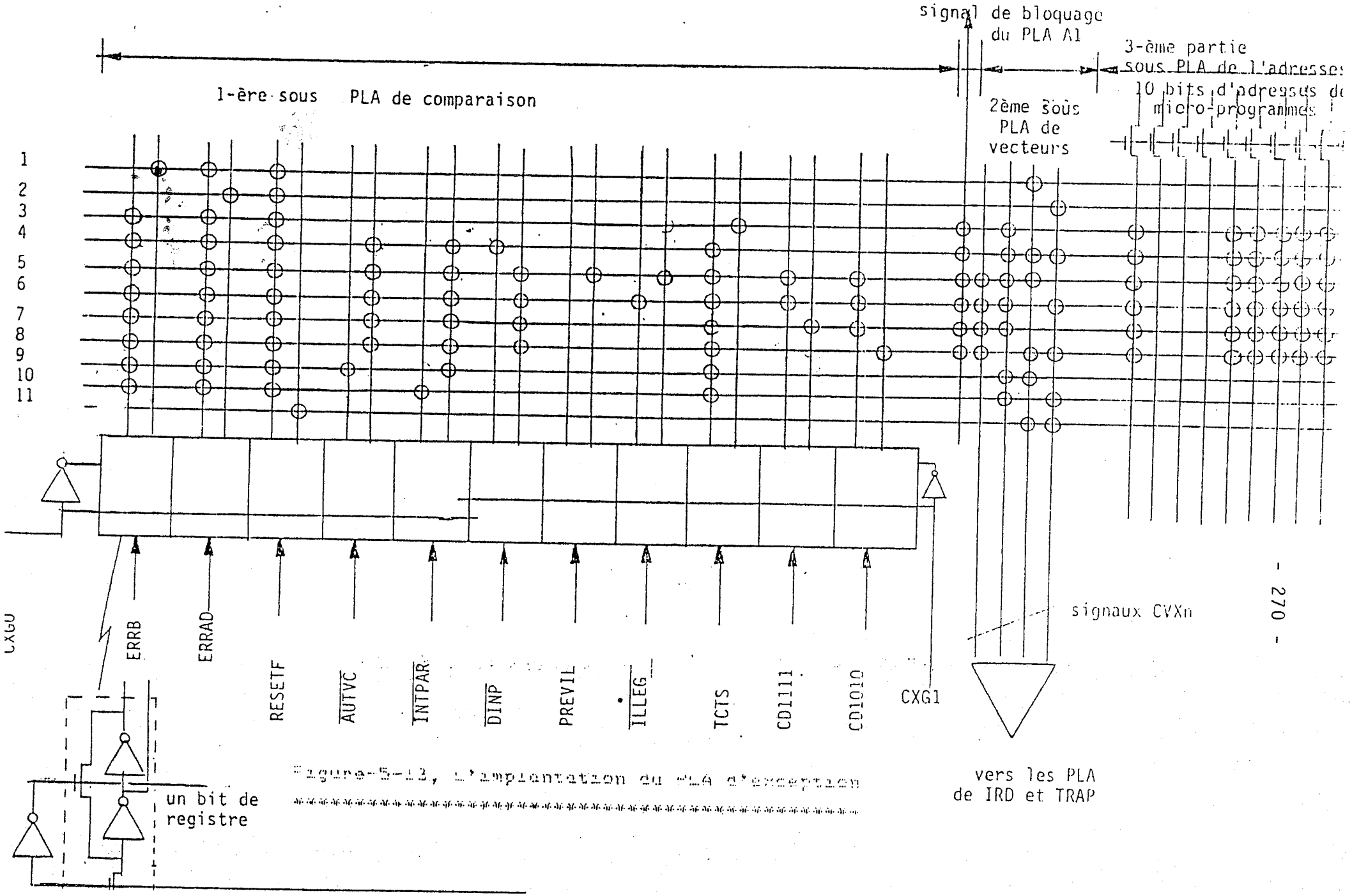


Figure-5-13, L'implantation du PLA d'exception

1-18-Sous PLA de priorité

Pour chaque commande d'exception, deux bits (direct et complémentaire) entrent dans le sous-PLA de priorité. Ces deux bits sont issus du registre d'entrées.

L'implantation des transistors de cette partie impose une priorité de fonctionnement pour les exceptions prioritaires Autrement dit, les colonnes plus prioritaires de ce PLA en cas d'activation masquent l'effet des colonnes moins prioritaires. Chaque ligne appartient à une des exceptions qui reste activée aprés la comparaison des entrées.

Le signal RESETF ayant la priorité la plus élévée est capable de masquer toutes les autres exceptions. Ce signal est la troisième entrée autorisée par le signal CXG0. L'ordre de priorité correspond au regroupement des exceptions presenté dans le tableau d'exceptions.

Chaque ligne du PLA A0, réservée pour un signal d'exception, génére le vecteur et l'adresse du micro-programme correspondant. Les lignes numérotées sur le PLA de la figure V-1-13, sont activées pour les exceptions suivantes:

No. de ligne	exception concernée	signal véhiculant cette exception
1	L'erreur de bus	ERRB
2	L'erreur d'adresse	ERRAD
3	En cas de mode TRACE et TEST	TCTS
4	Interruptions (touts les niveaux)	DINP'
5	L'état privilégié	PREVIL
6	Les codes op. invalides	ILLEG'
7	Détection du code op. 1010	CD1010
8	Détection du code op. 1111	CD1111
9	L'interruption auto-vectorisée	AUTVC'
10	L'interruption parasite	INTPAR'
11	L'initialisation par RESET	RESETF

L'activation des lignes du PLA A0 pour chaque exception.

1-19-La structure du PLA de vecteur d'exception

La deuxième partie du PLA-A0 envoie quatre fils de commandes, CVX1->4, vers le PLA IRD. Ce PLA, associé au PLA TRAP, décode les vecteurs d'exceptions. Ces vecteurs définis par le tableau de vecteurs d'exceptions sont envoyés aux plots d'adresses à travers la P.O. Le tableau suivant présente ces vecteurs. En cas de reset le vecteur 000 est envoyé sur les plots d'adresses pour accéder au pointeur de pile superviseur. C'est le SSP initial qui est traité d'abord par le reset. L'adresse du compteur ordinal est prise à l'adresse 004(Hexa) où le processeur trouve les codes d'initialisation du système. La séquence de reset presentée précedamment , appèle le PLA-A0, pour initialiser les vecteurs correspondants.

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
–	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Unitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		–
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		–
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		–
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		–

Le tableau des vecteurs d'exceptions

****

1-20-Commande de chargement du PLA A0

Le chargement du PLA-A0 est issu d'une logique de décodage activée par les bits I,B et C de la micro-instruction . La figure 5-14, présente cette logique. Les différents champs de la micro-instruction en cas de séquencement normal (bit B=0) et en cas de branchement (bit B=1) sont présentés. Les bits I,C et B commandent la logique de génération de CXG1 et CXG0. La commande CT provenant du circuit de test est un signal actif à l'état haut. Ce dernièr independant des entrées I,B et C, permet d'activer la sortie CXG1. En cas de fonctionnement normal, si B=0, CXG1 prend la valeur C'. I et CXG0 est égale à C. En cas de branchement B=1, CXG0 est inactive (CXG0=0) et CXG1=I. Les deux tables de verités suivantes présentent l'état des sorties CXG1 et CXG0.

SI B=0

I	C	I	I	I	CXG1	CXG0
0	I	0	I	0	I	0
0	I	1	I	1	I	0
1	I	0	I	0	I	1
1	I	1	I	0	I	1

SI B=1

I	C	I	I	I	CXG1	CXG0
X	I	0	I	0	I	0
X	I	1	I	1	I	0

CXG0 active les quatre colonnes du PLA de génération du numéro du vecteur d'exception. Le PLA de génération de l'adresse de micro-programme n'est, par contre, pas affecté par cette commande.

CXG1 active tous les PLA; d'une part cette commande initialise le vecteur d'éxception (2 ème sous-PLA) d'autre part elle génère les 10 bits d'adresses de début du traitement d'exceptions.

Le traitement d'une macro-instruction commance par le chargement

du registre d'instruction avec le code opération. Ceci est commandé par le bit I de la micro-instruction. L'activation de CXG1 est aussi issue de ce bit , ce qui engage la prise en compte du PLA A0. Donc, en cas d'exception le traitement de la macro-instruction n'est pas considéré et l'adresse du micro-programme d'exception est envoyée à la ROM de contrôle. Ce fonctionnement assure le priorité des exceptions par rapport au traitement normal des macro-instructions.

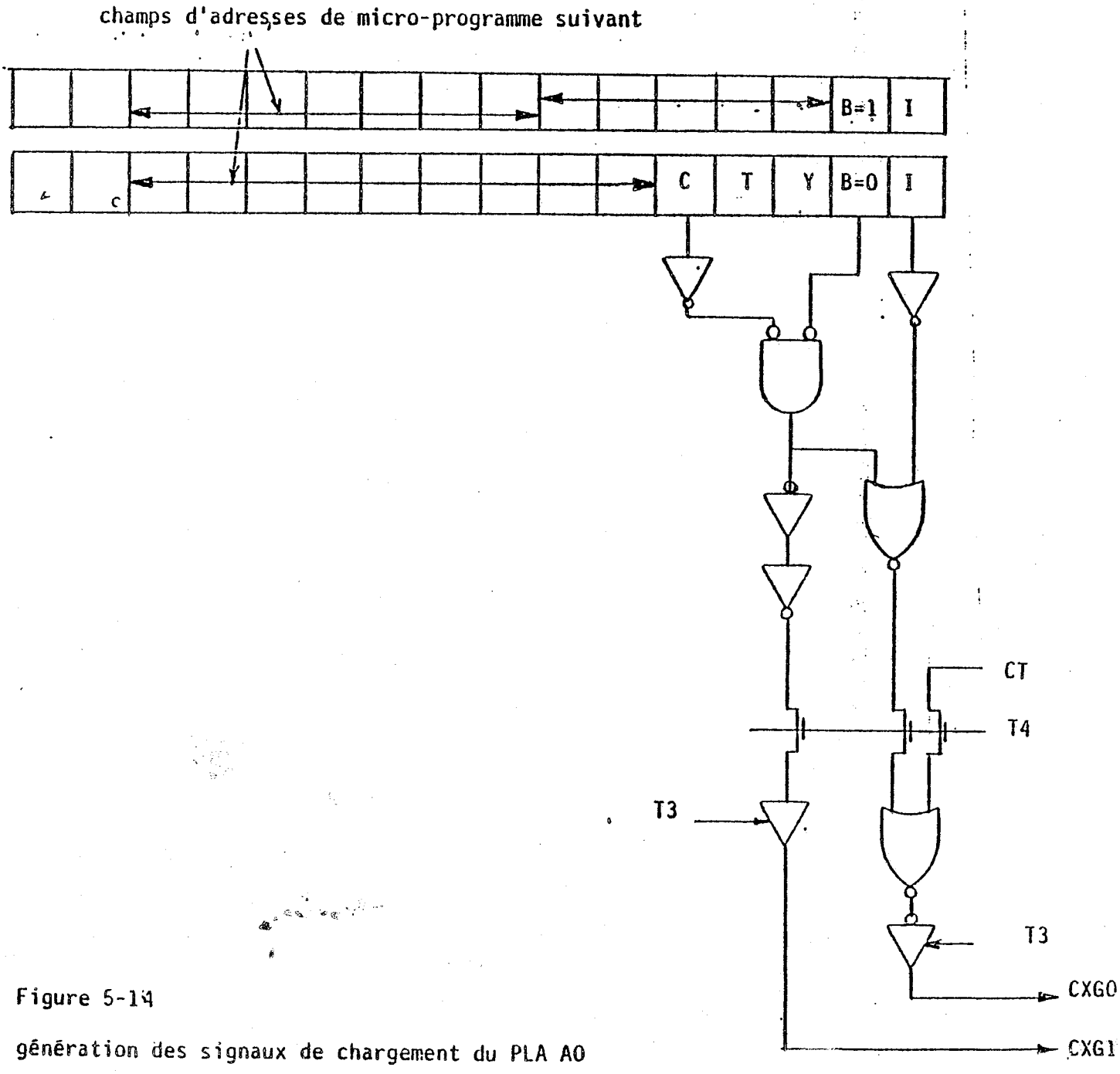


Figure 5-14

génération des signaux de chargement du PLA A0

CXG0 charge les signaux d'exception sans intervenir sur la génération de l'adresse du micro-programme. En fait, la génération de l'adresse du micro-programme du groupe "0" correspondait à reset, erreur de bus et d'adresse qui est organisée différemment. Les quatre signaux d'initialisation du vecteur d'exception sont également générés.

Le signal CXG1, charge les signaux de requète d'exceptions générant une adresse de micro-programme. Les quatre sorties d'initialisation des vecteurs d'exceptions sont également générées par le deuxième PLA.

1-21-Autorisation de l'état HALT

Les signaux d'exceptions du groupe zero activent la porte NOR 1 de la figure 5-16, à travers ce PLA. La sortie EHA' peut être générée à partir de chacun des ces signaux, assurant l'affectation du PLA-A0. Le signal EHA' est chargé de la prise en compte de l'état halt en cas de presence de l'erreur de bus et d'erreur d'adressage simultané.

1-22-Regroupement des signaux d'entrées du PLA-A0

Les entrées du PLA-A0 sont regroupées selon la logique de priorité des exceptions. Elles peuvent être classées suivant les groupes de priorité:

RESETF	I
ERRAD	I---> Group "0"
ERRB	I
TCTS	I
INTPAR'	I
AUTVC'	I
IPEND	I---> Group "1"
CD1010	I
CD1111	I
ILLEG'	I
PRIVIL	I

l'état inactif du PLA A0	I
traitement des instructions:	I
TRAPV	I--> Group "2"
CHK	I
DIV. Illégale	I

La prise en compte du PLA A0 par les routines de traitements des macro-instructions est considérée à la fin de chaque procédure, au moment où le bit C de la micro-instruction prend en compte la demande de traitement d'exception. La prise en compte du PLA A1 à la fin du traitement d'une macro-instruction est contrôlée par le PLA-A0. La figure 5-12, montre cette priorité de fonctionnement par les deux portes ET 2 et 3.

1-23-Priorité des exceptions

La prise en compte d'une des exceptions bloque la génération par le PLA A1 de l'adresse de la micro-instruction suivante . Par concequent l'adresse de traitement de cette exception est choisie comme étant l'adresse de la micro-instruction suivante. Cette

logique de choix entre les PLA est située sur le chemin de validation du PLA-A1. Les exceptions du groupe "1" sont traitées de cette manière. tandis que les exceptions du groupe zero, ont une priorite supérieur et forcent directement l'adresse de micro-programme. Cette priorité de fonctionnement est garantie par la modification imposée sur le fonctionnement des générateurs d'horloges.

Le groupe "2" d'exception se trouve dans la séquence normale de traitement. des macro-instructions. Autrement dit les exceptions de ce groupe sont traitées comme des macro-instructions. La conception de ces logiques répond au tableau de priorité ainsi qu'au regroupement des exceptions. Ce tableau a été presenté lors les specifications du MC68000 au chapitre IV du présent rapport.

1-24-Les signaux d'exception d'entrées au PLA-A0

ERRB :

Signal généré à partir du plot BER', presentant une erreur de bus. La condition de détection de cette erreur correspont à un temp trop long entre l'activation du plot AS' et la réponse du périphèrique.

ERRAD :

Signal généré de façon interne par l'apparition d'une adresse fause par processeur.

AUTVC' :

Signal activé par le plot VPA' en cas d'arrivée d'interruptions.

INTPAR':

En cas de reconaissance simultanée d'erreurs de bus ou d'interruptions ce signal génère la reconnaissance d'une interruption parasite.

DINP° :

Signal de demande d'interruption est issu de la logique d'interruptions.

PRIVIL:

signal généré par les PLAs de décodage A2 et A3 décodant une instruction de mode privilégié (si le bit S du registre d'état n°est pas à "1").

ILLEG° :

Détection d'un code opération invalide par le PLA des codes invalides.

TCTS :

En cas de fonctfonnement normal en mode "trace" ce signal est activé. Il est issu du registre d'état, le produit du bit T et la ligne test (mode test).

CD1010 et CD1111:

Ces signaux proviennent du PLA A1 et indiquent les codes opérations des types: 1010 et 1111.

II-TRAITEMENT DES INTERRUPTIONS

2-1-Les plots d'interruptions

Les trois plots d'entrées IPL0',IPL1' et IPL2' annoncent la présence d'une interruption. Ces bits présentent 8 codages possibles pour les demandes d'interruption. Le niveau zero (aucune des broches IPLn' n'est activées) correspond à l'état de non-interruption. Par contre le niveau sept de demande d'interruption (priorité la plus élévée) correspond à l'activation de tous les plots IPL'0->2. IPL0' est le bit poids faible et IPL2' représente le bit de poids fort.

2-2-Le comportement du processeur en cas d'interruption

Sept niveaux de priorité d'interruption sont prévus. Les peripheriques externes peuvent demander l'interruption du processus en cours à travers l'un de ces sept niveaux. Un nombre non limité de périphériques, peut être chaînés par l'intermèdiaire de transcodeurs d'interruptions.

Les niveaux d'interruptions sont numérotées de 1 à 7. Le niveau sept ayant la priorité la plus élevée peut masquer les autres demandes d'interruption. Le registre d'état contenant 3 bits de masque d'interruption, indique le niveau de priorité d'interruption courant. Les interruptions ayant un niveau de priorité inférieur sont ainsi masquées.

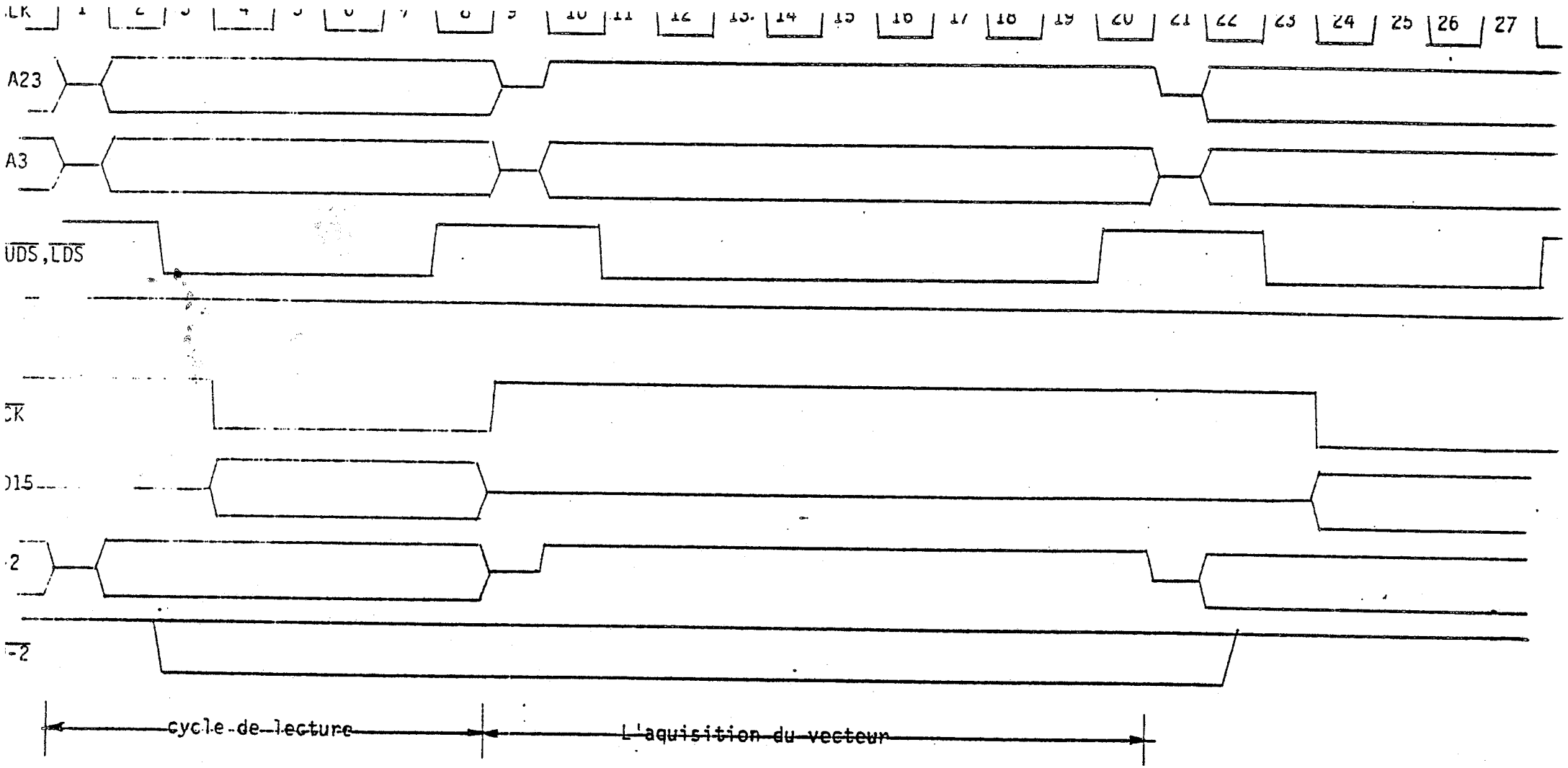
Un transcodage du niveau d'interruption des plots IPLn', active les commandes d'interruption internes, pour le niveau demandé.

Les requètes d'interruption ne forçent pas immédiatement le processeur en état d'exception. En effet elles ne sont pas détectées durant l'éxécution d'une macro-instruction. Si la priorité du niveau d'interruption demandé est inférieure ou égale au niveau d'interruption courante, l'éxecution de l'instruction suivante va continuer. Si le niveau de priorité de la nouvelle interruption est supérieur, la séquence de traitement d'exception commence. Les opérations suivantes sont effectuées par cette

séquence.

1- Une copie du registre d'état est gardée et la bascule S est positionnée à l'état superviseur. Si on se trouve en mode "trace" ce mode est supprimé et le niveau de priorité est positionné à celui qui vient d'être reconnu. Le processeur demande le numéro du vecteur de cette interruption, par l'intermèdiaire des plots de données, et annonce le niveau de priorité d'interruption reconnu , sur les bits d'adresses A1,A2 et A3. Dans le cas d'une demande auto-véctorisée, le processeur génère un vecteur d'interruption, selon le niveau demandé. Le diagramme des temps de la figure 5-15, indique le positionnement des plots du processeur en cas d'arrivée d'interruptions.

Si une logique externe, annonce une erreur de bus, en cas de traitement d'interruptions, celle-ci est considérée comme une fausse interruption. Le numèro du vecteur d'interruption généré, référence le vecteur d'interruption parasite. Le processeur continue le traitement d'exception. Il sauvegarde le compteur ordinal et le contenu du registre d'état en sommet de la pile (où il est référencé par le pointeur de pile superviseur). Le contenu du vecteur d'interruption déjà obtenue est référencé, et le compteur ordinal est chargé avec cette nouvelle valeur. L'éxécution normale commence, par le traitement de la routine d'interruption. La figure suivante présente l'horlogerie du MC68000, en cas d'une demande d'interruption.



L'horlogerie de traitement d'interruptions (vu des plots)
Figure 5-15

Le niveau de priorité sept est un cas particulier car il ne peut pas être inhibé par le masque de niveau de priorité. C'est le cas d'interruption non-masquable du MC68000. Une interruption est générée chaque fois qu'un niveau d'interruption est d'un niveau inferieur vers le niveau sept.

Les vecteurs d'interruptions sont présentés par le tableau suivant:

interruption	I espace de mém.	I adresses (Hexa)
fause interr.	I S D	I 060
niveau 1 d'interr.	I S D	I 064
niveau 2 d'interr.	I S D	I 068
niveau 3 d'interr.	I S D	I 06C
niveau 4 d'interr.	I S D	I 070
niveau 5 d'interr.	I S D	I 074
niveau 6 d'interr.	I S D	I 078
niveau 7 d'interr.	I S D	I 07C
interruption auto-véctorisée	I S D	I 100 I

Les vecteurs d'interruptions réservés pour le traitement d'interruptions

2-3-Le system d'interruption

L'interruption demandée par les plots IPLn', intervient dans le séquencement du MC68000 tel qu'il est presenté par la figure 5-16. La logique d'interruption traite les demandes d'interruption afin de générer un signal de reconnaissance d'interruption prioritaire. Ce signal nommé DINP' est envoyé au PLA d'exception PLA A0. Le signal de validation du PLA A0 issu du bit I de la micro-instruction, prend en compte l'arrivée du signal DINP' à la fin du traitement de chaque macro-instruction. Lorsque DINP' est actif (mis à zero), ce signal d'exception sera mis en comparaison avec les autres exceptions. Le fonctionnement de la logique d'exception et la structure du PLA-A0 est étudié dans la partie précédente. Donc la priorité de prise en compte du signal DINP' est inférieure à celle de reset quelque soit le niveau demandé.

Dans le cas où les signaux plus prioritaires ne sont pas actifs, le traitement d'exception de l'interruption sera initialisé par le PLA-A0. Les signaux prioritaires correspondent aux exceptions du groupe "0", y compris le signal RESETF. En cas de coincidence d'arrivée de reset avec une interruption la demande d'interruption, sera masquée par le reset.

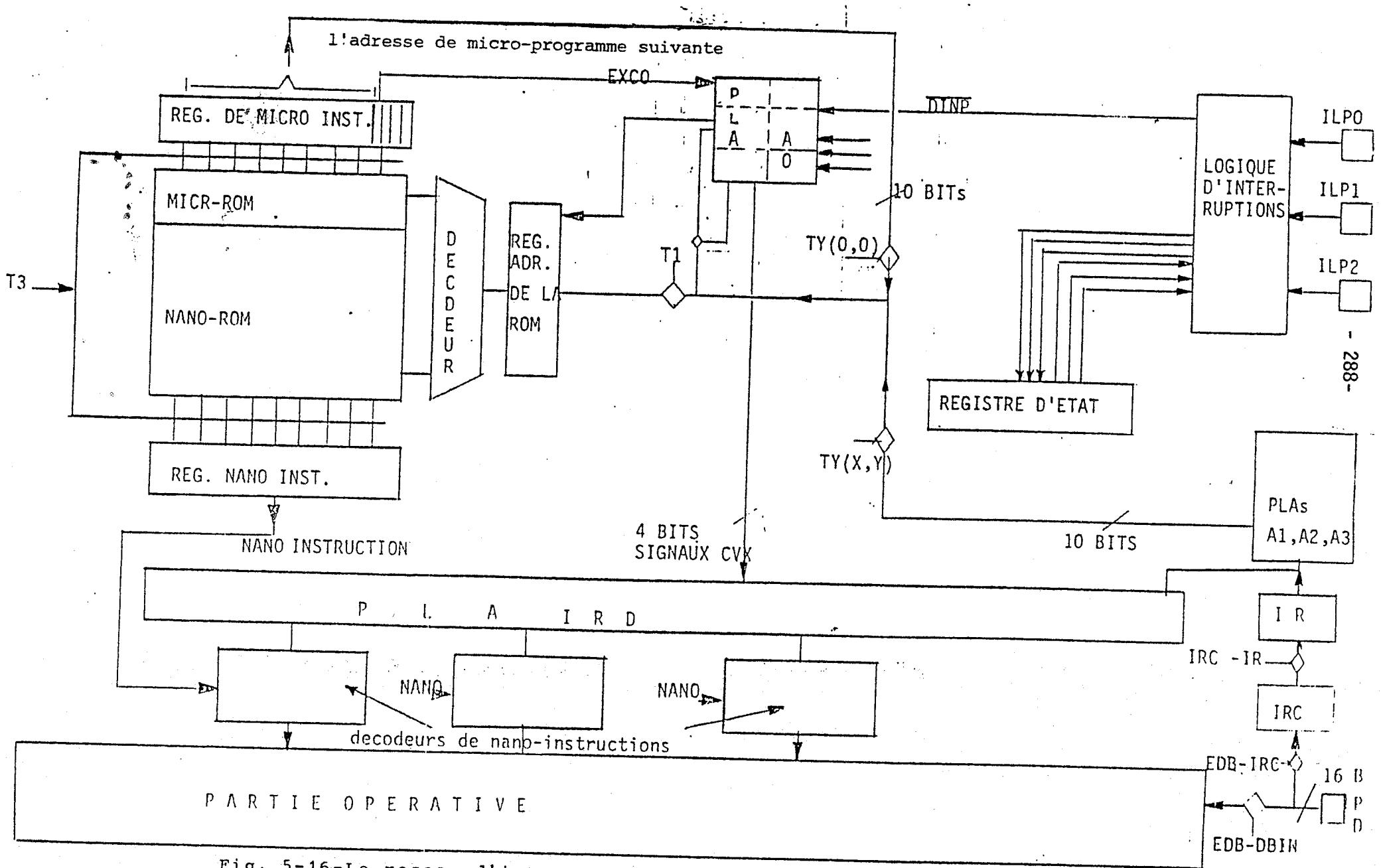


Fig. 5-16-Le reseau d'interruptions du MC68000

****

Lors de la reconnaissance d'interruption par le signal DINP', le PLA-A0 envoie l'adresse du traitement des micro-instructions concernées aux décodeurs de la rom de micro-programmes. Cette adresse est 1C4, en Hexadecimal, qui correspond au début de la séquence d'interruption. Quatre signaux véhiculent le codage du vecteur d'exception. Ceux-ci sont envoyés au PLA-IRD pour un calcul eventuel du vecteur d'exception. Les vecteurs sont chargés dans la partie opérative au cours de l'exécution des interruptions.

2-4-Le bloc d'admission d'interruption

Ce bloc est constitué des quatre sous-blocs suivants:

1- bloc de détection d'apparition d'une demande d'interruption différente.
2- Les registres de mémorisation d'interruption
3- Le PLA de contrôle de priorité
4- Le générateur de demande d'interruption DINP'.

Le circuit du bloc d'admission d'interruption est presenté par la figure 5-17.

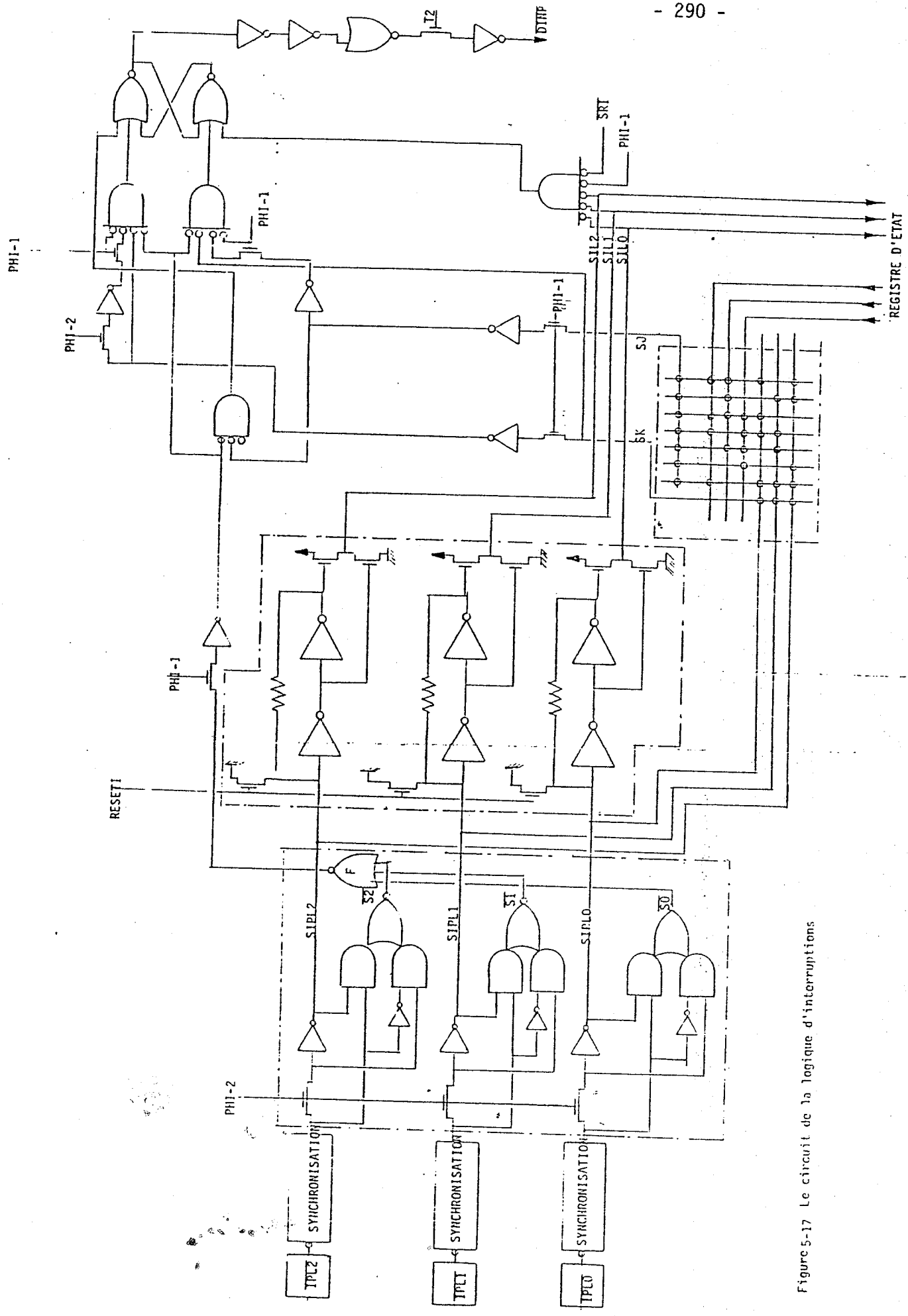


Figure 5-17 Le circuit de la logique d'interruptions

Ce circuit peut être présenté par les sous blocs suivants

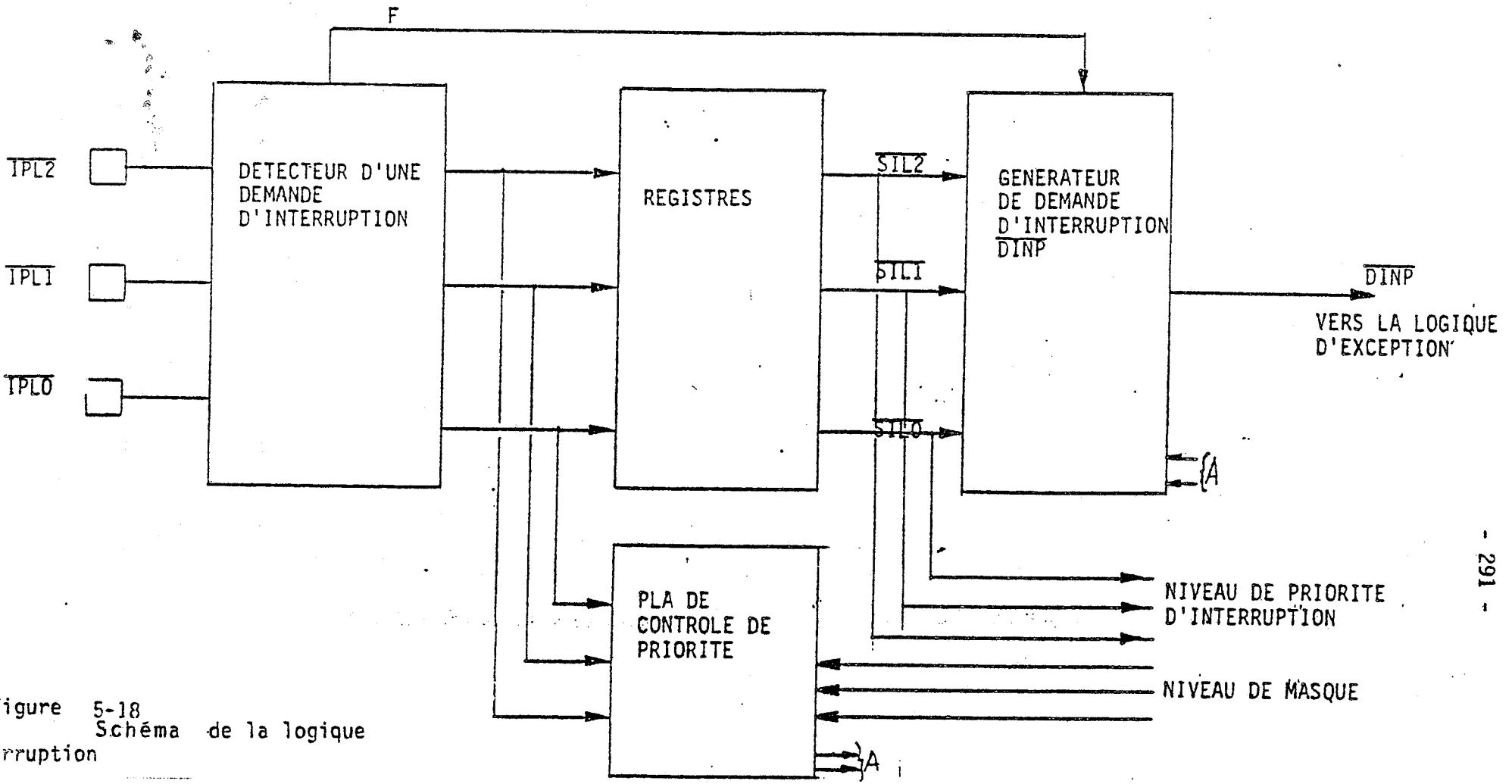


Figure 5-18
Schéma de la logique interruption

1- Détecteur d'une demande d'interruption

Ce circuit a pour fonction de détecter un niveau d'interruption différent , sur les plo ts IPLn'. Ce bloc est constitué de trois cellules identiques, admettant les signaux des plots. Une cellule est presentée par la figure 5-19.

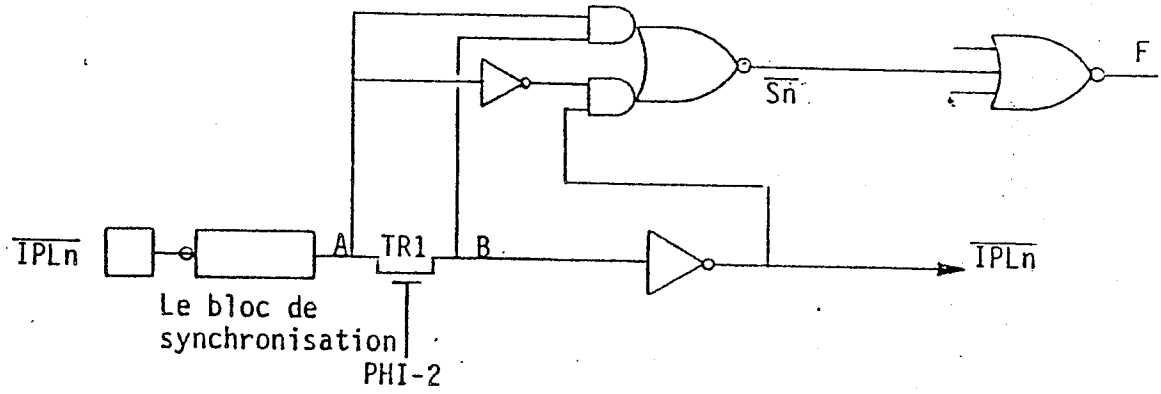


Figure 5-19-Le détécteur d'apparition d'une interruption

****

Le signal RESETI, provenant du plot de reset', est conduit vers la logique d'interruption. L'entrée de chaque registre d'interruption est forçée à zero par le signal RESETI. L'intervention de RESETI sur les interruptions, met ce circuit au niveau 7 d'interruptions, car toutes les entrées SIPLn' sont actives à l'état bas.

Dans ce cas-là aucune interruption ne peut pas être admise par la logique de contrôle.

Le transisitor TR1 impose plusieurs contraintes sur le chargement des registres. La commande de la grille de TR1 est conditionnée par CHIR et TST. CHIR échantillonnée sur T4, provient de la logique de chargement du registre d'instruction. Ce signal est issu du bit (I) de la micro-instruction. Il est généré si le traitement de la macro-instruction au cours de laquelle l'interruption est demandée, a été terminé. C'est au moment du

chargement du registre d'instruction que les registres d'interruption sont chargés par CHIR. CHIR est valide sur T3 donc les registres d'interruption seront validés sur le temps T1 suivant.

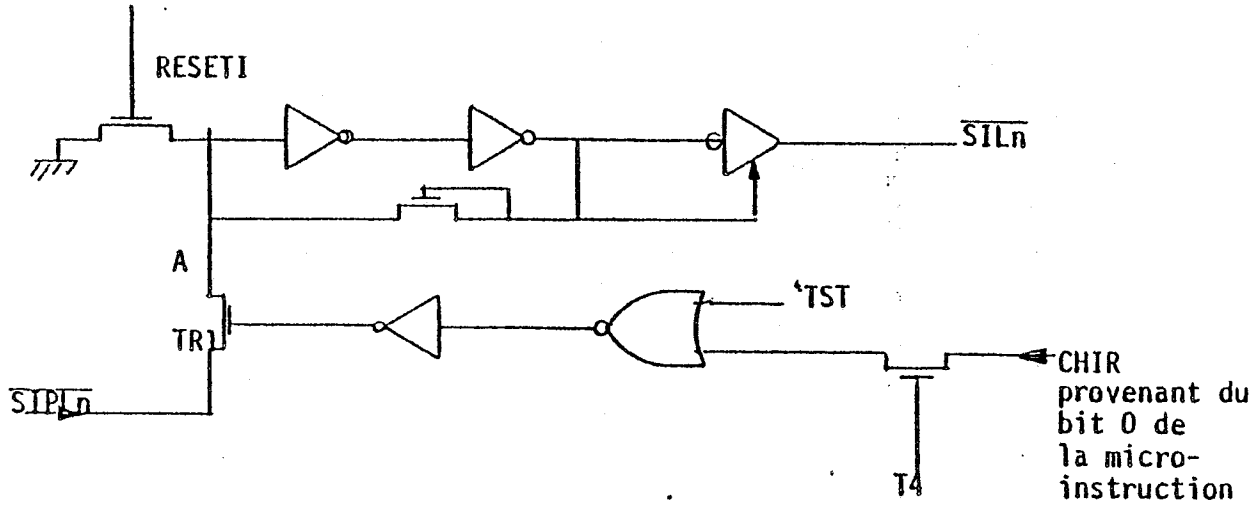


Figure 5-20 Le registre d'interruption..

****

3- Le PLA de contrôle de priorité

Deux series des fils entrent dans ce PLA. Les premiers sont les 3 bits du niveaux d'interruption demandé. Ces bits sont présentés par l'entrée SIPLn'.

La deuxième serie correspond aux bits du masque d'interruption issus du registre d'état (RETn). Deux sorties SK et SJ, annoncent le résultat de la comparaison.

Les IPLn' sont actifs à l'état bas et les RETn sont actifs à l'état haut. Le principe de comparaison est basé sur l'implantation et le fonctionnement du PLA. A chaque monôme (MA à MH), qui n'est pas mis à zero par les bits RETn, correspond un où plusieurs codages du niveau d'interruption que ce monôme laisse à 1.

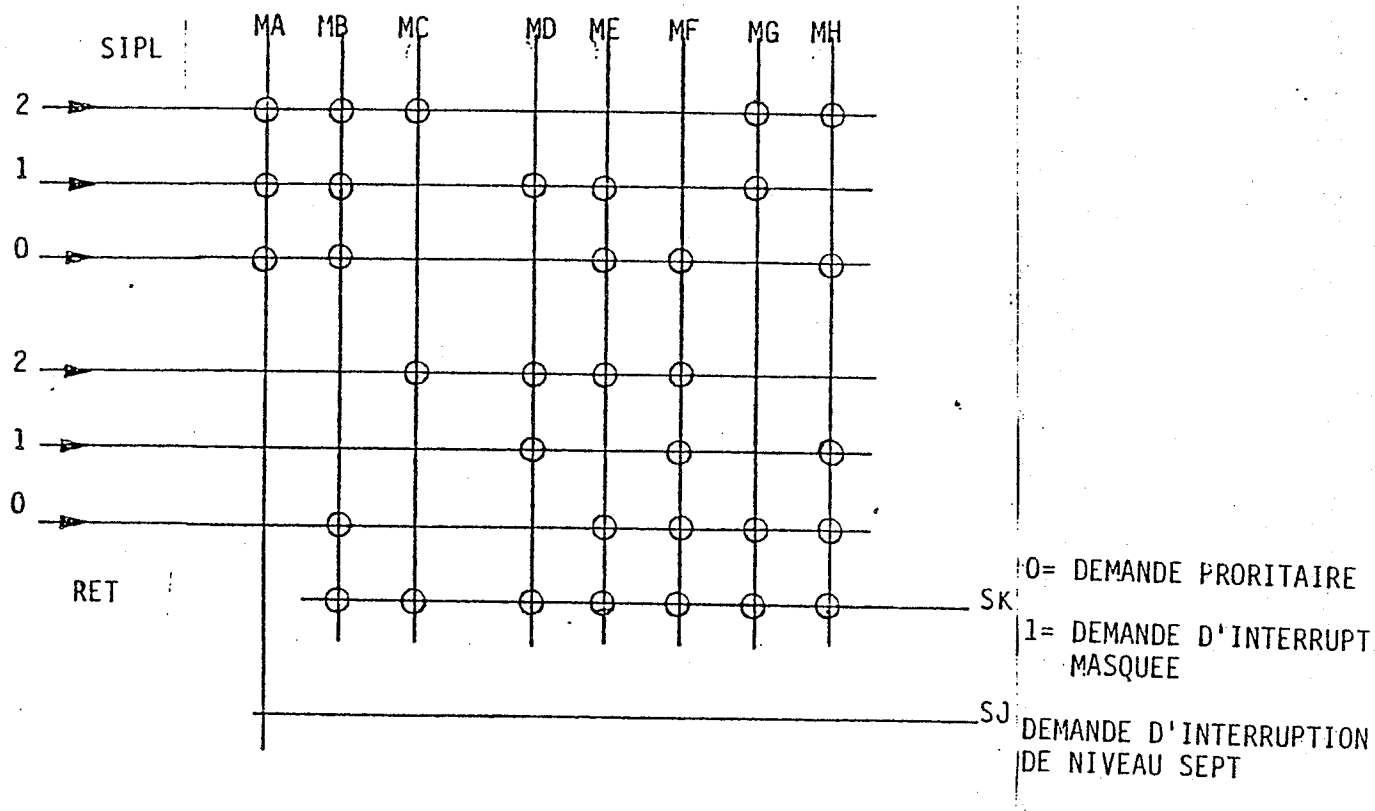


Figure 5-21- Le PLA de comparaison des niveaux d'interruptions

****

En effet un monôme de ce PLA restera à 1, si l'interruption demandée est de niveau supérieur au niveau courant. L'entrée d'un niveau d'interruption supérieur laisse au moins l'une des colonnes de ce PLA à 1, et contrairement à un niveau inferieur où égal, met tous les monôme à zero. Le tableau suivant comprend les monomes qui ne sont pas mis à zero par le niveau d'interruption d'entrée pour les différents niveaux de masques.

Niveau du masque.	monomes non mis à zero	Niveau d'interruption demandé laissant la monome à 1
0	B	7
	C	4,5,6,7
	D	2,3,6,7
	E	3,7
	F	1,3,6,7
	G	6,7
	H	6,7
1	C	4,5,6,7
	D	2,3,6,7
	G	6,7
2	B	7
	C	4,5,6,7
	E	6,7
3	C	4,5,6,7
4	B	7
	G	6,7
	H	6,7
5	G	6,7
6	B	7
7	A	7

Le tableau de comparaison de priorité

****

Les sorties SK et SJ annoncent les résultats de comparaison suivants:

La sortie SK est à zéro pour une demande d'interruption prioritaire, et elle est à 1 pour une demande masquée.
La sortie SJ, est à 1 pour une demande de niveau 7.

4- Le bloc de génération du signal DINP'

Ce circuit a pour fonction de générer le signal DINP' qui est envoyé au PLA-A0. Ce dernier comprend le résultat final de la comparaison, et la prise en compte des conditions pour admettre une interruption. Ce bloc reçoit les entrées suivantes:
SK, qui indiqe la détection d'une demande d'interruption prioritaire.
SJ, une demande d'interruption de niveau 7.
Le signal SF est issu du circuit de détection d'une nouvelle demande d'interruption. Cette sortie est mise à zero par l'apparition d'une nouvelle interruption sur PHI-1. En PHI-2 suivant SF est à nouveau à 1.
Les signaux SILn' sont les sorties des registres d'interruption.
SIR', signal de reconaissance d'interruption. Le processeur répond aux interruptions par la mise à 1 des sorties FC0-2. dans ce cas, SRI' est mis à zero.

La sortie DINP' est admise par la logique d'exception sur T3, à la fin du traitement de la macro-instruction en cours. Ceci est autorisé par bit 0 de la micro-instruction. C'est à cause de ce classement que la priorité des interruptions y comris celle du niveau 7, est inférieure à celle du reset.

DINP', reste activée jusqu'au chargement du registre d'état, commandé par la routine de traitement de cette interruption. Après la mise à jour du registre d'état, le nouveau masque entre dans le PLA de priorité. Ce circuit se trouvant en égalité de masque, l'interruption met alors les signaux SK et SJ à 1, et DINP' est désactivé.

2-5-La logique d'exception et les intérruptions

La demande d'interruption, génère le signal DINP', par l'intermèdiaire de la loqique d'interruption. Ce signal est validé jusqu'à ce que l'interruption soit prise en compte. Celui-ci est pris en compte par la logique d'exception (le PLA-A0). En effet c'est le signal CXG0 issu du bit 0 de micro-instruction (I), qui le valide. L'initialisation de la séquence d'interruption et la mise à jour du registre d'état annule le signal DINP'.

La réponse de processeur au périphérique demandant l'interruption, est issue des deux bits de micro-instruction 15 et 16. Ces deux bits sont positionnés pour mettre à 1 les sorties FC0-2. La séquence d'interruption démarre à partir de l'adresse 1C4 en hexadecimal. Cette adresse est également fournie par le PLA-A0. Ce PLA prenant en compte l'état d'exception du processeur, génére cette adresse. Il est évident que l'état d'exception doit être plus prioritaire que celui d'interruption de processeur.

2-6-L'interruption parasite et auto-vecteur

A partir du signal DINP', deux autres signaux interviennent dans le reseau d'interruption. L'un est provoqué par la logique d'erreur de bus et l'autre est issu du plot VPA'.

Si durant la validation des FC0-2, une erreur de bus se déclare, la ligne INTPAR' est activée. Celle-ci est connectée au PLA-A0. par conséquent le INTPAR', indique la présence d'une interruption parasite.

Le mode d'interruption autovectorisée est distingué par le plot VPA', activé par le monde extérieur. Deux cas peuvent etre différenciés selon l'état du VPA':

1- VPA'=0, le processeur génère le vecteur d'interruption pour le cas auto-vecteur. Le signal AUTVC' est activé par cet état.

cas auto-vecteur. Le signal AUTVC' est activé par cet état.
2- VPA'=1, Le périphérique par l'intermédiaire des 8 bits de données envoie le vecteur d'interruption.

2-7-Codage des différents modes d'interruptions sur les signaux TVNn.

Les sorties du PLA-A0, prennent les combinaisons suivantes pour les différents cas d'interruptions.

TVN	3	2	1	0
INTPAR ACTIVE	0	1	0	1
AUTV ACTIVE	0	1	1	0
Interruption fournie par le peripherique	0	1	1	1

2-8 La séquence d'interruption

La procédure de traitement d'interruption se déroule à patir de l'adresse 1C4. Les 15 adresses du micro-programme suivant sont parcourues dans la rom, constituant la séquence de reconnaissance d'interruption.

adresse	nom- état
1C4	grilr
234	time3
235	time4
0EB	brill
236	time5
118	bnra2
292	oplw2
360	vccx7
0FF	cfrt3
367	rtta3
11A	httyu
2B7	vvrw1
11C	cfrty
363	rtta2
B4C	vccx3

Séléction du PLAA1

ADRESSE PRESENT	NOM ETAT	MICRO INSTRUCTION 15,16	adresse future	C X Y 0	B I
0111000100	grilr	0 0	1000110100	0 0 0	0 0
1000110100	time3	0 0	1001110100	0 0 0	0 0
1000110101	time4	0 0	0011101011	0 0 0	0 0
0011101011	brill	0 1	1010110100	0 0 0	0 0
1000110110	time5	1 1*	0100011000	0 0 0	0 0
0100011000	bnra2	0 0	1010010010	1 0 0	1 0
1010010010	oplw2	0 0	1100100001	0 0 0	0 0
1101100000	vccx7	0 1	0011101111	0 0 0	0 0
0011101111	cfrt3	0 1	1111100101	0 0 0	0 0
1101100111	rtta3	0 1	0110011000	0 0 0	0 0
0100011010	httyu	1 0	1011110110	0 0 0	0 0
1010110111	vvrw1	1 0	0100011100	0 0 0	0 0
0100011100	cfrty	0 0	1111100001	1 0 0	0 0
1101100011	rtta2	1 0	1100001101	0 0 0	0 1
1101001100	vccx3	0 0	0000000000	0 0 1	0 0

tableau des micro instructions executées par l'interruption

* reconnaissance d'interruption

III- CODES INVALIDES

3-1-Instructions invalides et instructions non-implémentées

Les codes opérations invalides correspondent à des profils de bits dont la configuration n'appartient pas à une instruction légale. Pendant l'exécution normale d'un programme si une telle instruction est détectée, l'exécution du micro-programme d'instruction invalide est initialisé.

Les codes opérations non-implémentés, correspondent aux instructions dont les bits 15 à 12 ont les configurations 1010 et 1111. La détection de tels codes non-implémentés conduit le processeur à l'exécution des micro-programmes de traitement des instructions non-implémentées. Ainsi trois types de vecteurs sont générés pour les codes invalides, ces vecteurs sont les suivants:

L'adresse en Hexa	Espace	Type de code invalide
010	SD	Illégale
028	SD	ligne 1010, émulateur
02C	SD	ligne 1111, émulateur

3-2-Traitement des instructions invalides

1-Les instructions invalides.

Les codes opérations invalides sont détectées par le PLA de codes invalides. Celui ci est chargé de décoder les profils de codes opérations invalides dans le jeu d'instruction du MC68000. Le résultat de ce décodage est la génération du signal ILLEG', envoyé au PLA d'exception PLA-A0. Ce signal conduit le processeur au traitement de l'exception concernée.

2- Les instructions non-implémentées

Le PLA-A1 décode ces instructions pour activer les signaux CD1010 pour les codes opérations du type 1010 et CD1111 pour les

codes opérations du type 1111. Ces deux signaux sont envoyés au PLA-A0 pour initialiser les requètes d'exceptions concernées.

3-3-Traitement des codes opérations invalides

Le détail de la logique d'exception est presenté dans le chapitre consacré à ce sujet. Les signaux d'entrée du PLA-A0, ILLEG', CD1010 et CD1111, se trouvent dans le premier groupe d'exceptions, ils sont donc moins prioritaire que le reset. Ces exceptions initialisent la séquence de traitement des codes invalides. Les vecteurs d'exceptions et l'adresse du micro-programme correspondants sont génères par ce PLA. Le niveau de priorité des interruptions représentées par les signaux INTPAR', AUTVC et DINP' est le même que celui des codes invalides se trouvant dans le groupe 1.

3-4-Logique de chargement des PLA de décodage

La figure 5-22 présente les différents éléments engagés pour décoder les codes opérations.

Le bus de données est relié à la partie opérative via deux buffers, qui permettent l'entrée/sortie de données. La liaison des plots de données à la partie contôle est réalisée à travers deux registres: IR et IRC. Les commandes de chargements de ces registres sont (EDB)--> IRC et (IRC)--> IR.

Le chargement de IR (registre d'instruction) à l'aide du code operation, envoie les 32 bits (directs et complémentés), aux PLA de décodage.

La logique de chargement de ces registres, est combinée à un commande issue du mode test. Lorsque le signal TEST n'est pas actif, on peut présenter la logique de chargement des registres par la figure 5-23.

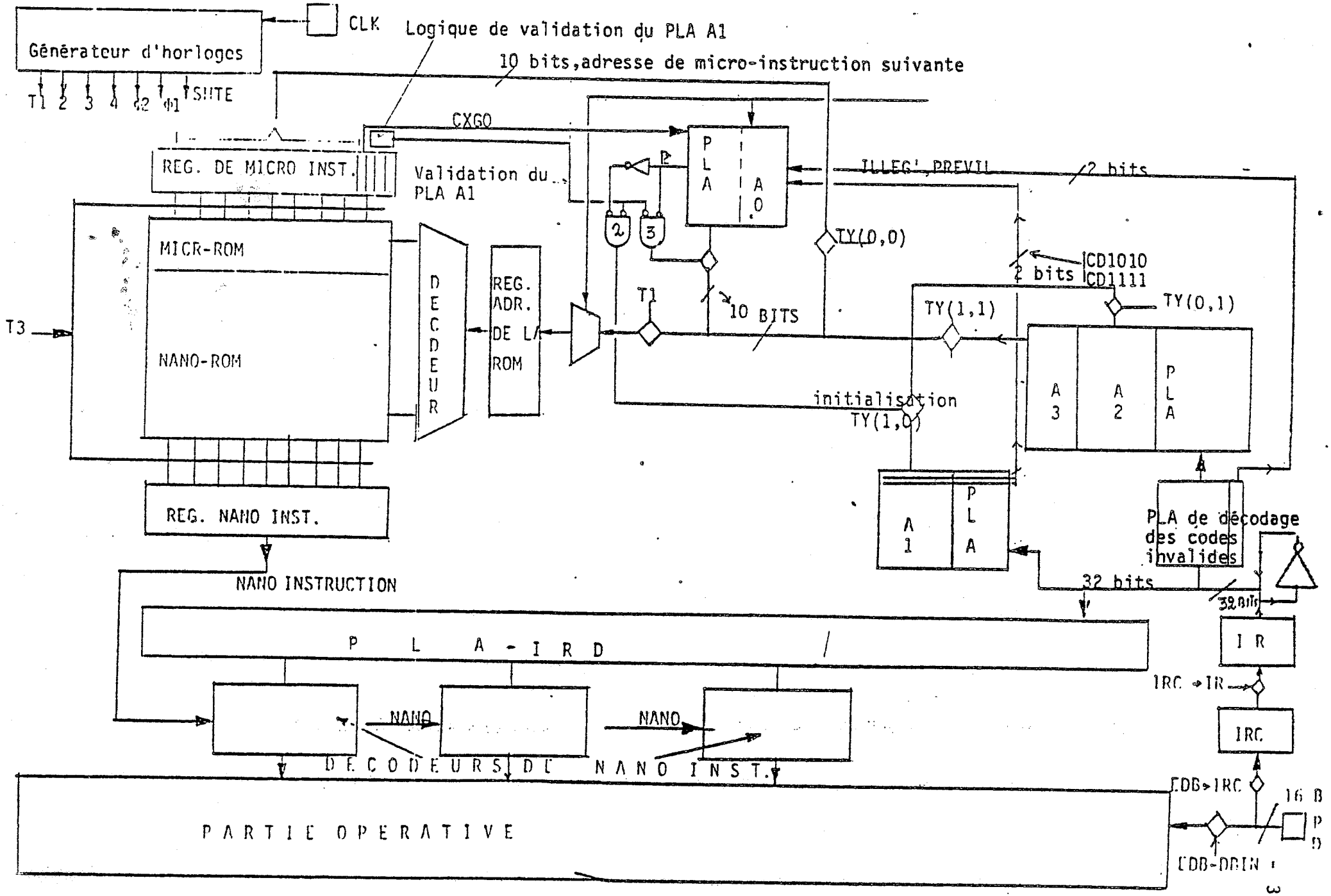

Générateur d'horloges
CLK
Logique de validation du PLA A1
10 bits,adresse de micro-instruction suivante
T1 2 3 4 Φ2 Φ1 SUITE
CXGO
REG. DE MICRO INST.
Validation du PLA A1
P L A
A 0
ILLEG', PREVIL
2 bits
MICR-ROM
NANO-ROM
DECODEUR
REG. ADR. DE L ROM
T3
T1
10 BITS
TY(0,0)
CD1010
CD1111
2 bits
TY(1,1)
TY(0,1)
initialisation
TY(1,0)
A 3
A 2
P L A
A 1
P L A
PLA de décodage des codes invalides
REG. NANO INST.
NANO INSTRUCTION
32 bits
32 bits
P L A - I R D
I R
IRC → IR
IRC
NANO
NANO
DECODEURS DE NANO INST.
EDB→IRC
16 B
P
D
EDB-DBIN
PARTIE OPERATIVE

Figure 5-22

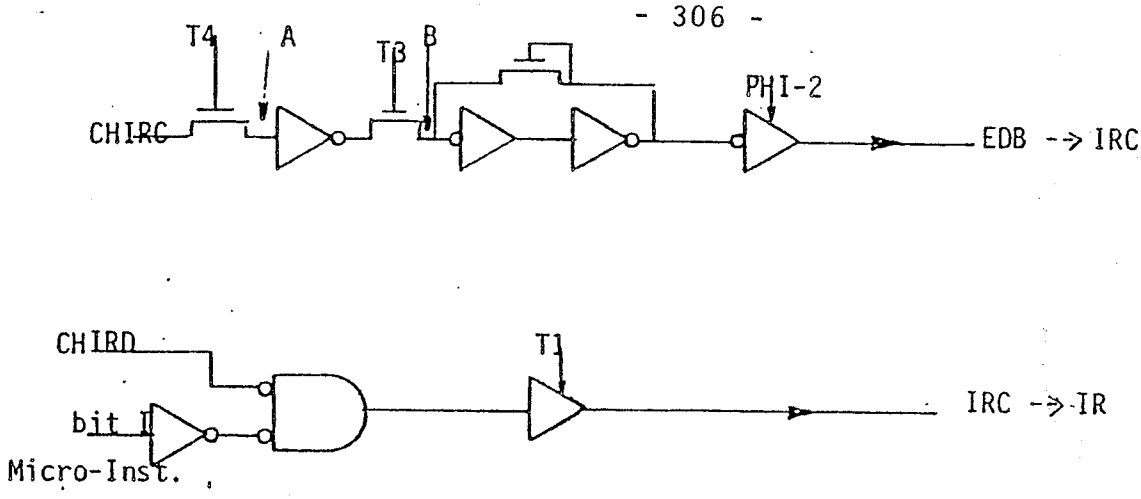


Figure 5-23 Logique de chargement des registres IR et IRC

****

Les commandes de chargement sont issues des bits de la nono-instruction et le bit I de la micro-instruction. On distingue un décalage temporaire à cause des éléments de synchronisation sur le chemin des signaux. Les deux chronogrammes suivants expliquent le timing des signaux EDB->IRC et IRC-->IR.

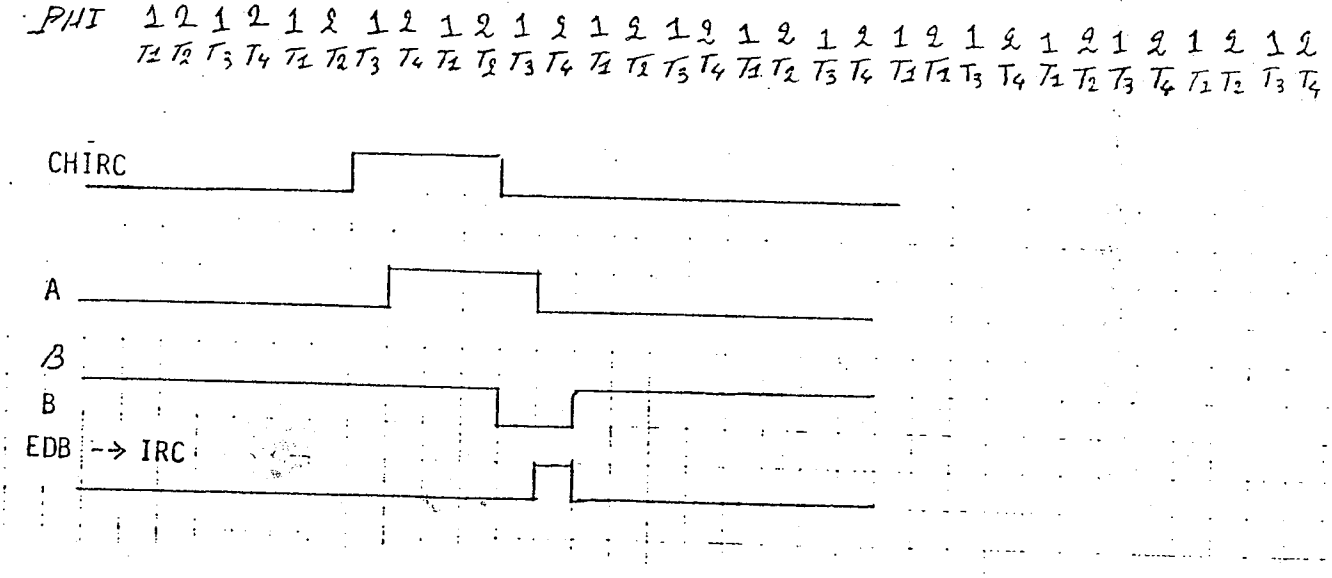


Figure 5-24-A Chronogramme de chargement du registes IRC.

****

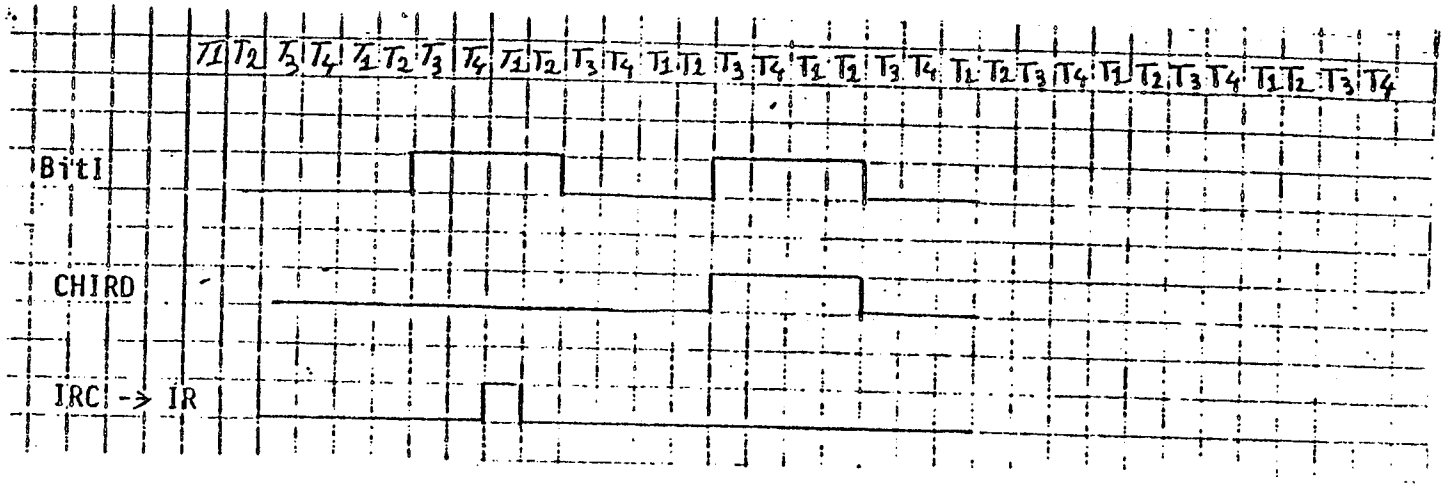


Figure 5-24-B Chronogramme de chargement du registre IR

****

3-5 Détection des codes invalides

Les bits du registre IR, sont envoyés aux PLA. Les deux monômes du PLA-A1, qui décodent les profils 1010 et 1111, sont à charge statique. Les deux signaux générés, CD1010 et CD1111, sont envoyés au PLA-A0.

Le PLA de décodage des codes invalides

Ce PLA est chargé de décoder les profils des codes non-utilisés. Les 32 bits directs et complémentés du registre instruction activent des lignes du PLA de codes invalides. 69 monomes sont décodés; l'implantation des transistors est presentée à la figure 25. Un certains nombre de monômes sont mis en ET logique avec d'autres monômes pour décoder des profils particuliers.

La partie OU de ce PLA est une ligne activée par chacun des monômes de la partie ET. Le signal ILLEG', est déduit directement de cette ligne. Le signal de détection des codes privilégiés est aussi activé par des monômes de ce PLA.

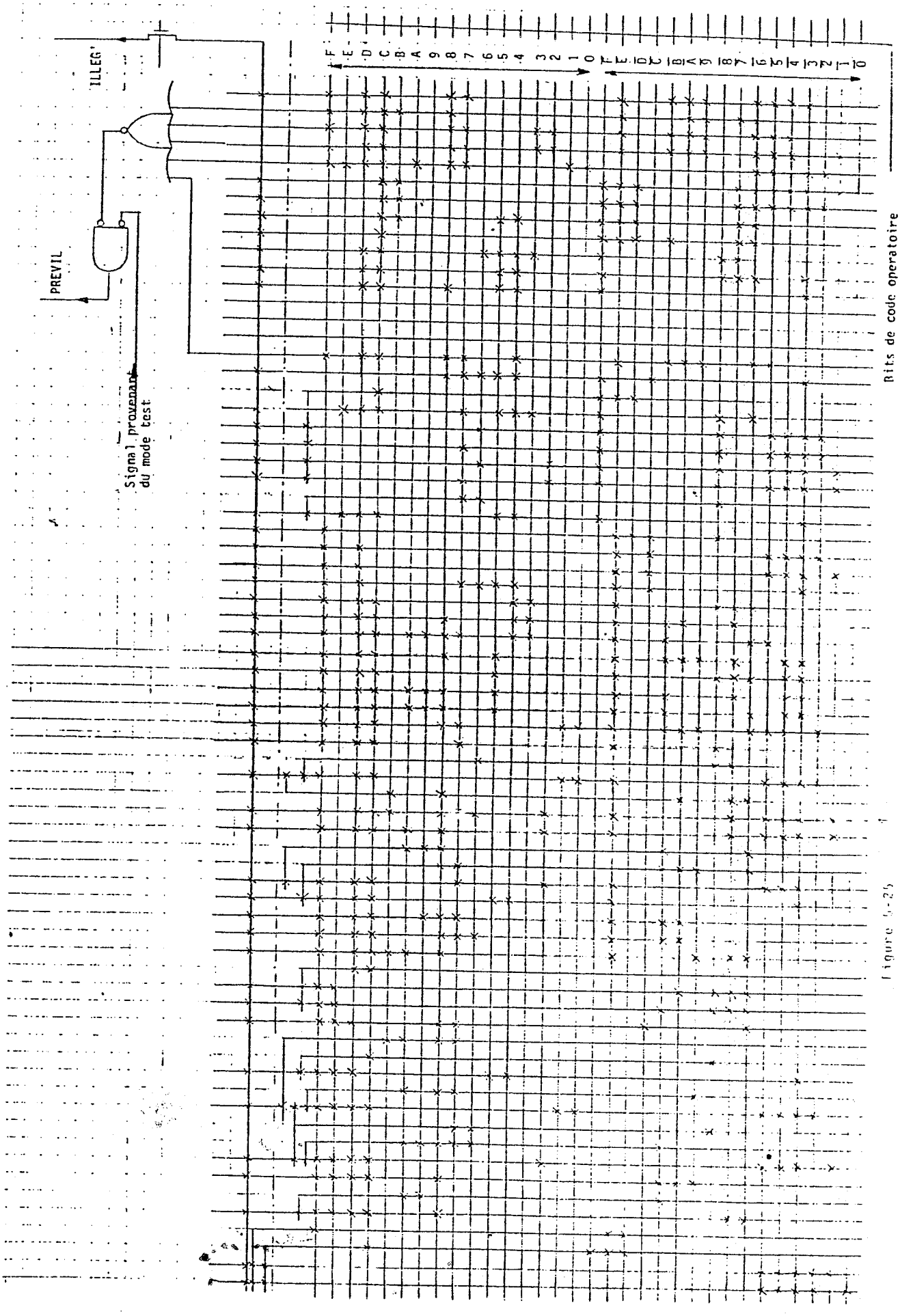


Figure 5-25-L'implantation du PLA des codes invalides

****

d'instruction est présentédans par le tableau suivant:

instruction ori	nombre de codes valides	153	nombres de codes invalides	103 soit	4.823444E1 %
instruction andi	nombre de codes valides	153	nombres de codes invalides	103 soit	4.823444E1 %
instruction subi	nombre de codes valides	158	nombres de codes invalides	104 soit	4.148628E1 %
instruction addi	nombre de codes valides	158	nombres de codes invalides	104 soit	4.148628E1 %
instruction eori	nombre de codes valides	153	nombres de codes invalides	103 soit	4.823444E1 %
instruction cmpi	nombre de codes valides	158	nombres de codes invalides	104 soit	4.148628E1 %
instruction btst	nombre de codes valides	474	nombres de codes invalides	38 soit	7.421875 %
instruction bchg	nombre de codes valides	458	nombres de codes invalides	42 soit	1.218994E1 %
instruction bclr	nombre de codes valides	458	nombres de codes invalides	42 soit	1.218994E1 %
instruction bset	nombre de codes valides	458	nombres de codes invalides	42 soit	1.218994E1 %
instruction movep	nombre de codes valides	256	nombres de codes invalides	0 soit	0.000000 %
instruction inv.	nombre de codes valides	0	nombres de codes invalides	256 soit	1.000000E2 %
instruction movea	nombre de codes valides	976	nombres de codes invalides	568 soit	3.645833E1 %
instruction move	nombre de codes valides	8758	nombres de codes invalides	2882 soit	1.881985E1 %
instruction addq	nombre de codes valides	1328	nombres de codes invalides	288 soit	1.354167E1 %
instruction subq	nombre de codes valides	1328	nombres de codes invalides	289 soit	1.354175E1 %
instruction scc	nombre de codes valides	888	nombres de codes invalides	96 soit	1.071429E1 %
instruction dbcc	nombre de codes valides	128	nombres de codes invalides	0 soit	0.000000 %
instruction bra	nombre de codes valides	256	nombres de codes invalides	0 soit	0.000000 %
instruction bsr	nombre de codes valides	256	nombres de codes invalides	0 soit	0.000000 %
instruction bcc	nombre de codes valides	3584	nombres de codes invalides	0 soit	0.000000 %
instruction moveq	nombre de codes valides	2048	nombres de codes invalides	2048 soit	5.000000E1 %
instruction divu	nombre de codes valides	424	nombres de codes invalides	88 soit	1.718750E1 %
instruction divs	nombre de codes valides	424	nombres de codes invalides	88 soit	1.718750E1 %
instruction sbcd	nombre de codes valides	128	nombres de codes invalides	0 soit	0.000000 %
instruction or	nombre de codes valides	2288	nombres de codes invalides	464 soit	2.255434E1 %
instruction subx	nombre de codes valides	976	nombres de codes invalides	48 soit	4.687500 %
instruction subx	nombre de codes valides	384	nombres de codes invalides	0 soit	0.000000 %
instruction sub	nombre de codes valides	2488	nombres de codes invalides	288 soit	1.041667E1 %
instruction cmp	nombre de codes valides	1488	nombres de codes invalides	136 soit	8.854167 %
instruction cmpa	nombre de codes valides	976	nombres de codes invalides	48 soit	4.687500 %
instruction cmpm	nombre de codes valides	192	nombres de codes invalides	0 soit	0.000000 %
instruction eor	nombre de codes valides	1288	nombres de codes invalides	144 soit	1.071429E1 %
instruction muls	nombre de codes valides	424	nombres de codes invalides	88 soit	1.718750E1 %
instruction mulu	nombre de codes valides	424	nombres de codes invalides	88 soit	1.718750E1 %
instruction abcd	nombre de codes valides	128	nombres de codes invalides	0 soit	0.000000 %
instruction exgd	nombre de codes valides	64	nombres de codes invalides	0 soit	0.000000 %
instruction exgd	nombre de codes valides	64	nombres de codes invalides	0 soit	0.000000 %
instruction exga	nombre de codes valides	64	nombres de codes invalides	64 soit	5.000000E1 %
instruction and	nombre de codes valides	2288	nombres de codes invalides	488 soit	1.517857E1 %
instruction addx	nombre de codes valides	384	nombres de codes invalides	0 soit	0.000000 %
instruction adda	nombre de codes valides	976	nombres de codes invalides	48 soit	4.687500 %
instruction add	nombre de codes valides	2488	nombres de codes invalides	288 soit	1.041667E1 %
instruction asl.r	nombre de codes valides	852	nombres de codes invalides	172 soit	1.679688E1 %
instruction lsl.r	nombre de codes valides	852	nombres de codes invalides	172 soit	1.679688E1 %
instruction roxl.r	nombre de codes valides	852	nombres de codes invalides	172 soit	1.679688E1 %
instruction rol.r	nombre de codes valides	852	nombres de codes invalides	172 soit	1.679688E1 %
instruction negx	nombre de codes valides	158	nombres de codes invalides	42 soit	2.187500E1 %
instruction move f sr	nombre de codes valides	58	nombres de codes invalides	14 soit	2.187500E1 %
instruction clr	nombre de codes valides	152	nombres de codes invalides	184 soit	4.166667E1 %
instruction neg	nombre de codes valides	158	nombres de codes invalides	42 soit	2.187500E1 %
instruction move t ccr	nombre de codes valides	58	nombres de codes invalides	14 soit	2.187500E1 %
instruction not	nombre de codes valides	158	nombres de codes invalides	42 soit	2.187500E1 %
instruction move t sr	nombre de codes valides	53	nombres de codes invalides	11 soit	1.718750E1 %
instruction nbcd	nombre de codes valides	58	nombres de codes invalides	14 soit	2.187500E1 %
instruction swap	nombre de codes valides	8	nombres de codes invalides	0 soit	0.000000 %
instruction pea	nombre de codes valides	28	nombres de codes invalides	28 soit	5.000000E1 %
instruction ext	nombre de codes valides	16	nombres de codes invalides	0 soit	0.000000 %
instruction movem r m	nombre de codes valides	68	nombres de codes invalides	44 soit	3.906250E1 %
instruction tst	nombre de codes valides	158	nombres de codes invalides	42 soit	2.187500E1 %
instruction tas	nombre de codes valides	58	nombres de codes invalides	14 soit	2.187500E1 %
instruction movem m r	nombre de codes valides	72	nombres de codes invalides	184 soit	7.187500E1 %
instruction jsr	nombre de codes valides	28	nombres de codes invalides	28 soit	5.000000E1 %
instruction jmp	nombre de codes valides	28	nombres de codes invalides	36 soit	5.625000E1 %
instruction trap	nombre de codes valides	16	nombres de codes invalides	0 soit	0.000000 %
instruction unlk	nombre de codes valides	8	nombres de codes invalides	0 soit	0.000000 %
instruction link	nombre de codes valides	8	nombres de codes invalides	0 soit	0.000000 %
instruction move f usp	nombre de codes valides	8	nombres de codes invalides	0 soit	0.000000 %
instruction move t usp	nombre de codes valides	8	nombres de codes invalides	0 soit	0.000000 %
instruction reset	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction nop	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction stop	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction rte	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction rts	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction trapv	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction rtr	nombre de codes valides	1	nombres de codes invalides	0 soit	0.000000 %
instruction lea	nombre de codes valides	184	nombres de codes invalides	296 soit	5.078125E1 %
instruction chk	nombre de codes valides	474	nombres de codes invalides	96 soit	1.718750E1 %

Tableau des codes invalides pour chaque serie d'instruction

****

3-6 Priorité de traitement des codes invalides

La figure 5-26, présente la logique par laquelle, la priorité de traitement des codes invalides est assurée. Les deux dernières entrées du PLA-A0 activent les monômes M7 et M8. Chacun de ces monômes peut activer le signal P. Les deux portes ET 2,3 valident l'une des deux séries d'interrupteurs. Par conséquent si aucun des signaux d'exception, y compris ceux des codes invalides, ne sont valides alors l'adresse du micro-programme est décidée par PLA-A1. Par contre, la reconaissance d'un code invalide active le signal P et l'adresse du micro-programme désignée par PLA-A0 est choisie. Celle-ci, constituée de 10 bits d'adresse est envoyée au registre d'adresse de micro-programme

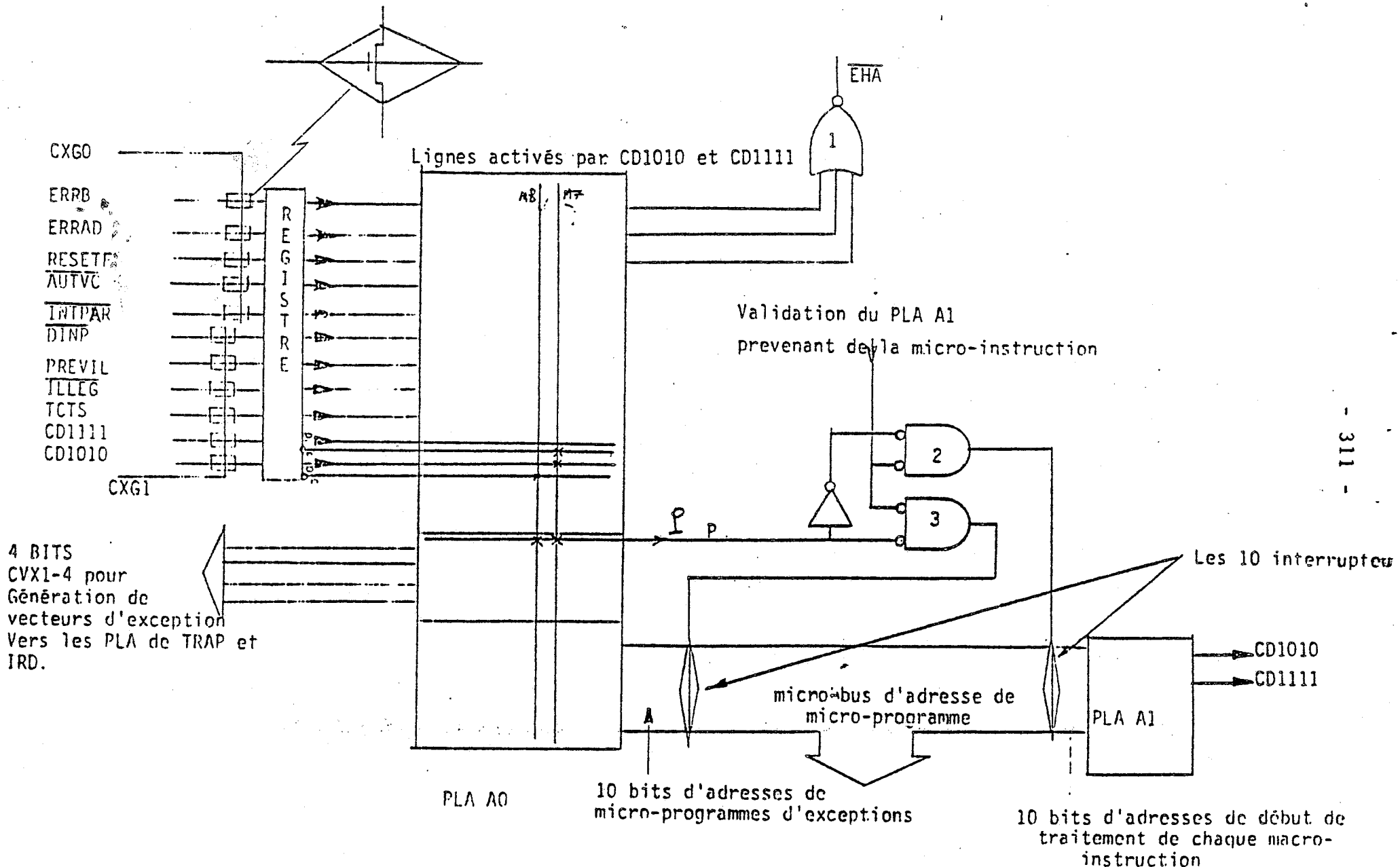


Figure 5-26- La logique de priorité de traitement des codes invalides

****

La prise en compte du PLA-A0 à la fin de l'exécution de l'instruction courante, envoie l'adresse du micro-programme 1C0 pour trois signaux ILLEG', CD1010 et CD1111. Les signaux CVXn (n:1-->4), sont eventuellement positionnés pour présenter les vecteurs concernant les types d'instruction (invalide ou non-implémentée). L'adresse 1C0, lance la routine de traitement des codes invalides, constituée des 11 états suivants.

adresse binaire	nom micro-inst.	adresse hexa	Micro-inst
0111000000	grilr	1C0	00 1111100010 000 00
1110100011	aase1	3A3	00 1110000011 000 00
1111000010	ffed8	3C2	01 1100100001 000 00
1101100000	vccx7	360	01 0011101111 000 00
0011101111	cfrt3	0EF	01 1111100101 000 00
1101100111	rtta3	367	01 0110011000 000 00
0100011010	htyyu	11A	01 1011110110 000 00
0110110111	vvrw1	2B7	10 0100011100 000 00
0100011100	cfrty	11C	00 1111100001 100 00
1101100011	rtta2	363	10 1100001101 000 01
1101001100	vccx3	34C	00 0000000000 001 00

Séléction du PLA-A1

Les états vvrw1 --> vccx3 sont communs avec les séquences d'interruption et de reset. Durant l'execution de cette séquence le compteur ordinal et le registre d'état, sont sauvegardés. Le programme démarre à l'adresse mémoire correspondant au vecteur d'exception calculé à partir du type de code (invalide ou non-implémenté).

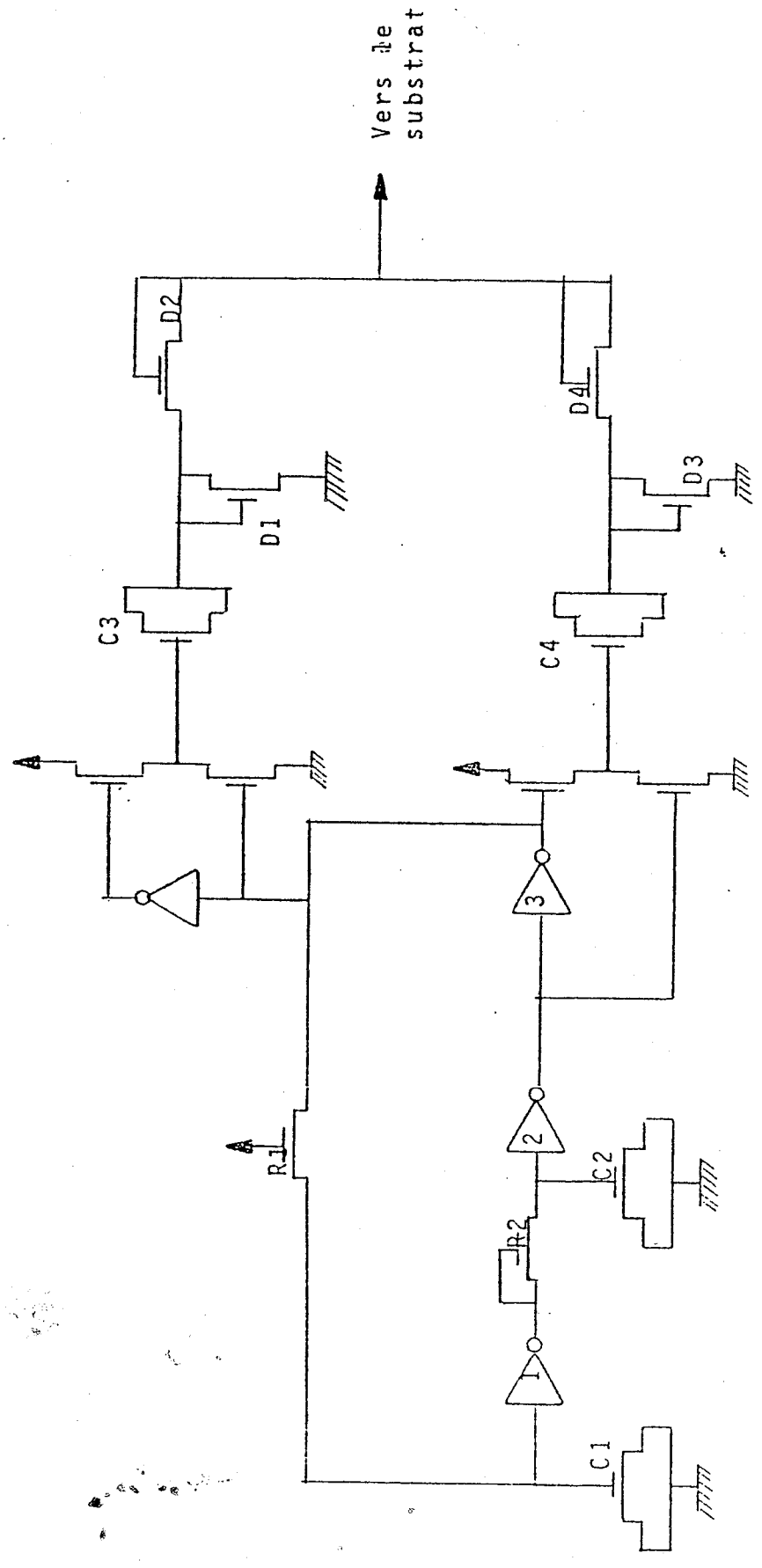
4- FONCTIONNEMENT SPECIAUX

4-1-Mise sous tension

L'application de la tension +5 V, au MPU, met en fonction tous les blocs du systeme. Le temps d'établissement d'une tension Vcc sur tous les points du microprocesseur dépend de la charge de l'ensemble CPU et les circuits associés. Un temps d'environ 100 millisecondes assure la stabilisation de Vcc. Un bloc de génération de tension négative polarise le substrat du microprocesseur. C'est un bloc interne au microprocesseur. La mise sous tension n'initialise pas le microprocesseur, seule l'application des signaux RESET et HALT externe peut l'initialiser.

4-1-1 Générateur de tension de substrat

Le substrat du MC68000 est chargé avec une tension négative de -2.5 Volts. Cette tension est obtenue par rapport aux sources des transistors canal N, diminuant la capacité entre drains et substrat. En effet la tension de seuil (Vth) est augmentée pour ces transistors. Le schéma de ce circuit de génération est donnée à la figure 5-27.



Fgiure 5-27 Générateur de polarisation du substrat

****

Ce circuit est constitué d'un oscillateur, alimenté par Vcc. Les inverseurs 1,2,3 forment cet oscillateur. La fréquence d'oscillation est déterminée par la constante du temps du circuit, constitué des résistances R1 et R2 et des capacités C1,C2. Les portes de puissance de sorties débitent le courant nécessaire pour charger le substrat. Les sorties des amplificateurs sont couplées aux diodes D1->D4, via les capacités C3 et C4. Par conséquent la tension alternative.charge le plot du substrat, négativement par les diodes de redressement. La connection Sb est reliée au substrat du microprocesseur. La mise sous tension démarre ce circuit pour atteindre une tension de -2.5 Volts. Le temps d'etablissement de Vb est de l'ordre de quelque milisecondes selon la courbe suivante:

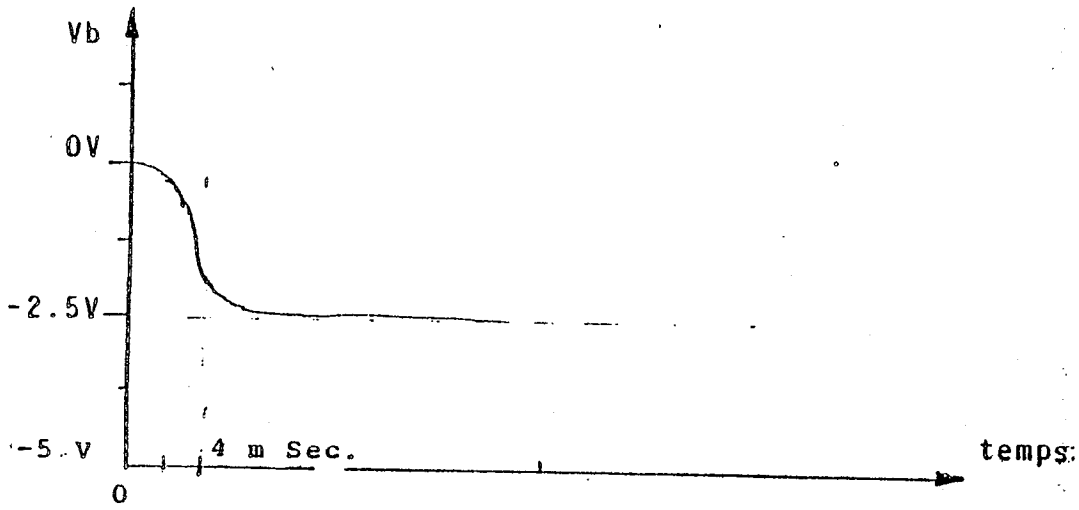


Figure- 5-28-Temp d'établissement de la tension de substrat.

****

Les tests éffectués sur le circuit indique une gamme de variation des fréquences d'oscillation. Cette fréquence varie entre quelque centaines de kilocycles jusqu'à une dixaine de mégacycles. Une fréquence d'oscillation basse ne peut pas maintenir la tension Vb au niveau de -2.5V, alors Vb augmente et le courant de fuite augmente.

A la mise sous tension,selon la présence ou l'absence du signal d'horloge CLK, deux cas peut être distingués:

1- Horloge arrêtée

La mise sous tension ne démarre pas le circuit, car le signal CLK n'existe pas. Les générateurs de signaux d'horloges internes PHI-1 et PHI-2 ne fonctionnent pas. Par conséquent les instants T1->T4, ne sont pas générés et le processeur ne peut pas fonctionner.

2- Horloge présente

Dans ce cas, la mise sous tension, aprés l'établissement d'un niveau acceptable, démarre le fonctionnement de la partie contrôle. L'état dans lequel, se trouve la partie contrôle est indéterminé, car une adresse quelconque de micro-programme peut être générée par les differents PLA. La présence de l'horlogerie normal, continue la séquencement jusqu' au moment où l'instruction suivante est chargée par le PLA-A1. C'est à partir de cette instruction que processeur fonctionne selon les codes opérations présent sur les plots de données. Afin de commencer le séquencement à partir d'un point connu, le plot reset doit être tiré à la masse durant une periode de 100 milisecondes de démarrage à la mise sous tension (établissement du Vcc sur tous les points du système). La broche Halt est activée en même temps pour éviter les parasites probables au début du signal reset. La mise sous tension et l'initialisation simultanée , réinitialise tout le systeme. Le fonctonnement est tel que celui décrit dans la partie consacré à ce sujet (paragraphe I).

4-2-1-Description de la séquence interne de démarrage lors d'un reset

Le paragraphe I détaille l'initialisation du processeur par la broche reset. La mise à zéro de cette broche, génère le signal RESETF. Ce signal réprésente la demande de réinitialisation. L'arrêt de l'horlogerie secondaire, et la génération des horloges d'exception, garantissent la prise en compte du signal reset. L'adresse de l'exception reset est forçée par RESETF et l'horlogerie d'exception. La séquence de reset est prise en compte pour la suite de l'initialisation. Le traitement en cours est abandonné et le microprocesseur exécute la séquence d'initialisation. La séquence de reset est déclanchée par la montée du signal reset sur le plot.

Les organes influencés par reset sont présentés par la figure 5-29. Le signal RESETI est la conséquence de la mise à zéro du plot Reset. La génération des quatre phases T1 -> T4 est affectée par le signal RESETI. Les trois phases T1->T3 sont arrêtées et T4 transmet une copie de PhI-2. L'horlogerie de l'exception de reset est générée par la suite. Celle-ci correspond aux signaux SHTE et T4 modifiée.
Les chronogrammes d'une telle modification d'horlogerie du MC68000 sont présentés à la figure 5-30.

Les autres blocs influencés par le signal RESETI sont:

1- Le bloc d'interruptions, mis au niveau 7 d'interruption. En effet aucune interruption n'est plus acceptée.

2- Le bloc d'initialisation de Reset soft.

3- Le PLA-A0 est commandé. Le prise en comte de toutes les exceptions est masquée (voir V-I-16).

4- Le registre d'adresse de micro-programme est forcé par l'adresse de première micro-instruction de reset.

L'horlogerie des différentes étapes de prise en compte du reset, l'arrêt des horloges et le démarrage des séquences de reset sont présentés à la figure 5-31. Le détail d'initialisation,

pour chaque micro-instruction est indiquée dans la figure 5-32. L'adresse de départ pour la version T6E, est 000(hexa) et pour celle de DL6, 002 .

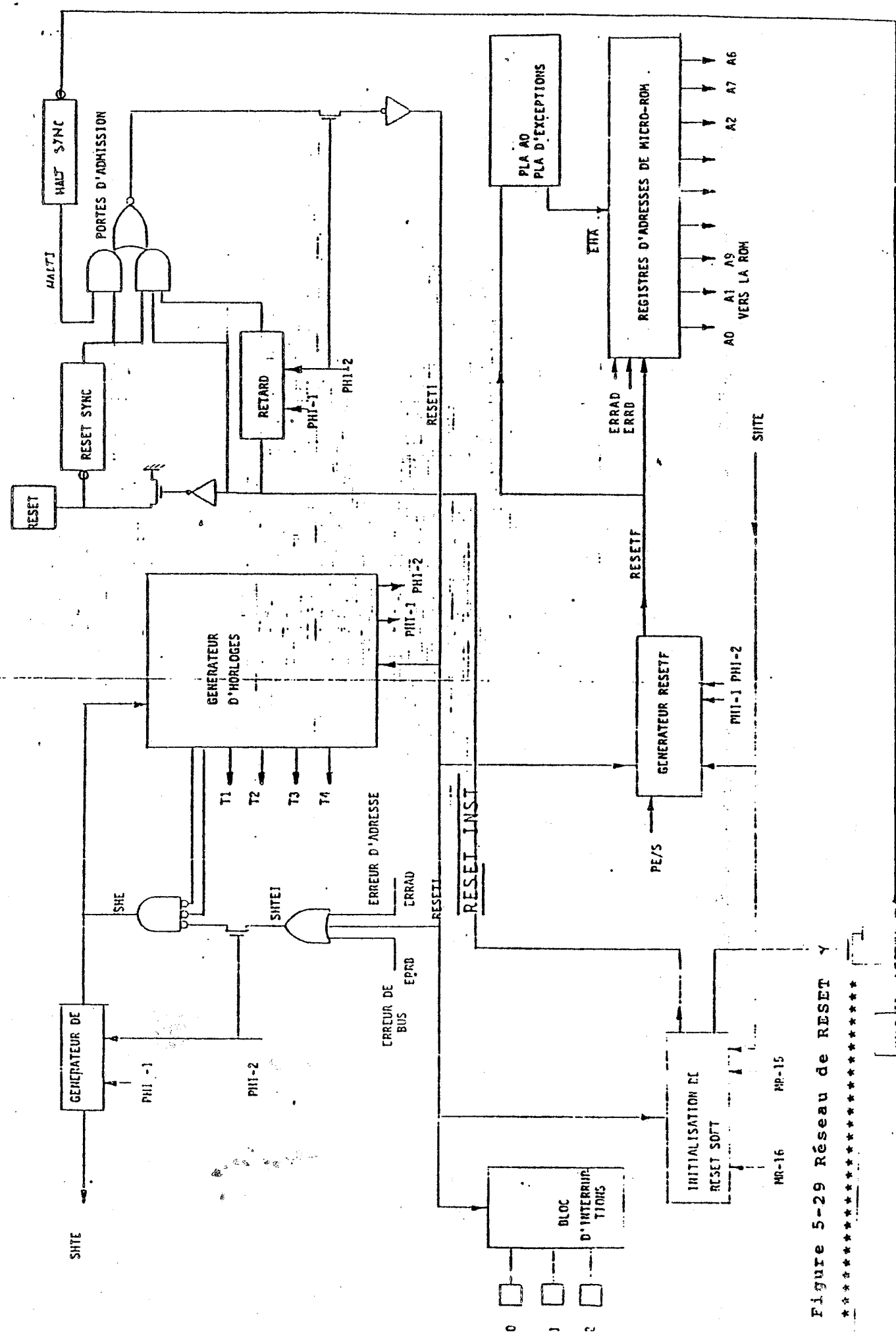


Figure 5-29 Réseau de RESET

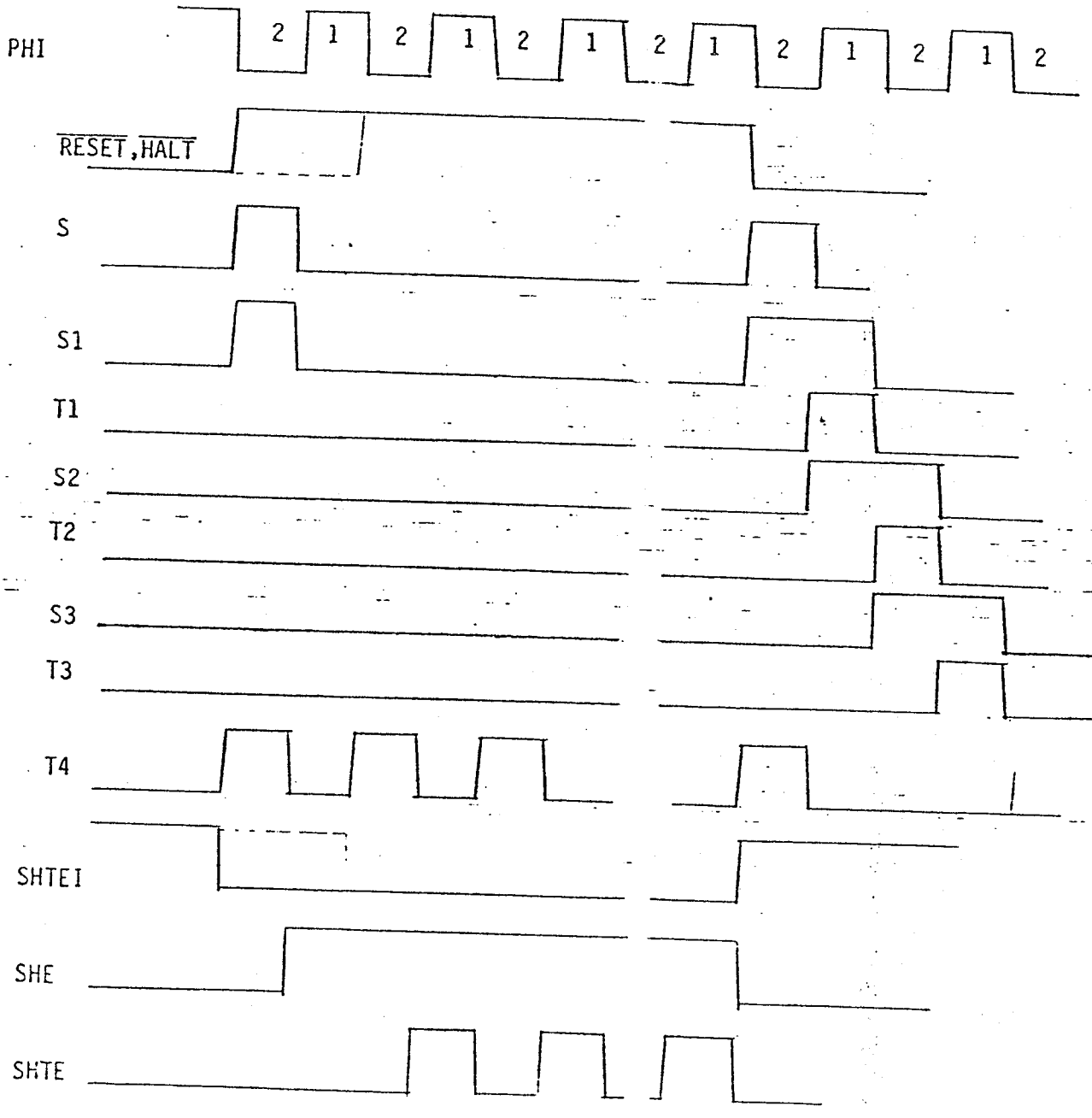


Figure 5-30- Modification des instants T1->T4 par Reset

****

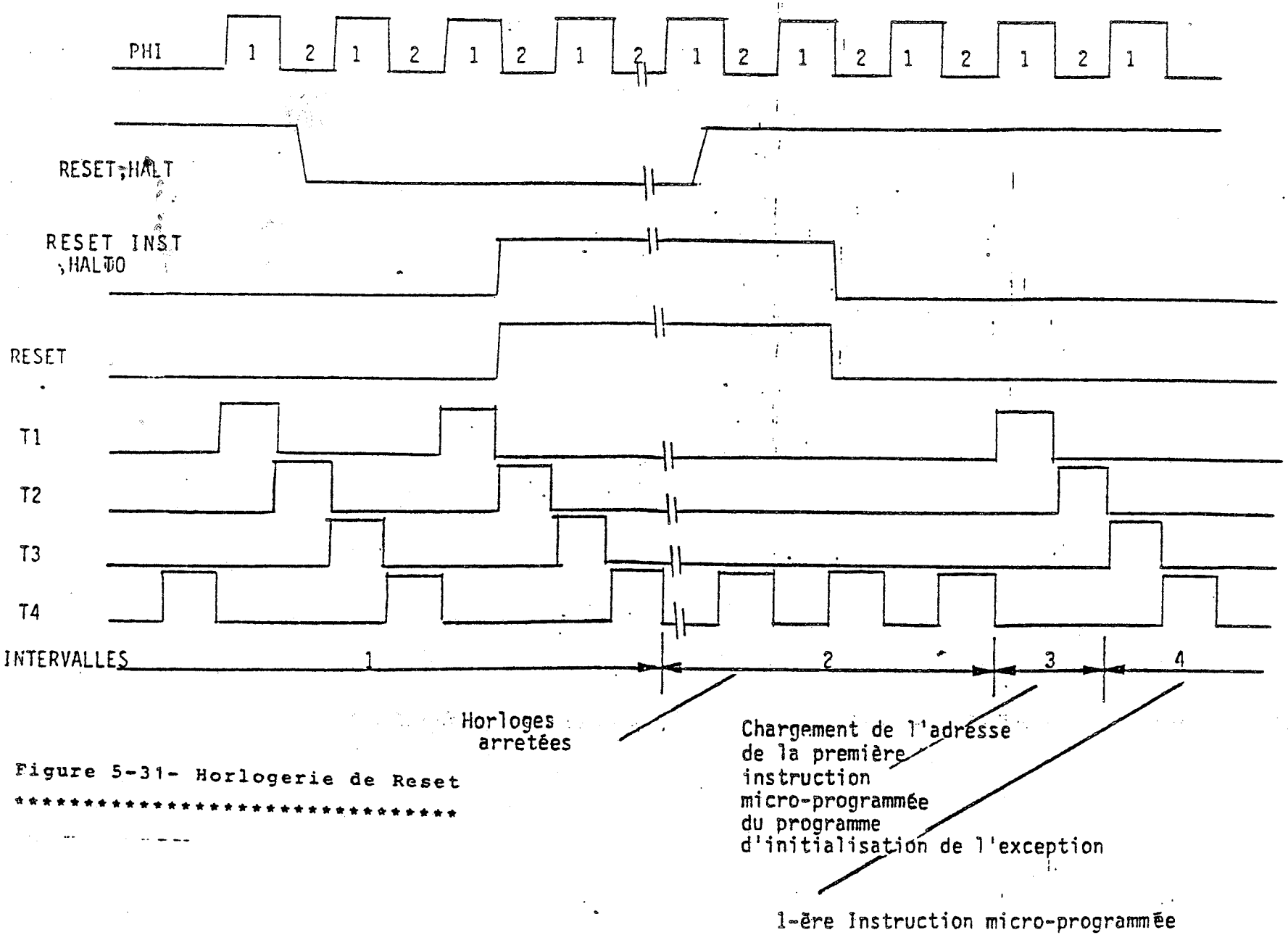
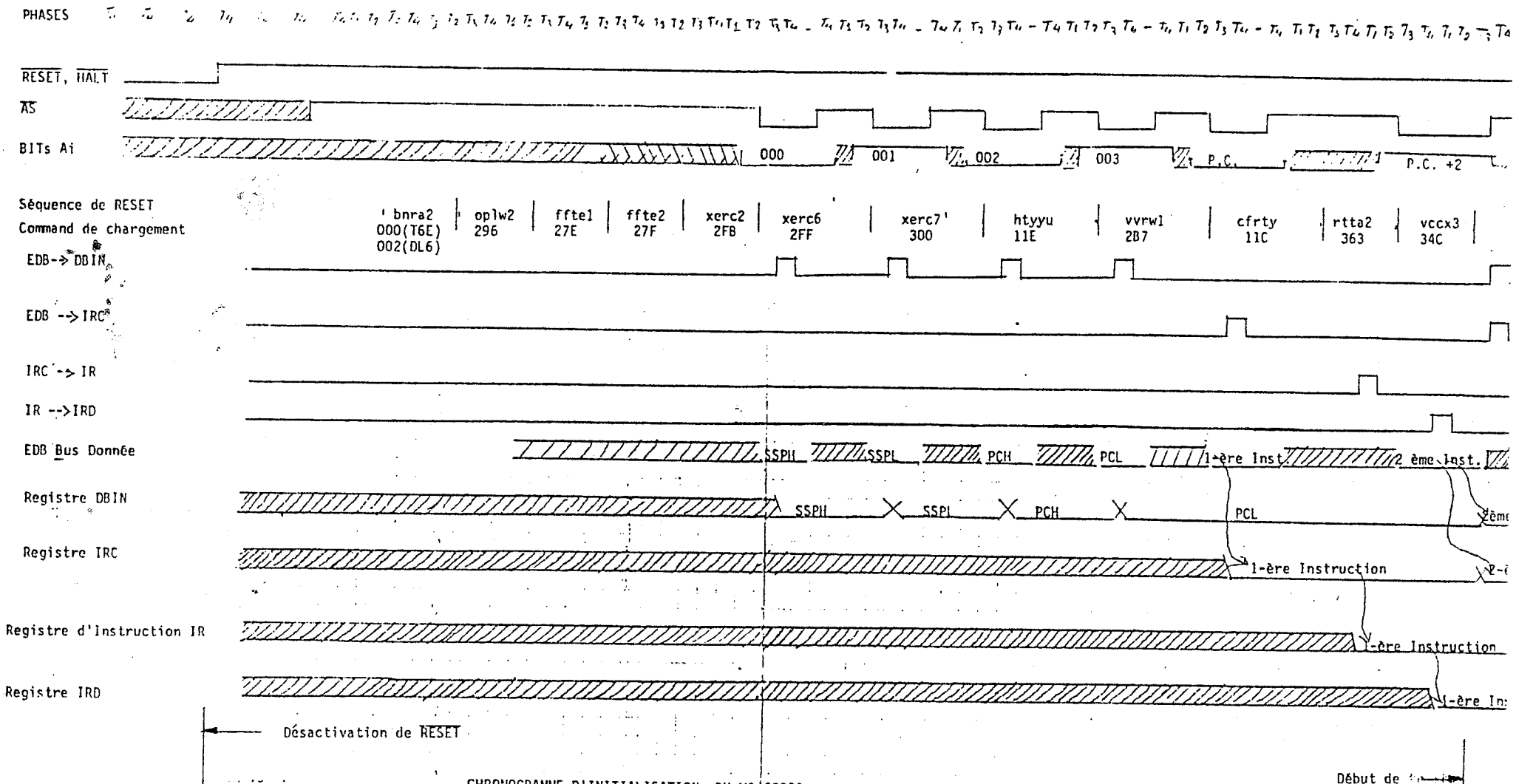


Figure 5-31- Horlogerie de Reset

****



CHRONOGRAMME D'INITIALISATION DU MC 68000
Figure 5-32

4-2-2-Modification de la séquence de l'adressage des exceptions (Groupe 0)

La version DL6 du microprocesseur MC68000, comprend un certain nombre de modifications, vis à vis de l'implantation des transistors du PLAF. Les adresses de départs pour les séquences d'exceptions sont modifiées de la façon suivante:

L'adresse forçée (Hexa)	Séquence d'exception
000	Halt
002	RESET
003	Erreurs de bus et adresses

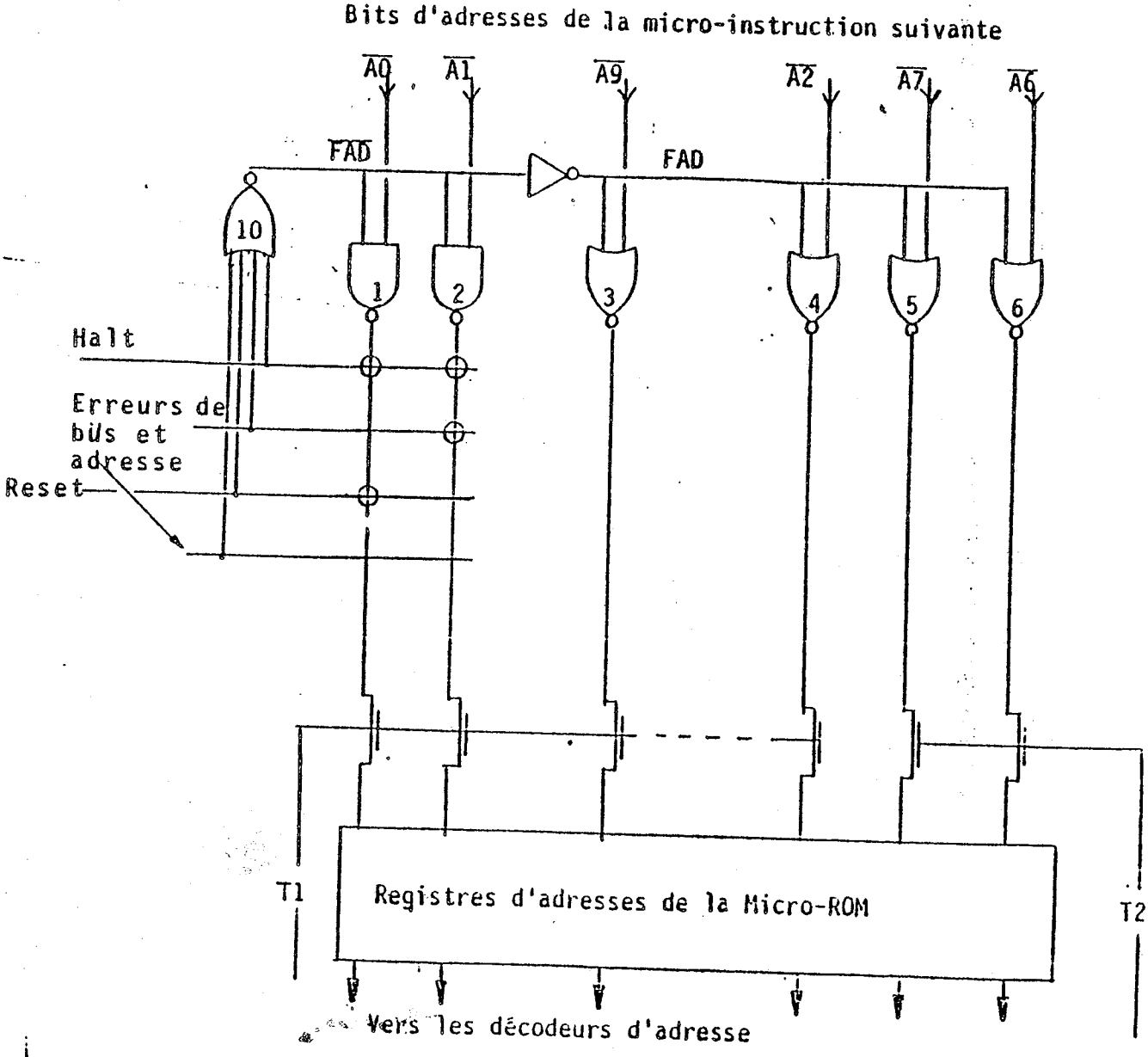


Figure 5-33 Modification de l'adressage des exceptions (groupe 0)

****

Ainsi, les adresses de HALT (1) et de l'erreur de bus ou d'adresse (3) ne sont pas modifiées. Par contre, l'adresse de démarrage de la séquence d'initialisation RESET est modifiée: de 0 elle passe à 2.

De plus, en zéro on trouve une séquence de deux microinstructions.

La première microinstruction est la copie de HALT, elle ne fait donc rien, la seconde est la copie de la première microinstruction du traitement de TRACE (PC-2 → AU) puis cette microinstruction adresse le PLA A1 pour la suite du traitement (nouvelle instruction).

Ainsi, en cas d'activation simultanée des lignes HALT et RESET et/ou erreur de bus et d'adresse (cf. figure 5.33), au lieu de se bloquer dans l'un des états, on exécute cette séquence qui va prendre en compte la valeur placée dans le registre IR.

On peut noter que ce comportement reflète ce qu'il peut se passer au démarrage (mise sous tension), en effet, les lignes HALT RESET, Erreur de bus et d'adresse de la figure 5.33 sont commandées par des points mémoire dont on ne peut garantir la mise à zéro au démarrage du CPU. Ainsi, à la mise sous tension, on va exécuter une instruction quelconque (IR) plutôt que de bloquer sur des cas non prévus (simultanéité RESET + HALT + Erreur de bus et d'adresse).

Il est nécessaire de distinguer deux cas liés à l'arrêt du processeur:

1/ L'état HALT (arrêt permanent) dérivé de la reconnaissance d'une double erreur du bus: c'est-à-dire détection de l'erreur de bus durant l'exécution d'une exception du groupe 0 (RESET, Erreur du bus, Erreur d'adresse). Cet état HALT force le microprocesseur dans un état piège: microinstruction HALT qui ne fait rien et boucle sur elle-même (dans cet état, les phases T1, T2, T3, T4 continuent à se dérouler, on décode toujours la même microinstruction).

2/ La mise en arrêt temporaire du MC 68000, par l'activation du plot HALT. Dans ce cas, l'activation du plot va bloquer l'automate de synchronisation du bus qui, dès qu'il va être sollicité, répondra qu'un périphérique externe occupe le bus et qu'il faut attendre jusqu'à sa libération. La phase d'attente est obtenue par blocage des horloges et génération de la séquence T4 TO T4 TO T4...

On peut remarquer que l'instruction STOP arrête le processeur d'une troisième façon:

En effet, cette instruction, après avoir chargé la nouvelle valeur dans le registre d'état, va boucler sur elle-même. Elle ne bloque pas les horloges, elle empêche seulement le chargement du registre IR. Ainsi, la dernière microinstruction adressant le PLA A1, valide à nouveau le décodage de l'instruction STOP qui se trouve toujours dans IR, on recommence ainsi l'exécution jusqu'à ce qu'un RESET et/ou un HALT externe soit appliqué au processeur.

4-3-Etude de l'infuence du signal reset sur le fonctionnement

Le signal RESETI est issu du signal reset et tant que le plot est mis à zero, ce signal est généré. Les blocs de génération des horloges secondaires sont bloqués par ce signal. Par conséquent le séquencement est arrêté. . Le séquencement du MC68000, est basé sur les instants T1-->T4. Le chargement de la micro-instruction suivante dans la Rom est arrêté, par suite du bloquage des instants Tn. Le cycle mineur de processeur comprend le traitement de la micro-instruction en cours, celui-ci est terminé avant la prise en compte de l'initialisation. Alors le signal reset garde le processeur en état bloqué. Les plots du type 3-états sont mis à haute impédance, assurant ce bloquage. Figures 5-30 et 5-31, présentent le modification d'horloges. Les instants T1,T2 et T3 ne sont plus générés par l'activation de reset..La montée du signal reset (désactivation de la broche reset), permet au processeur d'exécuter la séquence d'initialiation (voir figure 5-32)

4-4-Etude du fonctionnement lors de l'application d'une interruption non masquable lors d'un reset

Le reset ayant la priorité la plus élevée, peut masquer les autres traitements en cours, y compris les interruptions. La logique est basée sur l'autonomie de fonctionnement de reset assurée par son circuit et l'arrêt des horloges. Le PLA d'exceptions PLA-A0, reçoit le signal RESETF pour masquer les autres exceptions. Ainsi la demande de l'exception d'interruptions "DINP'" est masquée par le PLA-A0. L'adressage direct de première micro-instruction de la procédure de traitement du reset est prévue par le PLAF comme la montre la figure 5-9.

4-5-Etude du fonctionnement lors de l'application d'une interruption non-masquable à la mise sous tension.

L'établissement de Vcc met en fonction les circuit du MC68000. Aprés ce temp d'établissement, l'activation des plots d'interruptions, génère le signal DINP', dirigé vers le PLA-A0. Ce PLA initialise l'adresse de micro-programme d'interruptions (Voir la partie consacrée à la logique d'exceptions et la structure du PLA-A0).

La prise en compte du PLA-A0 dépend du bit I de micro-instruction. Ce PLA est activé à la fin du traitement d'une macro-instruction (codes op.'s). Alors à la mise sous tension la micro-rom va générer une séquence quelconque de micro-instructions. Cette séquence peut varier de quelques micro-instructions jusqu'à la séquence la plus longue de 88 micro-instructions. Le temps le plus long avant de considerer la commande DINP' est donc 176 cycles PHI-1 et PHI-2.

En cas d'exeption de l'état Halt, la demande d'interruption n'est pas prise en compte.

Les bits du masque d'interruption contenus dans le registre d'état peut être mis à 1 empêchant ainsi la prise en compte du signal DINP'. On peut remarquer qu'une interruption d'un niveau quelconque peut être démarrée pourvu que le niveau du masque à la mise sous tension ait un niveau plus faible.

4-6-Etude du fonctionnement interne du microprocesseur lorsqu'il reçoit un code opération invalide

Le chapitre V-3, détaille le fonctionnement du MC68000, lorsqu'il reçoit un code invalide. Le PLA de décodage des codes invalides, active la ligne ILLEG'. Cette commande provoque le micro-programme de traitement du code invalide venant d'être détécté (invalide ou non-implémenté). Le PLA-A1 n'est plus considéré, comme source pour fournir l'adresse de départ pour le traitement suivant.

4-7-Etude de la dépendance de l'exécution d'une instruction en fonction de la précédente

L'exécution des instructions est synchronisée par la prise en compte du PLA A1, lui-même activé à la fin de chaque routine de traitement. Les codes valides, sont pris en compte, l'un après l'autre. Le traitement du code courant commence dés que le traitement de la précédente s'achève. On peut remarquer que le MC68000 fonctionne comme une machine pipe-line. En effet, dès qu'il le peut if effecte une ou deux lectures d'avance afin de changer ses registres tampons: le registre DBIN dans la partie opérative et le registre IRC dans la partie contrôle. Il dispose ainsi, selon le cas, d'un mot opération d'avance et/ou d'un opérande (ou extension d'adresse) avant d'en faire la demande il peut donc stocker le prochain mot opération 80 cycles PHI-1 PHI-2 avant l'utilisation de celui-ci. Mais seule l'avant-dernière micro-instruction de l'opération courante validera le chargement IRC->IR et permettra ainsi le décodage du nouveau code opération. Donc il n'y a pas de dépendance entre l'execution des instructions. Tous les codes valides et invalides du MC68000 sont décodés par les PLA de décodage. Le PLA-A1, prend en compte les codes operations, et le PLA de décodage des codes invalides, détecte les codes invalides. Le traitement des codes invalides est considéré comme plus prioritaire à celui des code opérations.

4-8-Traitement d'un code invalide après la fin du traitement d'un code correct.

La fin du traitement d'un code opération est déterminée par la micro-instruction qui est chargée de la prise en compte du code opération suivant. Le premier bit de micro-instruction est positioné pour valider le PLA-A1. Ce PLA, fourni l'adresse de la première micro-instruction qui débute la procédure de traitement de ce code. En cas de détection d'un code invalide le signal ILLEG' est activé. L'activation est basée sur le décodage d'un PLA qui détecte les codes invalides. Le ILLEG' est envoyé au PLA d'exception PLA-A0. Dans un cas où aucune exception plus prioritaire que ILLEG' n'est détectée le processeur exécute la routine de traitement des codes invalides. L'adresse du micro-programme désignée par PLA-A1 n'est plus considérée. C'est le PLA-A0 qui fournit l'adresse de départ pour la routine de traitement des codes invalides.

L'explication détaillée de la logique d'exception PLA-A0 est apportée au paragraphe I-16. Le paragraphe III détaille le traitement des codes invalides.

V - CONCLUSION SUR LES ANOMALIES DE FONCTIONNEMENT DU MC 68000

1. Définitions

On appelle RESET (hard): la génération du signal RESET depuis l'exétrieur, par activation externe de la broche RESET.

On appelle RESET (soft): la génération du signal RESET depuis l'intérieur, par activation interne de la broche RESET ; ceci correspond à l'exécution de l'instruction RESET.

On appelle HALT (hard): la génération du signal HALT depuis l'extérieur par l'activation externe de la broche HALT.

On appelle HALT (soft): la génération du signal HALT depuis l'intérieur, par activation interne de la broche HALT ; ceci correspond à l'exécution de la microinstruction du HALT (version T6E) ou des deux microinstructions de HALT (version EL6).

2. Rappels

On rappelle que:

Le RESET(soft) a pour but de réinitialiser les périphériques du MC 68000, en laissant le microprocesseur dans l'état où il se trouve (exécution d'instruction).

Le HALT (soft) a pour but de bloquer le microprocesseur lorsqu'une erreur de bus ou d'adresse survient lors du traitement des exceptions: erreur de bus, erreur d'adresse et RESET. Le blocage est réalisé par bouclage interne sur une microinstruction ne faisant rien.

Le RESET(hard) a pour but de réinitialiser le microprocesseur par exécution de la séquence d'initialisation.

Le HALT (hard) permet de bloquer momentanément le microprocesseur qui stoppe toute exécution d'instruction à la fin du cycle de bus en cours et ce jusqu'à ce que la broche HALT soit désactivée ; il continue alors son fonctionnement normal.

D'après les brochures MOTOROLA :

Le RESET(hard) permet une "réinitialisation du microprocesseur" tandis que l'activation simultanée RESET (hard) et HALT (hard) permet une "réinitialisation complète du système" (CPU et périphérique).

3. Double conflit entre RESET (hard) et RESET (soft)

Les deux signaux RESET: RESET(soft) et RESET (hard), provoquent deux conflits. Le premier conflit se produit entre ces deux signaux eux-mêmes: l'un masquant l'autre. Le second conflit se produit lors de l'excitation de la broche HALT.

3.1. Conflit des RESET:

Considérons le microprocesseur en exécution d'instructions, l'instruction en cours étant une instruction RESET(soft) destinée à réinitialiser la périphérie. Cette instruction dure 124 cycles Ø, Ø2 soit 15,5 microsecondes (pour un MC 68000 - 8 Mhz). Durant tout ce temps, l'excitation du bouton RESET (hard) en vue de la réinitialisation du microprocesseur n'aura aucune influence. En effet, au lieu d'entamer la séquence d'initialisation, le proceseur continuera l'exécution de l'instruction RESET (soft), puis passera à l'instruction suivante...

Dans ce cas, le seul moyen de redémarrer le système est l'excitation simultanée de HALT et RESET.

Donc:
Durant l'exécution de l'instruction RESET (soft) si l'on excite la broche RESET (hard) : Au lieu de lancer la routine de réinitialisation, qui est une routine d'exception du groupe 0, donc plus prioritaire qu'aucune instruction, on continue l'exécution de l'instruction RESET (soft) puis on enchaîne à la suivante.

Remarque :

Le comportement précédent reste le même, si, au lieu d'une seule instruction RESET (soft), on a une séquence d'instructions RESET (soft). En effet, pendant l'exécution de toute cette séquence, et quelle que soit sa longueur, il sera impossible de réinitialiser le microprocesseur par l'intermédiaire de la broche RESET (hard).

3.2. Conflit des RESETavec HALT:

Considérons les mêmes conditions initiales que précédemment: le microprocesseur est en exécution d'instruction, l'instruction en cours étant une instruction RESET (soft). Si, durant l'exécution de cette instruction, le microprocesseur reçoit un signal HALT (hard) en provenance de l'extérieur, alors, il y a anomalie de fonctionnement. En effet, au lieu de s'arrêter (HALT) et d'attendre la fin du blocage externe par HALT (hard), le microprocesseur entame la séquence de réinitialisation du système avec perte du contexte courant.

Deux sources sont possibles pour la génération du signal HALT (hard):

- un périphérique possédant une connexion sur la ligne RESET externe ; dans ce cas, il faut que ce périphérique ait lancé le HALT avant d'enregistrer le RESET en provenance du microprocesseur.
- un périphérique ne possédant pas de connexion sur la ligne RESET externe, mais générant des HALT pour utiliser les bus externes du MC 68000 (DMA par exemple). Dans ce cas, ce périphérique peut générer le signal HALT à un instant quelconque durant toute l'exécution de l'instruction RESET (soft).

Donc:

durant l'exécution de RESET(soft) l'arrivée d'un HALT (hard) va provoquer une réinitialisation du système, avec perte du contexte courant,

au lieu de bloquer momentanément le microprocesseur, puis de continuer l'exécution des instructions suivantes.

4. Remarque

On peut remarquer que la modification de l'adresse de démarrage du RESET: adresse 0 pour la version T6E, adresse 2 pour la version DL6, n'a aucune influence sur ces anomalies du comportement du MC 68000 vis-à-vis des brochures MOTOROLA.

Ainsi donc, ces anomalies sont-elles valables pour les deux versions successives T6E et DL6.

AUTORISATION de SOUTENANCE

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

VU les rapports de présentation de Messieurs

. F. ANCEAU, Professeur
. J.P MOREAU, Ingénieur

Monsieur DERANTONIAN Henry

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Informatique".

Fait à Grenoble, le 2 juillet 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble
P.O. le Vice-Président,