Une méthode de conception de microprocesseurs CMOS: application au 8048 (Intel)

Mohammed Djameleddine Sahbatou

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR-INGENIEUR
«INFORMATIQUE»

par

SAHBATOU Mohammed Djameleddine

UNE MÉTHODE DE CONCEPTION DE MICROPROCESSEURS CMOS: APPLICATION AU 8048 (INTEL)

Thèse soutenue le 12 Novembre 1984 devant la commission d'examen.

F. ANCEAU	Président
A. GUYOT G. NOGUEZ J.P. POTET J.P. SCHOELLKOPF	Examinateurs

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

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur : **Monsieur M. MERMET**
Directeur des Etudes et de la formation : **Monsieur J. LEVASSEUR**
Directeur des recherches : **Monsieur J. LEVY**
Secrétaire Général : **Mademoiselle M. CLERGUE**

Professeurs de 1ère Catégorie

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

Professeurs de 2ème catégorie

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

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

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

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

A ma tante Aicha
et à mes parents
à qui je dois tout.

REMERCIEMENTS

Que monsieur François ANCEAU, professeur détaché ENSIMAG/INPG, trouve ici l'expression de ma vive reconnaissance pour m'avoir accordé sa confiance et fait partagé sa passion pour l'architecture des ordinateurs; je le remercie également de m'avoir fait l'honneur de présider le jury de cette thèse.

Mes remerciements vont aussi à M. Jean Pierre POTET, responsable de la définition de nouveaux produits chez MATRA-HARRIS SEMICONDUCTEURS (Nantes) qui a suggéré cette étude et avec qui j'ai eu, à maintes reprises, des discussions fructueuses; à M. Gérard NOGUEZ, professeur à l'université Pierre et Marie CURIE (Paris VI), qui a bien voulu examiner ce travail et participer au jury; à M. Jean Pierre SCHOELLKOPF, ingénieur à BULL Systèmes (Les Clayes Sous Bois), qui m'a apporté son aide précieuse tout au long de ce travail; à M. Alain GUYOT, maître-assistant ENSIMAG/INPG, qui a manifesté son intérêt pour cette étude et dont ses remarques ont permis d'améliorer ce document.

Je remercie également tous les membres de l'Equipe de recherches en architecture des ordinateurs (IMAG TIM3) et le personnel du service de reprographie pour leur sympathie.

Mes remerciements vont enfin au Ministère algérien de l'Enseignement qui a rendu possible mon séjour en France.

- RESUME -

Cette thèse présente une méthode de conception du micro-ordinateur d'Intel 8048 en technologie CMOS. En appliquant une démarche descendante, notre travail consistait à étudier les spécifications du manuel d'utilisateur pour aboutir à la réalisation du circuit.

Chaque instruction a été décomposée en un algorithme d'interprétation, en se basant sur une structure à deux bus et une horloge à deux phases; l'objectif étant d'aboutir à une architecture régulière de la partie opérative ("bit-slice").

Les schémas logiques sont réalisés en circuiterie CMOS statique. Une méthode d'implantation est proposée. Elle consiste à travailler sur une structure ordonnée (matricielle) formée de lignes de polysilicium et de colonnes d'aluminium. Cela permet l'introduction d'une représentation symbolique pour le dessin des cellules. Cependant, cette stratégie impose d'avoir les bus de données de la partie opérative en polysilicium qui donnent toutefois un temps de propagation acceptable.

L'architecture de la partie contrôle est bâtie sur une structure à deux PLA avec un générateur de temps. Les contenus de ces PLA sont obtenus par compilation de l'algorithme d'interprétation, sous une forme exploitable par le système PAOLA. PAOLA permet de générer le dessin des PLA avec une optimisation topologique de la surface.

- MOTS-CLEF -

Algorithme d'interprétation- Partie opérative- CMOS statique- Méthode d'implantation- Assemblage de cellules- Compilation de l'algorithme de contrôle- PLA et générateur de temps-

- TABLE DES MATIERES -

TABLE DES MATIERES

- CHAPITRE 2 : ARCHITECTURE DE LA PARTIE OPERATIVE -

TABLE DES MATIERES

- CHAPITRE 3 : REALISATION DE LA PARTIE OPERATIVE -

TABLE DES MATIERES

TABLE DES MATIERES

- INTRODUCTION -

La conception d'un microprocesseur doit satisfaire des critères de coût, de performance et de fiabilité. Les différents compromis possibles ont donné naissance à des écoles de styles différents. Au début de la venue des microprocesseurs sur le marché, on distinguait deux méthodes:

1- La méthode progressive faisant évoluer une réalisation sûre et déjà existante. Elle était utilisée presque exclusivement par les industriels.
Cette stratégie de conception appliquée au départ, n'était pas faite dans un esprit futuriste pour gérer plusieurs milliers de transistors dans une puce comme actuellement. Elle a été alors délaissée au profit d'une méthode heuristique.

2- La méthode heuristique choisissant le style et les paramètres de la réalisation. Ces choix heuristiques sont fondés sur l'expérience acquise au cours des évaluations effectuées sur des réalisations existantes. Cette méthode donne lieu à deux démarches opposées, le but étant toutefois le même: REPONDRE AUX SPECIFICATIONS INITIALES.

a) Démarche ascendante

Elle regroupe des organes élémentaires existants (moyens) pour constituer des opérations de plus en plus complexes permettant d'aboutir aux spécifications initiales.

MOYENS ---> SPECIFICATIONS

b) Démarche descendante

Elle permet en analysant les spécifications initiales (besoins de l'utilisateur) d'aboutir à une architecture adaptée aux moyens dont dispose le concepteur.

SPECIFICATIONS ---> MOYENS

Malgrè l'adoption de cette dernière démarche par la plupart des concepteurs, on observe encore deux tendances dans le dessin topologique des circuits, dues aux contraintes de la surface et

du coût de la conception.

La première tendance consiste à tasser les cellules de base et à compacter les blocs pour gagner de la surface. Ce gain n'est parfois qu'apparent puisqu'on risque de céder en contrepartie une surface plus importante pour les interconnexions.

La deuxième tendance consiste à rechercher une régularité dans le dessin afin de réduire le coût de la conception au détriment de la surface.

Cette approche structurée a fait ses preuves dernièrement dans la conception de récents microprocesseurs tels que: MC 68.000 (MOTOROLA) et I 80C31 (INTEL). En effet, une conception est jugée bonne si son coût est moindre.

Notre travail qui consiste à concevoir le micro-ordinateur 80C48 a été mené dans cet esprit de conception régulière en se basant sur la méthodologie de conception descendante CAPRI (ref. 1). Il a pour but de tester l'efficacité des outils de CAO développés au laboratoire tels que LUBRICK (ref. 2), MACSIM (ref. 3) et PAOLA (ref. 4), et de prouver par la même occasion que la stratégie utilisée est un bon compromis entre la surface du circuit et le coût de la conception.

Ainsi, cette étude comporte quatre chapitres:

- Le premier chapitre résume les spécifications du micro-ordinateur 80C48.
- Le second chapitre concerne la définition de l'architecture de la partie opérative. Nous essayerons d'expliquer comment se fait la décomposition des instructions en algorithmes d'interprétation. Puis nous avons établi un plan de masse de la partie opérative de telle sorte que la surface du circuit soit globalement optimisée, obtenant alors des connexions directes entre différents blocs du micro-ordinateur (ref. 5).
- Le troisième chapitre porte sur la réalisation de la partie opérative. Des schémas logiques ont été élaborés et une méthode d'implantation des circuits en technologie CMOS a été proposée. Les cellules de base ont été assemblées par programmes à l'aide du système LUBRICK.

- Le quatrième chapitre traite de la réalisation des circuits auxiliaires, notamment les ports d'entrées-sorties et le système d'interruptions, et de la conception de la partie contrôle. Cette dernière est bâtie sur une structure multi-PLA avec un générateur d'instants. Elle est générée automatiquement par l'outil MACSIM à partir des algorithmes d'interprétation décrits en langage PASCAL.
Nous présenterons aussi les résultats finaux de l'implantation du circuit.

Enfin, nous terminerons cette étude en montrant ses difficultés et les améliorations qu'on peut apporter à cette démarche dans les recherches ultérieures.

D'autre part, il est nécessaire de préciser que la conception du 8048 a été un travail de groupe. La décomposition des instructions a été d'abord traitée en commun avec F.BERTRAND et une première architecture de la partie opérative a été définie dans le cadre d'un projet de DEA (ref. 9). Ce travail a été continué par F.BERTRAND pour arriver à un algorithme d'interprétation plus souple en recherchant le côté systématique des opérations pour l'ensemble des instructions. Ceci a pour but de faciliter la conception de la partie contrôle. Quelques modifications ont été apportées à cette seconde version et cette étape de travail est expliquée dans le deuxième chapitre de cette thèse.
Cette étude est donc le fruit d'un travail d'équipe qui s'est concrétisé par la réalisation de la partie opérative et la définition de la partie contrôle.
La partie opérative a été dessinée une première fois en représentation symbolique "STICK" avant d'être redessinée par P.OBJOIS et C.MATHERON en règles Lambda. Leur travail consistait à redessiner les cellules de base déjà définies, avec les dimensions réelles des transistors. Les programmes d'assemblage établis initialement pour la version "STICK" ont été adaptés à ces nouvelles cellules. Ceci a permis la génération de la partie

opérative réelle par le système LUBRICK (ref. 2). Des amplificateurs de commandes ainsi que des plots de test ont été ajoutés afin de pouvoir tester la partie opérative. Cette partie du circuit a été fabriquée dans le cadre du MPC 84 en CMOS par la société MHS (MATRA-HARRIS SEMICONDUCTEURS).
Enfin, l'algorithme d'interprétation des instructions a dû être repris et décrit conjointement en langage PASCAL et en langage IRENE (ref. 10) sous forme de commentaires pour déduire les contenus des PLA de la partie contrôle à l'aide de l'outil MACSIM (ref. 3). Cette tâche a été menée par F.BERTRAND et les résultats de la compilation de la description sont résumés dans le quatrième chapitre.

- CHAPITRE 1: SPECIFICATIONS DU 8048 -

1.1- DEFINITION DES DIFFERENTS BLOCS FONCTIONNELS DU MICRO-ORDINATEUR.

1.1.1- Bloc arithmétique et logique
1.1.2- Registre d'état PSW
1.1.3- Mémoire RAM
1.1.4- Mémoire ROM
1.1.5- Compteur T
  a) Mode Compteur d'évènements externes
  b) Mode Compteur d'instants
1.1.6- Système d'interruptions
1.1.7- Système d'entrées-sorties (I/O)

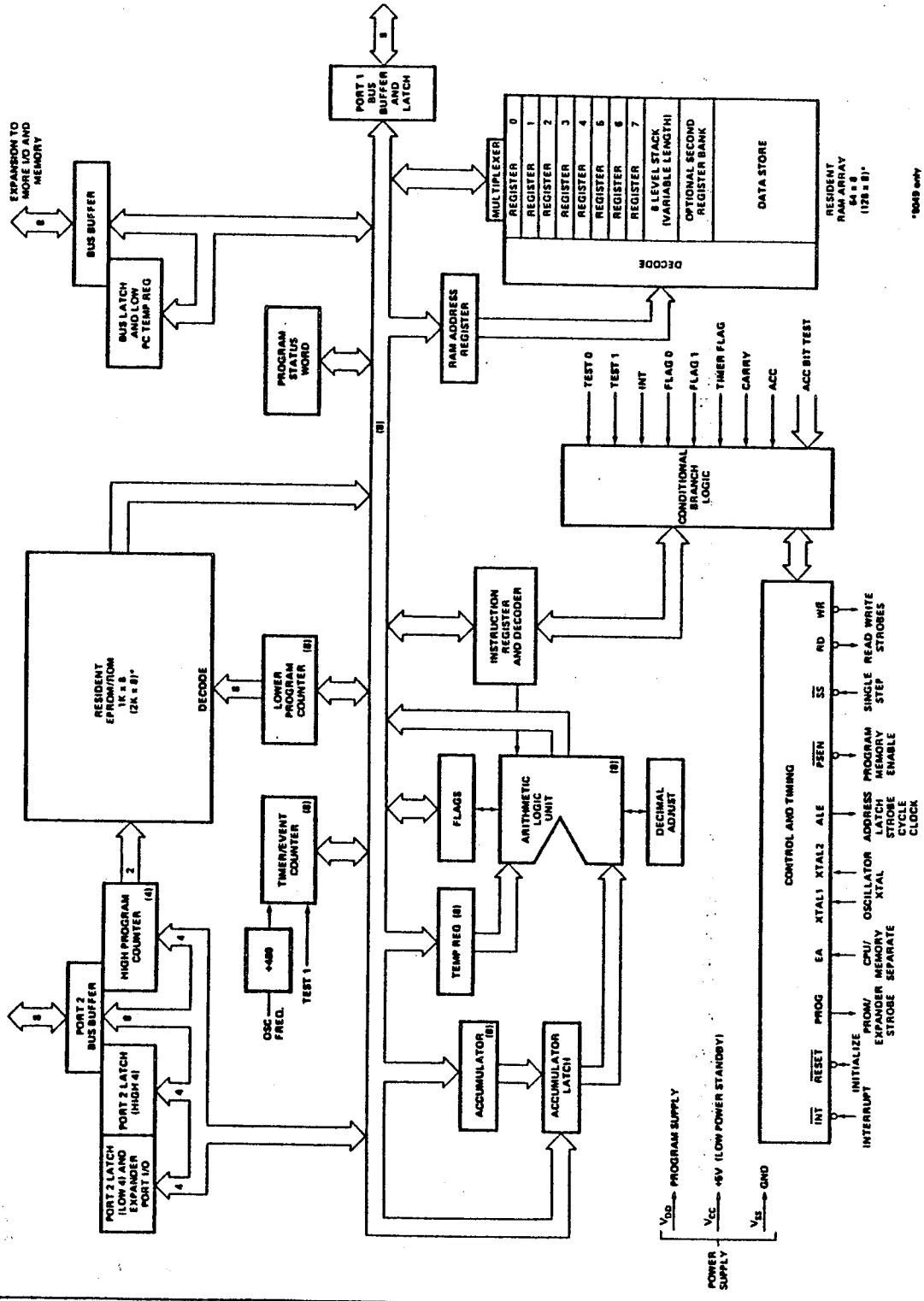
1.2- EXTENSION DU SYSTEME.

1.2.1- Extension de la mémoire ROM
1.2.2- Extension de la mémoire RAM
1.2.3- Extension des entrées-sorties (I/O)

1.3- CHRONOGRAMMES DES SIGNAUX ET HORLOGERIE.

1.3.1- Chronogrammes des signaux
1.3.2- Horlogerie et décomposition d'une instruction

1.4- JEU D'INSTRUCTIONS.



8048/8049 BLOCK DIAGRAM

CHAPITRE 1
- SPECIFICATIONS DU 8048 -

Ce chapitre exposera succinctement les spécifications du 8048 qu'on pourra trouver en détail dans le manuel de l'utilisateur (ref. 6).

1.1- DEFINITION DES DIFFERENTS BLOCS FONCTIONNELS DU MICRO-ORDINATEUR

Le 8048 est un micro-ordinateur de 8 bits. Sa structure interne se compose de:

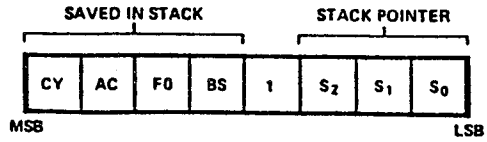
- un bloc arithmétique et logique,
- un registre d'état PSW (8 bits),
- une mémoire vive RAM (64 octets),
- une mémoire morte ROM (1 Kilo-octet),
- un compteur T d'instants et d'évènements externes (8 bits),
- un système d'interruptions,
- un système d'entrées-sorties (I/O).

1.1.1- Bloc arithmétique et logique

L'unité arithmétique et logique, associée à l'accumulateur, réalise les opérations suivantes:

- addition avec ou sans retenue,
- incrémentation et décrémentation,
- opérations logiques: calcul du complément, du "ET", du "OU" et du "OU" exclusif,
- rotation à droite et à gauche avec ou sans retenue,
- permutation entre poids fort et poids faible de l'accumulateur ("SWAP"),
- ajustement décimal de l'accumulateur.

1.1.2- Registre d'état PSW



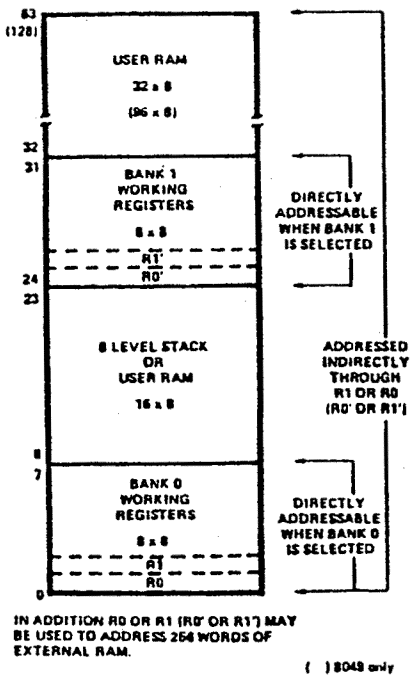
CY CARRY
AC AUXILLARY CARRY
F0 FLAG 0
BS REGISTER BANK SELECT

PROGRAM STATUS WORD (PSW)

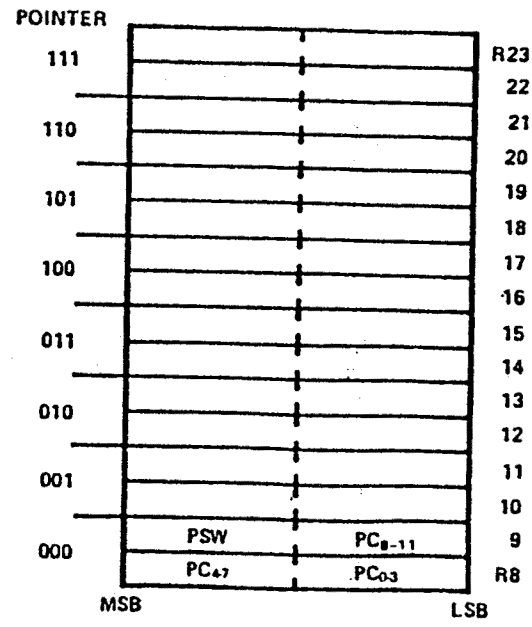
Le PSW est formé d'un ensemble de huit bascules. Il est considéré comme un registre de huit bits, contenant les informations suivantes:

- PSW<2:0> Pointeur de pile de la RAM.
- PSW<3> Bit non utilisé.
  Sa valeur est à "1" quand le PSW est lu.
- PSW<4> Ce bit 4, noté aussi BS, permet de sélectionner la banque des 8 registres de travail de la RAM (voir 1.1.3).
  BS = 0, la banque 0 est sélectionnée.
  BS = 1, la banque 1 est sélectionnée.
- PSW<5> Indicateur F0 d'usage général et peut être programmé par l'utilisateur.
- PSW<6> Retenue auxiliaire, notée AC, résultant de l'addition des 4 bits de poids faible.
- PSW<7> Retenue, notée CY, indiquant le débordement de l'accumulateur.

1.1.3- Mémoire RAM



DATA MEMORY MAP



PROGRAM COUNTER STACK

Elle est constituée de 64 registres de 8 bits et se subdivise en une banque 0 de registres, une pile, une banque 1 et divers registres de travail.

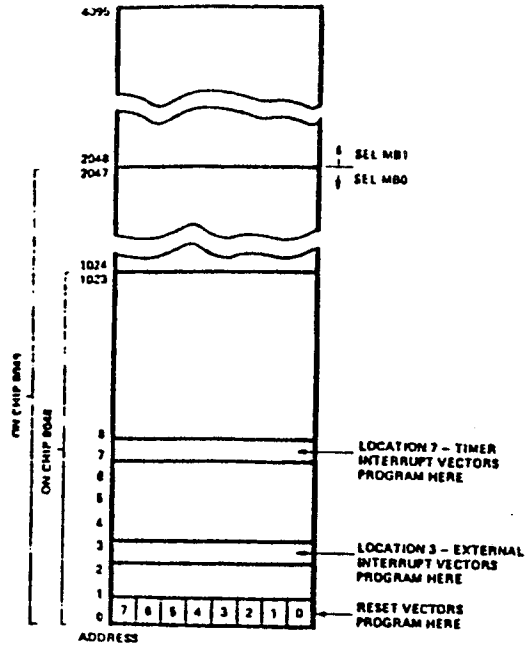
- La banque 0 se compose de 8 registres de travail (position 0 à 7) directement adressables.

- La pile se compose de 8 paires de registres (position 8 à 23). Ces mots mémoires sont adressés par un pointeur de pile. Durant un sous-programme (interruption ou CALL), les contenus du compteur de programme PC et du registre d'état PSW sont sauvegardés dans une paire de registres désignée par le pointeur (voir 1.1.2).

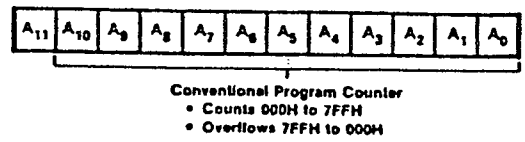
- La banque 1 est constituée de 8 registres de travail (position 24 à 31) directement adressables. La sélection d'une banque se fait par programme.

- Le reste de la RAM, formé de 32 registres (position 32 à 63) est adressable indirectement par les deux registres R0 ou R1, situés à l'adresse 0 ou 1 de la banque sélectionnée.

1.1.4- Mémoire ROM



MCS-48™ PROGRAM MEMORY MAP



PROGRAM COUNTER

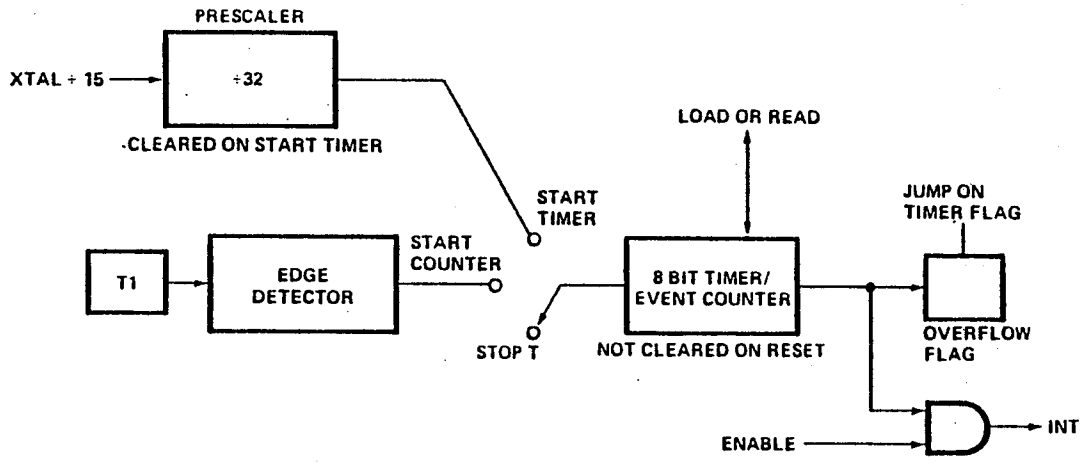
La mémoire morte contient 1 Kilo-octet. Elle est adressée par un compteur de programme (PC) de 12 bits. Le bit le plus significatif (bit 11) permet à l'utilisateur de sélectionner entre la ROM interne et la mémoire morte externe (voir 1.2.1).

Cependant, trois positions importantes sont à signaler dans cette mémoire interne:

- position 0 : elle contient la première instruction d'un programme à exécuter.
- position 3 : elle contient la première instruction du sous-programme d'interruption externe (voir 1.1.6).
- position 7 : elle contient la première instruction du sous-programme d'interruption interne due au compteur T (voir 1.1.6).

1.1.5- Compteur T

C'est un registre de 8 bits qui peut travailler soit en compteur d'évènements externes, soit en temporisateur.



TIMER/EVENT COUNTER

a) Mode Compteur d'évènements externes

L'entrée du registre est connectée à la broche T1. Chaque transition du niveau "1" au niveau "0" de la broche T1 incrémente le compteur T. Ces incrémentations s'effectuent, au minimum, dans un intervalle de trois cycles.

b) Mode Compteur d'instants

L'entrée du registre est connectée à une horloge interne divisée par 32. Le compteur est ainsi incrémenté tous les 32 cycles machine.

Indépendamment du mode choisi, un indicateur TF (Timer Flag) est positionné à "1" lors du débordement du compteur (passage de la valeur maximale FFH à zéro).

1.1.6- Système d'interruptions

On a deux types d'interruptions:

- une interruption externe est initialisée par application d'un niveau "0" sur la broche $\overline{INT}$. Celle-ci est échantillonnée par le signal ALE à chaque cycle machine.
  Une fois l'interruption détectée, un saut s'effectue au sous-programme situé à la position 3 de la mémoire ROM interne.

- une interruption interne est initialisée lors du débordement du compteur T. Une fois l'interruption détectée, un saut s'effectue à la position 7 de la ROM interne.

Cependant, il est à noter que le système d'interruptions est à un seul niveau. C'est à dire que dans le cas où les deux interruptions sont détectées en même temps, l'interruption externe sera prioritaire. Pendant l'exécution d'un sous programme d'interruption, toutes les futures demandes d'interruptions seront ignorées. Le retour au programme principal se fait par l'instruction RETR.

1.1.7- Système d'entrées-sorties (I/O)

Il est constitué de 27 lignes dont trois ports d'entrées-sorties de 8 bits (port1, port2 et BUS) et trois lignes de test (T0, T1 et $\overline{INT}$) en entrées.
Ces ports sont bidirectionnels. Cependant, le BUS (DB7 à DB0) peut être exclusivement utilisé soit en entrée ou en sortie selon les signaux de contrôle $\overline{RD}$ ou $\overline{WR}$ générés vers l'extérieur par le processeur. Par contre, les ports 1 et 2 peuvent traiter à la fois les entrées pour certains bits et les sorties pour les autres car la fonction des amplificateurs est programmée par l'utilisateur.
Toute émission de données vers l'extérieur est mémorisée statiquement au niveau du port.

1.2- EXTENSION DU SYSTEME

L'environnement du 8048 permet d'augmenter sa puissance de traitement, grâce aux extensions suivantes:

- la mémoire morte (pouvant aller jusqu'à 4 Kilo-octet),
- la mémoire vive (pouvant aller jusqu'à 320 octets),
- un nombre illimité d'entrées-sorties.

1.2.1- Extension de la mémoire ROM

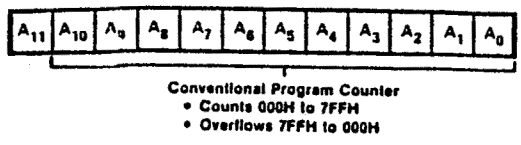
La sélection entre la mémoire morte interne et externe se fait à l'aide du bit le plus significatif du compteur de programme (PC).

PC<11> = 0 Sélection de la mémoire interne si PC<10> = 0 (voir remarque)

PC<11> = 1 Sélection de la mémoire externe.

Ce bit n'est pas altéré par une incrémentation du PC, mais son chargement s'effectue pendant les instructions CALL ou JMP par la lecture de la bascule DBF.

Remarque : Si lors de l'incrémentation du PC, le bit 10 est mis à "1", alors la mémoire morte externe sera automatiquement sélectionnée.



PROGRAM COUNTER

Un accès à la mémoire morte externe nécessite les séquences suivantes:

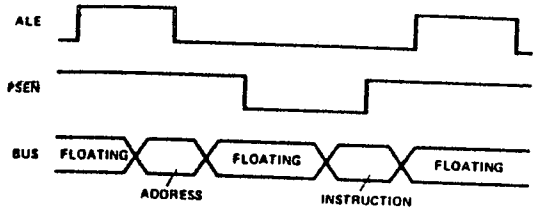
1°) Le contenu du compteur de programme (12 bits) est sorti sur le BUS (8 bits) et sur les quatre bits de poids faible du port2.

2°) Le signal ALE (Address Latch Enable) indique aux périphériques que l'adresse mémoire est disponible. L'adresse est vérouillée à l'extérieur, au front descendant de ALE.

3°) Le signal $\overline{\text{PSEN}}$ (Program Store ENable) active la mémoire

externe.

4°) Le BUS passe en mode entrée pour transférer la donnée au processeur.



INSTRUCTION FETCH FROM EXTERNAL PROGRAM MEMORY

1.2.2- Extension de la mémoire RAM

Les adresses et les données de la RAM externe sont transférées sur les huit lignes du BUS. L'accès à cette mémoire se fait de la façon suivante:

1°) Le contenu d'un des registres (8 bits) R0 ou R1 est sorti sur le BUS.

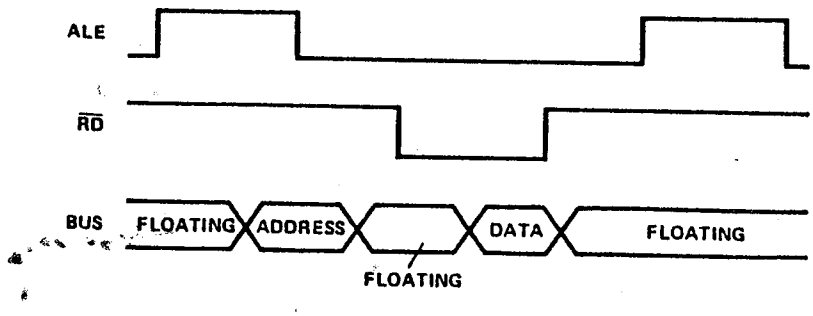
2°) Le signal ALE indique, comme précédemment, que l'adresse est disponible. Elle est mémorisée à l'extérieur, au front descendant de ce signal.

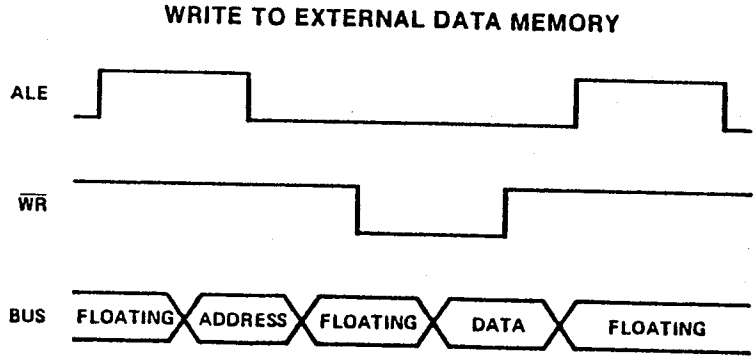
3°) Le signal de lecture $\overline{RD}$ ou d'écriture $\overline{WR}$ sur les broches de sortie correspondantes indique à la RAM externe le type d'accès en cours.

La donnée est disponible au front montant de ces signaux.

4°) La donnée est transférée sur le BUS.

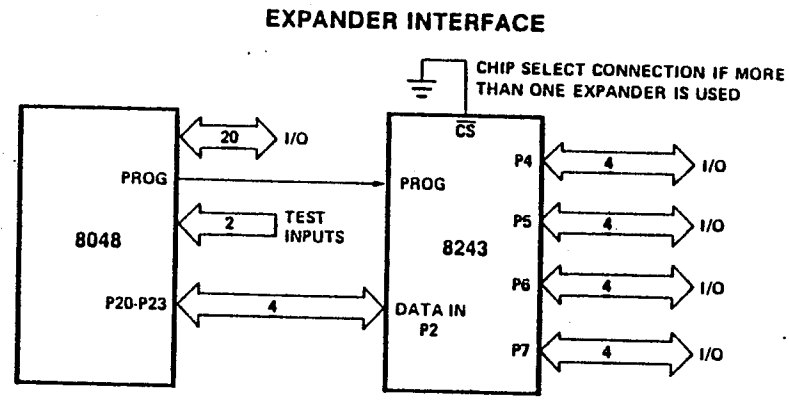
READ FROM EXTERNAL DATA MEMORY

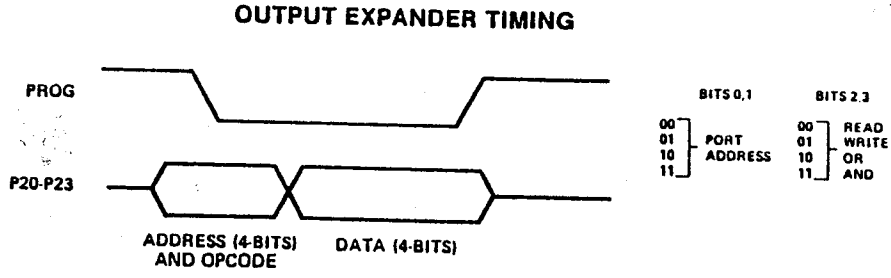




1.2.3- Extension des entrées-sorties (I/O)

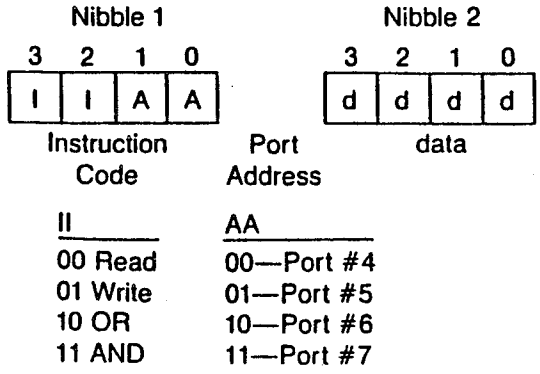
Toute communication entre le 8048 et le 8243 se fait sur les bits de poids faible du port2 (P2L), grâce au séquencement du signal PROG. Le front descendant de ce signal valide l'adresse du port et l'opération à effectuer; tandis que le front montant indique la présence de la donnée (4 bits).





Remarque : L'adresse des ports 4 à 7 est codée sur deux bits

et l'opération à effectuer sur deux autres bits (voir "Nibble 1").

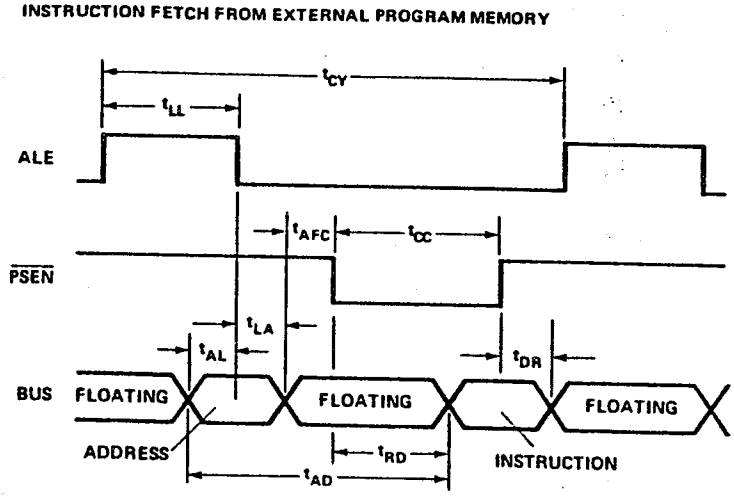


1.3- CHRONOGRAMMES DES SIGNAUX ET HORLOGERIE

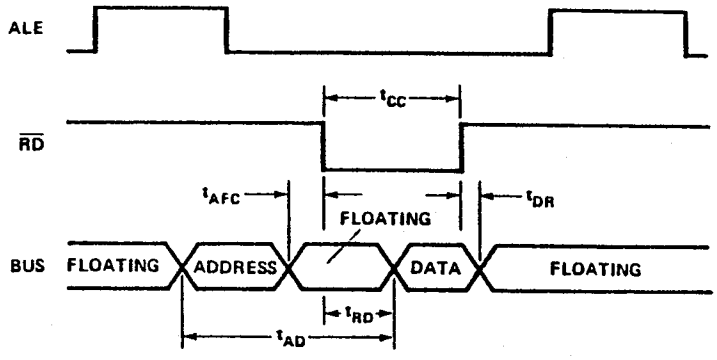
1.3.1- Chronogrammes des signaux

(voir annexe A pour le détail des caractéristiques des signaux)

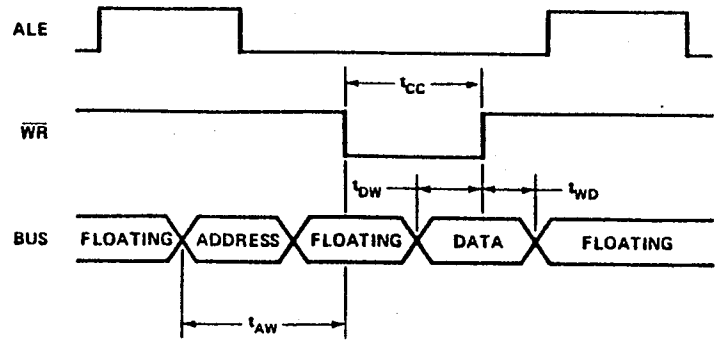
WAVEFORMS



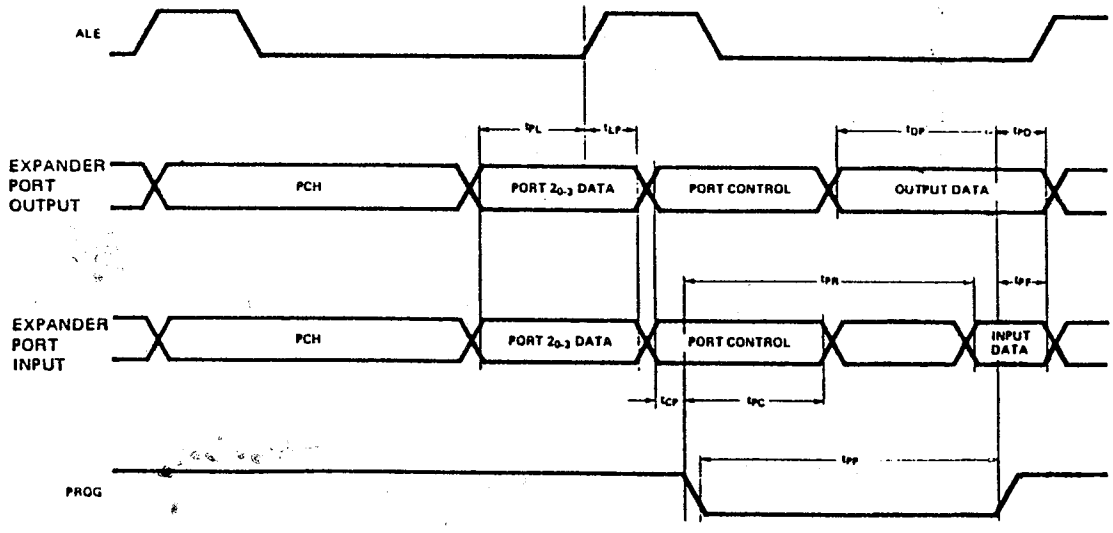
READ FROM EXTERNAL DATA MEMORY



WRITE TO EXTERNAL DATA MEMORY



PORT 2 TIMING

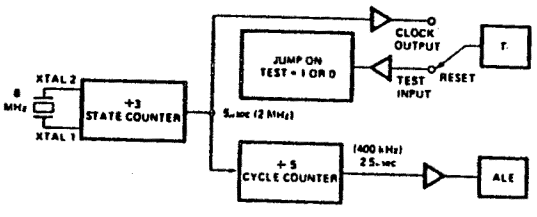


1.3.2- Horlogerie et décomposition d'une instruction

Dans l'horlogerie, on distingue trois blocs fonctionnels:

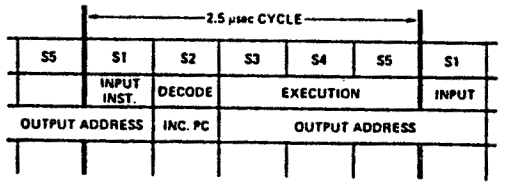
- un oscillateur : c'est un circuit série résonnant à gain élevé avec une fréquence comprise entre 1 et 6 MHz.
- un compteur d'états : la sortie de l'oscillateur est divisée par 3 pour créer une horloge CLK correspondant au cycle machine.
- un compteur de cycles : la division par 5 de l'horloge CLK permet de définir un cycle machine contenant cinq états. Ce cycle est indiqué par le signal ALE.

DIAGRAM OF 8048 CLOCK UTILITIES



Quant aux instructions du 8048, elles peuvent être de un ou de deux cycles machine. Chaque cycle se trouve ainsi décomposé en cinq états, notés S1, S2, S3, S4 et S5, de même durée.

INSTRUCTION CYCLE



1.4- JEU D'INSTRUCTIONS

Les instructions sont au nombre de 96 dont plus de 50% sont

exécutées en un cycle. Celles de deux cycles concernent les opérations d'entrées-sorties, de branchements et les instructions à deux octets.

8048/8049
INSTRUCTION SET SUMMARY

	Mnemonic	Description	Bytes	Cycle
Accumulator	ADD A, R	Add register to A	1	1
	ADD A, @R	Add data memory to A	1	1
	ADD A, #data	Add immediate to A	2	2
	ADDC A, R	Add register with carry	1	1
	ADDC A, @R	Add data memory with carry	1	1
	ADDC A, #data	Add immediate with carry	2	2
	ANL A, R	And register to A	1	1
	ANL A, @R	And data memory to A	1	1
	ANL A, #data	And immediate to A	2	2
	ORL A, R	Or register to A	1	1
	ORL A, @R	Or data memory to A	1	1
	ORL A, #data	Or immediate to A	2	2
	XRL A, R	Exclusive Or register to A	1	1
	XRL A, @R	Exclusive or data memory to A	1	1
	XRL A, #data	Exclusive or immediate to A	2	2
	INC A	Increment A	1	1
	DEC A	Decrement A	1	1
	CLR A	Clear A	1	1
	CPL A	Complement A	1	1
	DA A	Decimal Adjust A	1	1
	SWAP A	Swap nibbles of A	1	1
	RL A	Rotate A left	1	1
	RLC A	Rotate A left through carry	1	1
	RR A	Rotate A right	1	1
	RRC A	Rotate A right through carry	1	1
Input/Output	IN A, P	Input port to A	1	2
	OUTL P, A	Output A to port	1	2
	ANL P, #data	And immediate to port	2	2
	ORL P, #data	Or immediate to port	2	2
	INS A, BUS	Input BUS to A	1	2
	OUTL BUS, A	Output A to BUS	1	2
	ANL BUS, #data	And immediate to BUS	2	2
	ORL BUS, #data	Or immediate to BUS	2	2
	MOVD A, P	Input Expander port to A	1	2
	MOVD P, A	Output A to Expander port	1	2
	ANLD P, A	And A to Expander port	1	2
	ORLD P, A	Or A to Expander port	1	2
Registers	INC R	Increment register	1	1
	INC @R	Increment data memory	1	1
	DEC R	Decrement register	1	1
Branch	JMP addr	Jump unconditional	2	2
	JMPP @A	Jump indirect	1	2
	DJNZ R, addr	Decrement register and skip	2	2
	JC addr	Jump on Carry = 1	2	2
	JNC addr	Jump on Carry = 0	2	2
	J Z addr	Jump on A Zero	2	2
	JNZ addr	Jump on A not Zero	2	2
	JT0 addr	Jump on T0 = 1	2	2
	JNT0 addr	Jump on T0 = 0	2	2
	JT1 addr	Jump on T1 = 1	2	2
	JNT1 addr	Jump on T1 = 0	2	2
	JF0 addr	Jump on F0 = 1	2	2
	JF1 addr	Jump on F1 = 1	2	2
	JTF addr	Jump on timer flag	2	2
	JNI addr	Jump on $\overline{INT}$ = 0	2	2
	JBb addr	Jump on Accumulator Bit	2	2

	Mnemonic	Description	Bytes	Cycles
Subroutine	CALL	Jump to subroutine	2	2
	RET	Return	1	2
	RETR	Return and restore status	1	2
Flags	CLR C	Clear Carry	1	1
	CPL C	Complement Carry	1	1
	CLR F0	Clear Flag 0	1	1
	CPL F0	Complement Flag 0	1	1
	CLR F1	Clear Flag 1	1	1
	CPL F1	Complement Flag 1	1	1
Data Moves	MOV A, R	Move register to A	1	1
	MOV A, @R	Move data memory to A	1	1
	MOV A, #data	Move immediate to A	2	2
	MOV R, A	Move A to register	1	1
	MOV @R, A	Move A to data memory	1	1
	MOV R, #data	Move immediate to register	2	2
	MOV @R, #data	Move immediate to data memory	2	2
	MOV A, PSW	Move PSW to A	1	1
	MOV PSW, A	Move A to PSW	1	1
	XCH A, R	Exchange A and register	1	1
	XCH A, @R	Exchange A and data memory	1	1
	XCHD A, @R	Exchange nibble of A and register	1	1
	MOVX A, @R	Move external data memory to A	1	2
	MOVX @R, A	Move A to external data memory	1	2
	MOVP A, @A	Move to A from current page	1	2
	MOVP3 A, @A	Move to A from Page 3	1	2
Timer/Counter	MOV A, T	Read Timer/Counter	1	1
	MOV T, A	Load Timer/Counter	1	1
	STRT T	Start Timer	1	1
	STRT CNT	Start Counter	1	1
	STOP TCNT	Stop Timer/Counter	1	1
	EN TCNTI	Enable Timer/Counter Interrupt	1	1
	DIS TCNTI	Disable Timer/Counter Interrupt	1	1
Control	EN I	Enable external interrupt	1	1
	DIS I	Disable external interrupt	1	1
	SEL RB0	Select register bank 0	1	1
	SEL RB1	Select register bank 1	1	1
	SEL MB0	Select memory bank 0	1	1
	SEL MB1	Select memory bank 1	1	1
	ENT0 CLK	Enable Clock output on T0	1	1
	NOP	No Operation	1	1

- CHAPITRE 2 : ARCHITECTURE DE LA PARTIE OPERATIVE -

2.1- DECOMPOSITION DES INSTRUCTIONS EN UN ALGORITHME D'INTERPRETATION.

2.1.1- Le point de départ

2.1.1.1- Structure de bus

2.1.1.2- Définition des phases

2.1.1.3- Décomposition d'une instruction

2.1.2- Modes d'adressage de la RAM

2.1.2.1- Adressage direct

2.1.2.2- Adressage indirect

2.1.2.3- Adressage de la pile

2.1.3- Instructions à un cycle

2.1.3.1- Exemple de l'instruction XCH A,aRr

2.1.3.2- Cas d'interruptions

2.1.3.3- Cas d'extension de la ROM

2.1.4- Instructions à deux cycles

2.1.4.1- Exécution de sous-programmes

a) Instruction CALL

b) Instruction RETR

2.1.4.2- Instructions d'entrées-sorties

a) Extension du système d'entrées-sorties

b) Extension de la mémoire RAM

2.1.4.3- Exemple de l'instruction MOVP3 A,aA

2.2- ETABLISSEMENT DE L'ARCHITECTURE.

2.2.1- Constitution de la partie opérative

2.2.1.1- Bloc arithmétique et logique

2.2.1.2- Registres de la partie opérative

2.2.1.3- Circuits spéciaux

a) Génération des constantes

b) Circuits de test de l'accumulateur

2.2.2- Plan de masse de la partie opérative

2.2.2.1- Placement des registres

2.2.2.2- Disposition des bascules

2.2.2.3- Architecture de la partie opérative

CHAPITRE 2

- ARCHITECTURE DE LA PARTIE OPERATIVE -

L'établissement de l'architecture de la partie opérative repose sur les principes de la démarche descendante. En effet, partant du jeu d'instructions et des chronogrammes des signaux de contrôle ALE, $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$ et PROG, nous avons procédé comme suit:

- Décomposition de chaque instruction en un algorithme d'interprétation, respectant la synchronisation avec les signaux de contrôle en entrée et en sortie pour être compatible avec le 8048 (travail réalisé en collaboration avec F. BERTRAND).

- Elaboration progressive de l'architecture proprement dite.

2.1- DECOMPOSITION DES INSTRUCTIONS EN UN ALGORITHME D'INTERPRETATION

2.1.1- Le point de départ

2.1.1.1- Structure des bus

Nous avons choisi une architecture à deux bus internes a et b pour permettre deux transferts en parallèle. Cette structure présente donc une plus grande souplesse dans la décomposition des instructions. Les bus sont sans précharge parce que le circuit est réalisé en technologie CMOS avec une circuiterie statique.

2.1.1.2- Définition des phases

Nous avons introduit deux phases T1 et T2 avec non recouvrement dans chaque état élémentaire, obtenu à partir de la division d'un cycle machine en cinq états. Avec un cristal de 6MHz, l'état dure 500ns et chaque phase aura donc une durée de

200 ns avec un non recouvrement de 50ns.

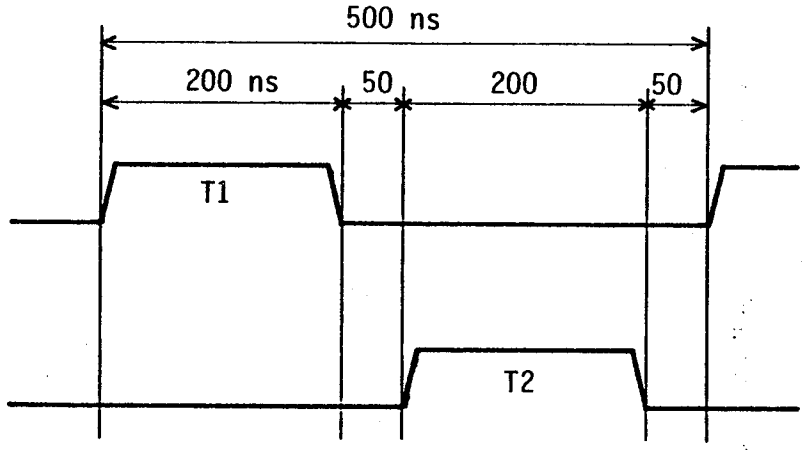


Fig. 2.1- Horloge à deux phases

Une phase autorise le transfert de registre à registre par un signal de lecture dans l'un et d'écriture dans l'autre. La lecture du registre se fait au début de la phase et doit être maintenue jusqu'à la fin de cette phase, pour assurer un bon transfert des données pendant l'écriture.

2.1.1.3- Décomposition d'une instruction

D'une manière générale, une instruction se déroule de la manière suivante:

1- Acquisition du code opération,

2- Décodage de l'instruction courante et préparation de l'adresse suivante,

3- Exécution de l'instruction en cours et recherche de l'instruction suivante.

Dans notre cas, une instruction peut être d'un ou de deux cycles. Et chaque cycle est subdivisé en cinq états S1, S2, S3, S4 et S5.

De ces considérations, on peut déduire une première décomposition pour une instruction à un cycle.

(S1) RI <--- ROM(PC) (Le registre d'instruction RI reçoit le code opération de l'instruction).

(S2) PC <--- PC+1 (Incrémentation du compteur de programme PC).

(S3)
(S4) (Exécution de l'instruction).
(S5)

Le compteur de programme étant de 12 bits, il sera formé de deux parties: l'une de poids faible (8 bits) notée PCL et l'autre de poids fort (4 bits) notée PCH.
Son incrémentation peut se faire à l'aide de deux incrémenteurs. Mais par souci d'économie de matériel, nous avons cherché à effectuer toutes les opérations arithmétiques et logiques à travers l'unique UAL qui sera implanté dans la partie opérative. Une opération se déroule ainsi en deux phases. Deux registres tampons U1 et U2, à l'entrée de l'UAL, sont chargés systématiquement pendant la phase T1, puis le résultat de l'opération sera disponible en T2.

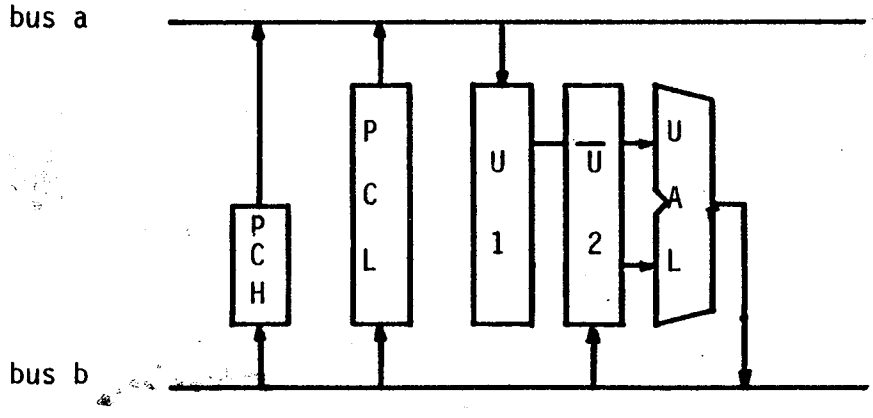


Schéma fonctionnel pour l'incrémentation du PC

Par conséquent, l'incrémentation du compteur de programme se fera en deux étapes: on incrémente d'abord le contenu du PCL et la retenue résultante sera ensuite ajoutée au contenu du PCH. A l'aide des deux phases T1 et T2 et, des deux bus a et b, cette opération peut être décrite comme suit:

	T1	T2
(1)	U1 <--- a:= PCL; cin:= 1 U2 <--- b:= 00H	PCL <--- b:= U1+U2; OVF:= cout
(2)	U1 <--- a:= PCH; cin:= OVF U2 <--- b:= 00H	PCH <--- b:= U1+U2

Note : La notation U1+U2 signifie qu'on additionne les deux opérandes U1 et U2 avec la retenue cin.

- cin et cout désignent respectivement la retenue entrante et sortante de l'UAL;
- la bascule OVF a été introduite pour mémoriser le débordement lors de l'incrémentation du PCL. L'opération sur PCH aura toujours lieu quelque soit le contenu de cette bascule.

On constate que d'une part, l'incrémentation du compteur de programme prend deux états complets et que d'autre part, le registre d'instruction doit être chargé auparavant par le code opération de l'instruction. Ceci entraîne le racourcissement du temps d'exécution. Ainsi, nous nous proposons de libérer le bus b, occupé pendant le chargement du registre tampon U2 par la constante zéro, et de commencer l'incrémentation du pointeur de programme dès le premier état S1. La remise à zéro de U2 nécessite l'introduction d'une commande RES2 (voir 3.1.2).
Les deux premiers états de l'instruction seront formulés de la

manière suivante:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= 1 RI <--- b:= ROM(PC)	PCL <--- b:= U1+U2 OVF:= cout
(S2)	U1 <--- a:= PCH; cin:= OVF	PCH <--- b:= U1+U2

Note : Pour des raisons de clarté, les opérandes U1 et U2 ne figureront pas dans la description lorsqu'ils sont mis à zéro.

Enfin, la phase d'exécution est spécifique à chaque instruction. Si on prend l'exemple des instructions qui font appel à la RAM, le calcul d'adresse de ses registres est préalable à l'exécution de l'opération relative à l'instruction. Ainsi, nous allons préciser les différentes manières utilisées pour adresser la RAM.

2.1.2- Modes d'adressage de la RAM

La RAM, étant de 64 octets, peut être adressée par un registre de 6 bits, noté W. Elle est constituée d'une banque 0 de 8 registres, d'une pile formée de 8 couples de registres, d'une banque 1 de 8 registres et d'une autre partie composée de 32 registres.

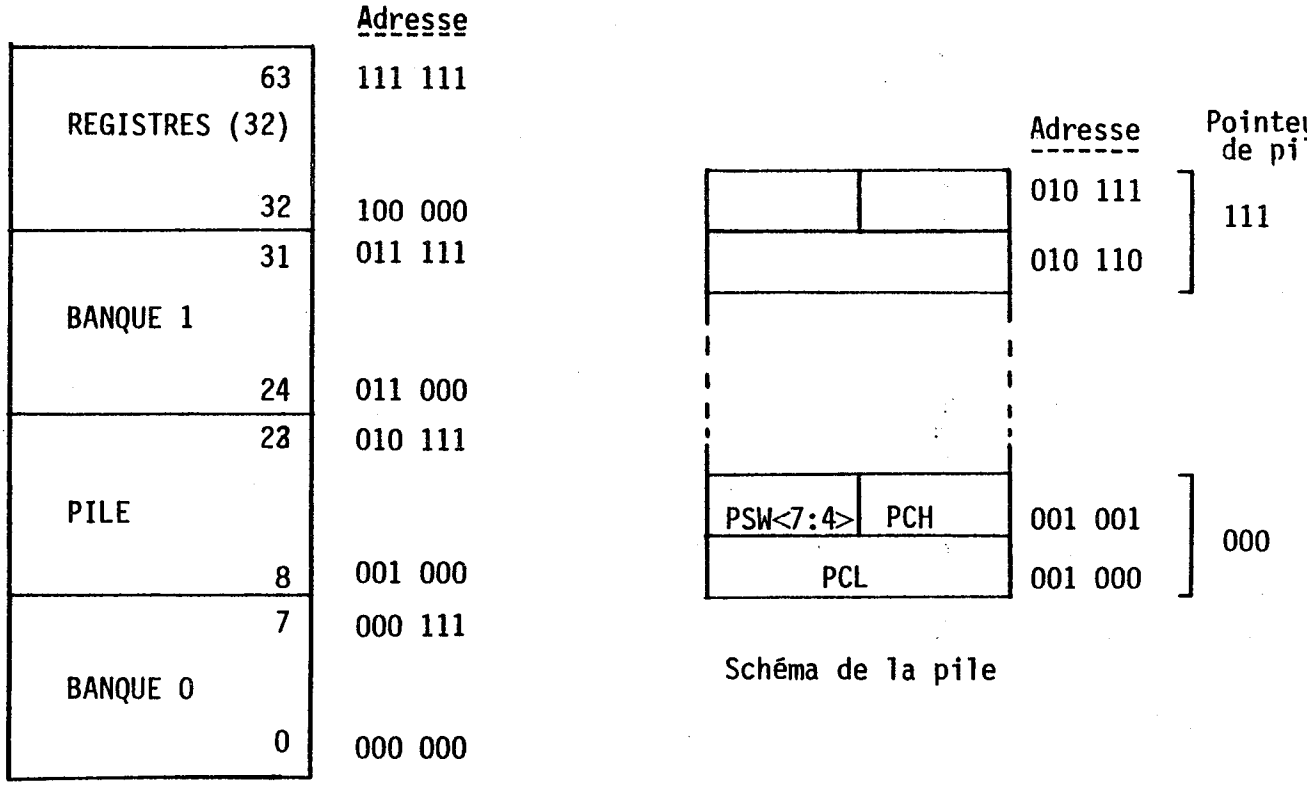


Fig. 2.2- Constitution de la RAM

La sélection de l'une des deux banques est réalisée en positionnant la bascule BS située dans le registre d'état PSW. Cette bascule envoie un compte-rendu à la partie contrôle.

BS=0, la banque 0 est sélectionnée.

BS=1, la banque 1 est sélectionnée.

L'adressage de ces deux banques est direct.

Remarque : Pour BS=0, on doit charger W<5:3> par 000.
Pour BS=1, W<5:3> prend la valeur 011.
Dans ces deux cas, la partie W<2:0> est chargée par RI<2:0> (puisque les 3 bits de poids faible du code opération des instructions faisant appel à la RAM renferment l'adresse du registre).

L'adressage de la partie contenant les 32 registres est

indirect. L'adresse de l'opérande est contenue dans l'un des deux premiers registres R0 ou R1 de la banque choisie, qui sont adressables directement.

Quant à la gestion de la pile, elle est assurée par un pointeur de 3 bits, situé dans le registre d'état PSW (bits de 0 à 2). L'adresse de la pile de la RAM se calcule à partir du contenu du PSW<2:0> auquel on ajoute la constante hexadécimale 04. Elle est finalement obtenue en décalant le résultat de l'opération à gauche.

Exemple : Soit à calculer l'adresse de la pile correspondante à la valeur 001 du pointeur de pile.

Pointeur de pile PSW<2:0>=001
Constante = 0000 0100 (= 04H)
Résultat = 001 + 0000 0100 = 0000 0101
Décalage à gauche donne 0000 1010
L'adresse sera donc 001010 pour le premier octet de la pile. Elle est ensuite incrémentée pour pointer l'octet suivant à l'adresse 001011.

La RAM possède donc trois modes d'adressage.

2.1.2.1- Adressage direct

Ce mode d'adressage est utilisé uniquement pour adresser les deux banques de la RAM. On a vu précédemment que W<5:3> reçoit le contenu 000 ou 011 selon que l'on sélectionne la banque 0 ou la banque 1 (voir fig.2.2).
D'autre part, l'adresse du registre est stipulée dans les bits 2 à 0 du code opération des instructions concernées.

Exemple : L'instruction ADD A,Rr a pour code opération 0110 1rrr. Le champ rrr désigne l'adresse du registre choisi parmi les 8 registres d'une banque de la RAM.

Le calcul d'adresse est simple. Il suffit en effet de transférer le code opération dans le registre W et de charger ses bits 5 à 3 par les constantes 000 ou 011. Ces transferts sont effectués en deux temps et le contenu de W sera finalement soit 000 rrr ou 011 rrr selon la banque sélectionnée.

2.1.2.2- Adressage indirect

Ce mode d'adressage permet de pointer tous les registres de la RAM, y compris ceux de la pile. L'adresse est contenue dans l'un des deux premiers registres R0 ou R1 d'une banque, spécifié par l'utilisateur en positionnant le bit 0 du code opération de l'instruction.

Exemple : L'instruction ADD A,aRr a pour code opération 0110 000r. Comme précédemment, le champ 00r désigne l'adresse du registre qui nous intéresse.

L'acquisition d'adresse se fait en deux étapes:

1.- W reçoit d'abord la valeur 000 00r ou 011 00r selon la banque choisie. Cela correspond à un adressage direct des registres R0 ou R1.

2.- Le contenu de ce registre de la RAM est ensuite transféré au registre d'adresse W, par l'intermédiaire d'un registre temporaire RT. On note cette action par deux transferts:

RT <--- RAM(W)

W <--- RT

2.1.2.3- Adressage de la pile

Généralement, on n'adresse la pile que pendant les instructions CALL et RETR (ou RET). Le calcul d'adresse s'effectue à travers l'UAL, à l'aide des constantes respectives 04H et 03H. On charge

en T1 le pointeur de pile PSW<2:0> dans le registre tampon U1 pendant que ses bits 7 à 3 sont remis à zéro; l'adresse du registre est obtenue en T2. Ce qui s'écrit:

en T1: U1 <--- a:= PSW<2:0>; U1<7:3>:=0
U2 <--- b:= (04H ou 03H)

en T2: W <--- b:= SL (U1+U2)*

* La notation SL (U1+U2) signifie qu'on effectue un décalage à gauche au résultat de l'opération d'addition.

Pour adresser l'octet suivant, il suffit d'incrémenter le contenu de W à travers l'UAL.

Remarque : Les constantes 04H et 03H peuvent être soit récupérées des quatre bits de poids faible du code opération de l'instruction CALL (aaa1 0100) et de l'instruction RETR (1001 0011), soit générées par un circuit de constantes.

En conclusion, on constate que l'adressage indirect est le plus long des trois modes, puisqu'il se fait en deux étapes. Ainsi, nous nous proposons de décomposer une instruction utilisant ce type d'adressage.

2.1.3- Instruction à un cycle

2.1.3.1- Exemple de l'instruction XCH A,aRr

Cette instruction permet de faire l'échange entre le registre de l'accumulateur et celui de la RAM. Pour effectuer ces transferts, on doit passer par un registre intermédiaire RT.

RT <--- A D'abord, le contenu de A est sauvegardé dans RT.

A <--- RAM(W) Puis, le contenu du registre de la RAM est transféré dans A.

RAM(W) <--- RT Finalement, le registre de la RAM reçoit le contenu initial de l'accumulateur par l'intermédiaire de RT.

Ces transferts prennent au minimum trois phases d'horloge d'une part et on doit satisfaire l'hypothèse suivante d'autre part.

Hypothèse : L'ACCES A LA RAM INTERNE PREND UN (1) ETAT.

En effet, l'optimisation de la surface de cette mémoire, exige en contre partie un temps d'accès plus long qui dure un état. Après acquisition de l'adresse par la RAM, l'émission ou la réception des données de cette RAM nécessite deux phases d'horloge.

Par conséquent, l'adressage indirect prendra deux états, (S3) et (S4), pendant lesquels on aura les transferts suivants:

	T1	T2
(S3)	W <--- 000 rrr ou 011 rrr	
(S4)	RT <--- RAM(W)	W <--- RT

Du fait de l'hypothèse ci-dessus, on remarquera qu'un nouvel accès à la RAM ne pourra avoir lieu qu'à partir de (S5,T2). Ce qui est insuffisant pour réaliser l'opération spécifique à l'instruction XCH A,aRr qui consiste à faire l'échange entre l'accumulateur et le registre de la RAM. Afin de résoudre ce problème, il est nécessaire de commencer le premier chargement de W avant l'état S3. Pour cela on l'effectue en deux temps:

- lors de la récupération du code opération en (S1,T1),

- par l'utilisation du bus b qui est libre en (S2,T1).

Ainsi, l'algorithme d'interprétation de l'instruction XCH A,aRr sera formulé de la façon suivante:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= 1	PCL <--- b:= U1+U2;
	W, RI <--- b:= ROM(PC)	OVF:= cout
(S2)	U1 <--- a:= PCH; cin:= OVF	PCH <--- b:= U1+U2
	W<7:3> <--- b:= 00H ou 18H	
(S3)	U1 <--- a:= RAM(W); cin:= 0	W <--- b:= U1+U2
(S4)	U1 <--- a:= A; cin:= 0	RT <--- b:= U1+U2
(S5)	U1 <--- a:= RAM(W); cin:=0	A <--- b:= U1+U2
		RAM(W) <--- a:= RT

Dans cette décomposition, on respecte bien un état pour le décodage de la RAM.

L'adresse du registre est récupérée du code opération de l'instruction en (S1,T1) qui est chargé simultanément dans le registre d'instruction RI et dans le registre d'adressage (W) de la RAM. L'information qui nous intéresse pour le décodage de la RAM, se trouve dans les trois bits de poids faible W<2:0>.

En (S2,T1), on doit compléter les bits 5 à 3 de W par la constante 000 ou 011 selon la banque sélectionnée par l'utilisateur. Mais puisque d'une part, les constantes générées sur le chemin de données sont de 8 bits et que d'autre part, le registre d'adresse W a été étendu à 8 bits pour l'utiliser dans certains cas comme registre temporaire, cela revient à charger W<7:3> par les constantes 00H ou 18H respectivement.

Ainsi, l'adresse du registre est prête en (S2,T1) et il faut attendre un état pour avoir accès à la RAM. Celle-ci nous délivre en retour la donnée en (S3,T1) qui est chargée dans W en (S3,T2). On a de nouveau accès à la RAM en (S4,T2). Mais, on a voulu faire systématiquement la lecture de la RAM pendant la phase T1 et son écriture en T2. Par conséquent, l'échange entre la RAM et l'accumulateur A ne s'effectue qu'à partir de l'état S5.

D'après ce principe de décomposition de l'instruction en un algorithme, on aura tous les registres de travail implantés dans la partie opérative, ainsi que le registre d'instruction RI, le registre d'état PSW et le registre d'adressage W de la RAM.
A l'aide de ces registres et d'une structure à deux bus, on peut décrire toutes les instructions en un algorithme simple. Cependant, il reste à considérer l'incrémentation éventuelle du compteur T (voir 1.1.5) qui se fait en S4. On a le choix entre deux solutions:

- Implanter le compteur dans la partie opérative et faire son incrémentation à travers l'UAL. Ceci entraîne la modification de l'algorithme précédent.
- L'implanter dans une autre partie du circuit et dans ce cas, on aura besoin d'un incrémenteur mais, par contre, l'algorithme précédent reste le même.

On a préféré la première solution puisque le compteur est de 8 bits et s'implante donc facilement dans la partie opérative. Du coup, la décomposition de l'instruction précédente se trouve essentiellement transformée à partir de l'état S4, puisque l'incrémentation du compteur T, si elle a lieu, se fait dans cet état. On aura alors, pour l'instruction XCH A,aRr, les modifications suivantes:

	T1	T2
(S4)	U1 <--- a:= T; cin:= 0 ou 1	T <--- b:= U1+U2

(S5) U1 <--- a:= RAM(W); cin:= 0 RAM(W) <--- a:= RT
RT <--- b:= A A <--- b:= U1+U2

Remarque : L'opération sur le compteur T se déroule en S4 pour toutes les instructions. La retenue entrante cin aura pour valeur "1" si l'incrémentation doit avoir lieu. Dans le cas contraire, elle sera nulle.

On est obligé de sauvegarder le contenu de l'accumulateur dans RT pendant la phase T1 de l'état S4 ou S5. On a choisi S5 pour être compatible avec d'autres instructions telles que XCH A,Rr, MOVA,Rr et MOV A,aRr. D'ailleurs, pour toutes les instructions, la seule opération effectuée en (S4,T1) est le chargement du compteur T dans U1.

Nous pouvons ainsi décomposer toutes les instructions selon le schéma ci-dessus (de la description de l'instruction XCH A,aRr). Pour terminer la présentation des instructions à un cycle, nous allons maintenant examiner les cas d'interruptions et d'extension de la ROM.

2.1.3.2- Cas d'interruptions

Dès la détection de l'interruption, l'instruction CALL est exécutée à la fin de l'instruction en cours. Cela revient à forcer le registre d'instruction RI à la valeur 14H, correspondant au code opération du CALL, au début de l'instruction suivante.
Selon le résultat du test de l'interruption réalisé à l'aide du signal IT (voir fig. 4.4), on charge RI par la constante 14H ou par le code opération de l'instruction suivante en (S1,T1). De plus, dans le cas d'une interruption,le compteur de programme ne doit pas être incrémenté. Ainsi, la retenue entrante cin prendra la valeur zéro en (S1,T1).
On pourra alors formuler le premier état de toutes les instructions de la manière suivante:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= (0 ou 1)	PCL <--- b:= U1+U2
	W, RI <--- (b:= 14H ou ROM(PC))	OVF:= cout

2.1.3.3- Cas d'extension de la ROM

Pour l'extension de la mémoire morte, on a vu d'après les spécifications (voir 1.2.1) que d'une part, l'adresse est sortie sur le port BUS et la partie de poids faible du port 2. Elle est validée à l'extérieur par le signal ALE en (S4,T2).
D'autre part, le contenu du port 2 ne doit pas être détruit pendant l'extension de la ROM. Ainsi, ce port sera constitué de deux parties:

- la partie mémorisant les données de 8 bits en sortie, notée DP2.
- la partie contenant les quatre bits d'adresse (de poids fort) de la ROM, notée AP2L. Ce registre sera chargé systématiquement et en même temps que PCH, en (S2,T2) et éventuellement en (S7,T2) pour les instructions à deux cycles.

Par contre, le port BUS n'est formé que d'un seul registre de sortie, noté aussi BUS. Il sert comme port d'entrées-sorties, mais aussi pour la sortie de l'adresse en cas d'extension de la mémoire ROM ou RAM (voir 1.2.2). Cette dernière utilisation détruit une donnée mémorisée auparavant dans ce port.
Le registre de sortie BUS ne reçoit le contenu du PCL que si le bit 3 ou le bit 2 du PCH est à "1". En effet, d'une part le bit 3 du PCH permet de sélectionner la mémoire morte interne ou externe. Il est affecté uniquement par les instructions CALL et JMP. D'autre part, le 8048 contient une mémoire ROM de 1 Kilo-octet et donc un compteur de programme de 10 bits (dont 2 bits pour le PCH) suffit pour l'adresser. Si le bit 2 du PCH est mis à "1" lors de son incrémentation, alors la mémoire externe

est automatiquement adressée. Puisque les registres AP2L et PCH contiennent les mêmes informations, alors le chargement de BUS est soumis à la condition AP2L<3> ou AP2L<2> égal à "1". Celle-ci est réalisée par un "OU" logique entre les bits 3 et 2 de AP2L, dont la sortie est envoyée comme compte-rendu à la partie contrôle.

En cas d'adressage de la mémoire externe, celle-ci est activée par le signal de sortie $\overline{PSEN}$ pendant le dernier état d'un cycle et le premier état du cycle suivant (voir fig. 2.3). La donnée est retournée par la ROM sur les broches DB du port BUS. Elle est lue en (S1,T1) et éventuellement en (S6,T1) pour les instructions à deux cycles.

En tenant compte de ces considérations et en introduisant un bus externe g, relié au bus b de la partie opérative, pour desservir les ports d'entrées-sorties; les deux premiers états de chaque instruction s'écrivent alors:

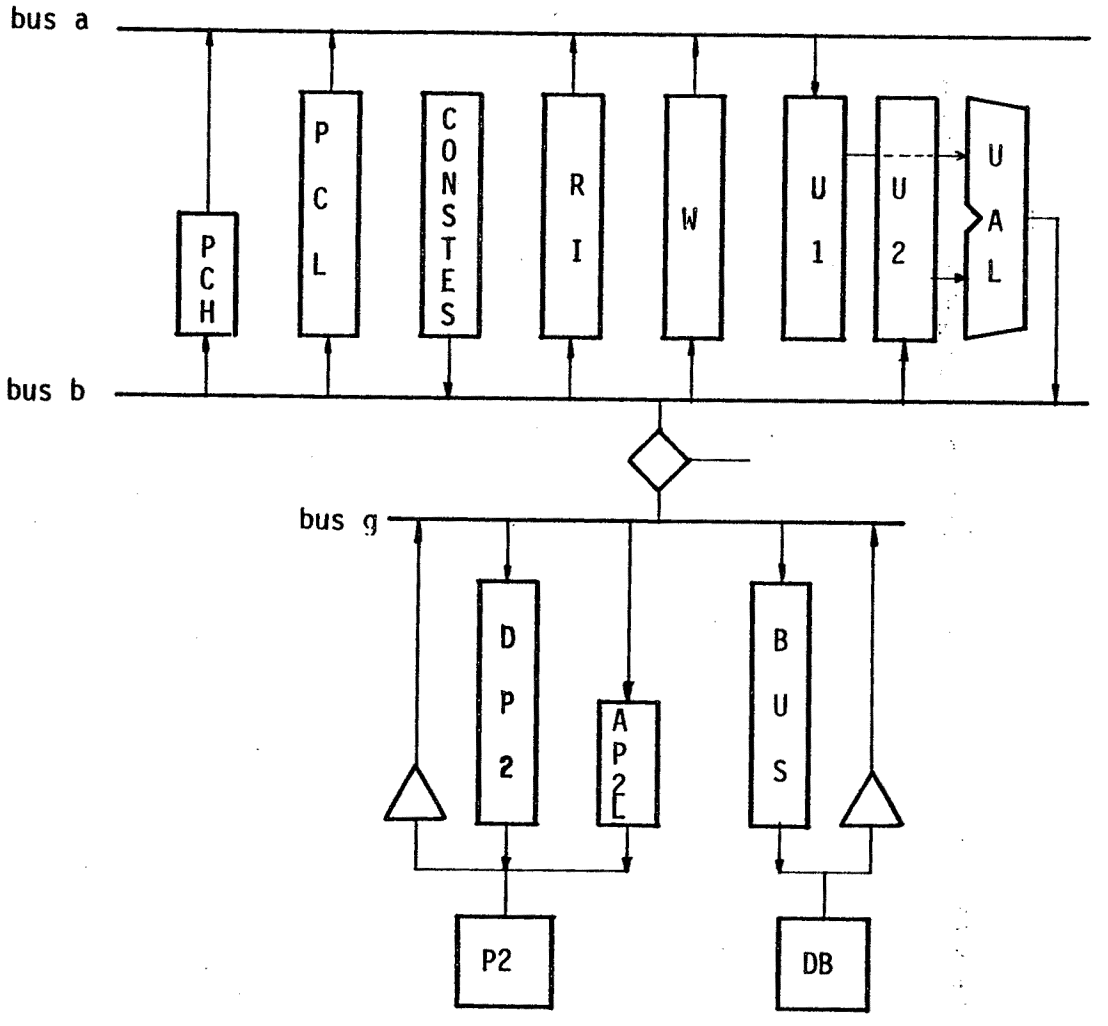
	T1	T2
(S1)	U1 <--- a:= PCL; cin:= (0 ou 1)	PCL <--- b:= U1+U2
	W, RI <--- b:= 14H	(BUS <---gb:= U1+U2)**
	ou (b:= ROM(PC) ou bg:= DB)*	OVF:= cout
(S2)	U1 <--- a:= PCH; cin:= OVF	PCH <--- b:= U1+U2
	W<5:3> <--- b:= (00H ou 18H)	AP2L <---gb:= U1+U2

* si AP2L<3>+AP2L<2>=1 alors W, RI <--- bg:= DB
sinon W, RI <--- b:= ROM(PC)

** Le registre BUS n'est pas chargé systématiquement comme AP2L, pour éviter la destruction de son contenu précédent. Le transfert n'aura lieu que si AP2L<3>+AP2L<2>=1.

Note : La notation gb ou bg signifie qu'on utilise les deux bus b et g pour véhiculer la donnée entre les entrées-sorties et les registres de la partie opérative, dans un sens ou dans

l'autre (voir schéma ci-dessous).



Pour les instructions à deux cycles et à deux octets, les états S6 et S7 seront respectivement formulés d'une façon identique à S1 et S2.

2.1.4- Instructions à deux cycles

Elles concernent surtout les instructions à deux octets (adressage immédiat, branchements et sous-programmes), ainsi que les instructions d'entrées-sorties et de transferts qui sont à un seul octet.

Le second cycle est formé des états S6 à S10. Il est décomposé

d'une manière identique au premier cycle.
Un nouveau chargement du compteur de programme s'effectuera pendant les deux premiers états S6 et S7 du second cycle.
La donnée (ou l'adresse) immédiate n'est disponible qu'au début du second cycle. Elle est stockée dans W (ou dans le registre tampon U2) pendant (S6,T1). Ainsi, pour l'ensemble des instructions, on aura la description suivante:

	T1	T2
(S6)	U1 <--- a:= PCL; cin:= 1	PCL <--- b:= U1+U2;
	W <--- (b:= ROM(PC) ou bg:= DB)	(BUS <---gb:= U1+U2)
		OVF:= cout
(S7)	U1 <--- a:= PCH; cin:= OVF	PCH <--- b:= U1+U2
		AP2L <---gb:= U1+U2

Pour les instructions à adressage immédiat, la donnée immédiate est chargée dans le registre W qui joue le rôle de registre temporaire en S6 et n'est utilisée en général qu'à l'état S10 pour effectuer l'opération à travers l'UAL.
Pour les instructions de branchement, l'opération en S6 est différente selon le résultat de leur test. Le compteur de programme PCL est soit incrémenté, soit chargé par l'adresse de branchement. Ce qui peut s'écrire:

	T1	T2
(S6)	(U1 <--- a:= PCL; cin:= 1)	PCL <--- b:= U1+U2
	ou	(BUS <---gb:= U1+U2)
	(U2 <--- b:= ROM(PC); cin:= 0)	OVF:= cout

Si le branchement a lieu, alors l'adresse correspondante est chargée dans le registre tampon U2 pendant que U1 est remis à zéro. Sinon, le programme se déroule normalement et l'état S6 sera identique à celui des instructions dans lesquelles l'incrémentation du compteur de programme se fait à cet instant.

Pour les instructions de sous-programmes, leur décomposition est un peu particulière. Pour cela, nous allons les étudier en détail.

2.1.4.1- Exécution de sous-programmes

L'appel à un sous-programme est possible grâce aux deux instructions suivantes:

- l'instruction CALL qui provoque l'exécution du sous-programme,
- l'instruction RETR (RET) qui permet le retour au programme appelant.

a) Instruction CALL

Elle permet de sauvegarder, avant l'exécution du sous-programme, le contenu du compteur de programme (PCL et PCH) et les quatre bits de poids fort du registre d'état PSW, notés PSH<7:4>, dans la pile.

On a vu qu'un calcul d'adresse est nécessaire pour adresser la pile. Il est effectué à travers l'UAL en S3 (voir 2.1.2.3). On utilise la constante 40H, au lieu de 04H, parce que le pointeur de pile occupera les bits 6 à 4 du PCH dans la partie opérative (voir 2.2.2.2).

L'instruction CALL peut s'écrire:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= (0 ou 1) W, RI <--- b:= 14H ou (b:= ROM(PC) ou bg:= DB) (P2:= AP2L)	RT, PCL <--- b:= U1+U2 BUS <---gb:= U1+U2 OVF:= cout (P2:= AP2L)
(S2)	U1 <--- a:= PCH; cin:= OVF W<7:3> <--- b:= (00H ou 18H)	PCH<2:0> <--- b:= U1+U2 AP2L<2:0> <---gb:= U1+U2
(S3)	U1<6:4> <--- a:= PCH; U1<7>,U1<3:0>:= 0 U2 <--- b:= 40H	W <--- a:= SL (SW (U1+U2))
(S4)	U2 <--- b:= T; cin:= (0 ou 1)	T <--- b:= U1+U2 RAM(W) <--- a:= PCL (DB:= BUS)
(S5)	U2 <--- b:= W; cin:= 1 (P2:= AP2L)	W <--- a:= U1+U2 (P2:= AP2L)
(S6)	U2 <--- (b:= 03H ou 07H)* ou (b:= ROM(PC) ou bg:= DB) (P2:= AP2L)	RT, PCL <--- b:= U1+U2 RAM(W) <--- a:= PSH<7:4>, PCH<3:0> (P2:= AP2L)
(S7)	U1 <--- a:= RI; cin:= 0	PCH<2:0> <--- b:= SR (SW (U1+U2)) AP2L<2:0> <---gb:= SR (SW (U1+U2))
(S8)	U1 <--- a:= PCH; cin:= 0	W <--- a:= SW (U1+U2) PCH<3> <--- b:= SW (U1+U2) AP2L<3> <---gb:= SW (U1+U2)

(S9) U1 <--- a:= RT; cin:=0 BUS <--- gb:= U1+U2
(DB:= BUS)

(S10) U1 <--- a:= W; cin:= 1 PCH<6:4> <--- b:= SW (U1+U2)
(P2:= AP2L) (P2:= AP2L)

* En cas d'interruption, on chargera PCL par les constantes:

07H pour une interruption due au compteur T,
03H pour une interruption provenant de l'extérieur.

L'opération sur le compteur T se déroule toujours en S4. Dans cet état, le bus b (libre pour les autres instructions) est utilisé pour la sauvegarde du contenu du PCL dans le sommet de la pile en (S4,T2).
En S5, on incrémente le contenu de W pour pointer le registre suivant de la pile. Celui-ci sera prêt à recevoir les contenus du PCH et du PSH<7:4> en (S6,T2).
En S6, l'adresse (8 bits de poids faible) du sous-programme est d'abord chargée dans le registre tampon U2 pour être transférée dans PCL en (S6,T2), à travers l'UAL. Le compteur de programme reçoit la donnée immédiate de la ROM dans le cas d'un sous-programme appelé par le programmeur. Il est forcé respectivement à l'adresse 03 ou 07 lors de la détection d'une interruption externe ou interne due au compteur T.
Les 3 bits de poids fort de cette adresse sont récupérés de ceux du code opération de l'instruction et chargés dans PCH en (S7,T2).
Le bit 3 du PCH est affecté en (S8,T2). Il est chargé par la bascule DBF, contenue dans le bit 7 du PCH (voir 2.2.2.2).
En S9, le chargement du registre BUS par la partie de poids faible de l'adresse de la sous-programme est conditionnel (voir 2.1.4.2,a).
Finalement, on incrémente le pointeur de pile en S10 pour sa mise à jour.

b) Instruction RETR

Elle permet de récupérer les contenus du compteur de programme et des quatre bits de poids fort du PSW, sauvegardés auparavant dans la pile par l'instruction CALL.

Le calcul d'adresse se fait comme précédemment en S3, mais avec la constante 30H, pour adresser la paire de registres contenant l'information qui nous intéresse.

En S4, on effectue uniquement l'opération sur le compteur T.

Afin de pouvoir incrémenter le registre d'adresse W en S5, on récupère l'adresse (8 bits de poids faible) du programme en (S5,T1) et on la stocke momentanément dans le registre temporaire RT.

Cette adresse est finalement chargée dans PCL en S6.

En S7, on récupère le contenu du registre suivant de la pile dont les 4 bits de poids fort sont chargés dans PSH<7:4> et ceux de poids faible dans PCH.

En S8, on transfère le contenu du pointeur de pile dans le registre d'adresse W pour être décrémenté en S10.

L'état S9 est identique à celui de l'instruction CALL.

Ce qui peut s'écrire:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= (0 ou 1)	RT, PCL <--- b:= U1+U2
	W, RI <--- b:= 14H ou (b:= ROM(PC) ou bg:= DB)	BUS <---gb:= U1+U2
	(P2:= AP2L)	OVF:= cout (P2:= AP2L)
(S2)	U1 <--- a:= PCH; cin:= OVF	PCH<2:0> <--- b:= U1+U2
	W<7:3> <--- b:= (00H ou 18H)	AP2L<2:0> <---gb:= U1+U2
(S3)	U1<6:4> <--- a:= PCH; U1<7>,U1<3:0>:= 0	
	U2 <--- b:= 30H	W <--- a:= SL (SW (U1+U2))

(S4)	U2 <--- b:= T; cin:= (0 ou 1)	T <--- b:= U1+U2 (DB:= BUS)
(S5)	RT <--- a:= RAM(W) U2 <--- b:= W; cin:= 1 (P2:= AP2L)	W <--- a:= U1+U2 (P2:= AP2L)
(S6)	U1 <--- a:= RT; cin:= 0 (P2:= AP2L)	RT, PCL <--- b:= U1+U2 (P2:= AP2L)
(S7)	U1 <--- a:= RAM(W); cin:= 0	(PSH<7:4> <--- b:= U1+U2)* PCH<3:0> <--- b:= U1+U2 AP2L<3:0> <---gb:= U1+U2
(S8)	U1 <--- a:= PCH; cin:= 0	W <--- a:= SW (U1+U2)
(S9)	U1 <--- a:= RT; cin:=0	BUS <---gb:= U1+U2 (DB:= BUS)
(S10)	U1 <--- a:= W; cin:= 0 U2:= FF (P2:= AP2L)	PCH<6:4> <--- b:= SW (U1+U2) (P2:= AP2L)

* Instruction RETR seulement (c'est la seule différence entre les instructions RETR et RET).

C'est une instruction à un seul octet, ainsi que les instructions d'entrées-sorties.

2.1.4.2- Instructions d'entrées-sorties

Grâce aux signaux de contrôle $\overline{WR}$, $\overline{RD}$, ALE, PROG et $\overline{PSEN}$, le micro-ordinateur peut communiquer avec l'extérieur.

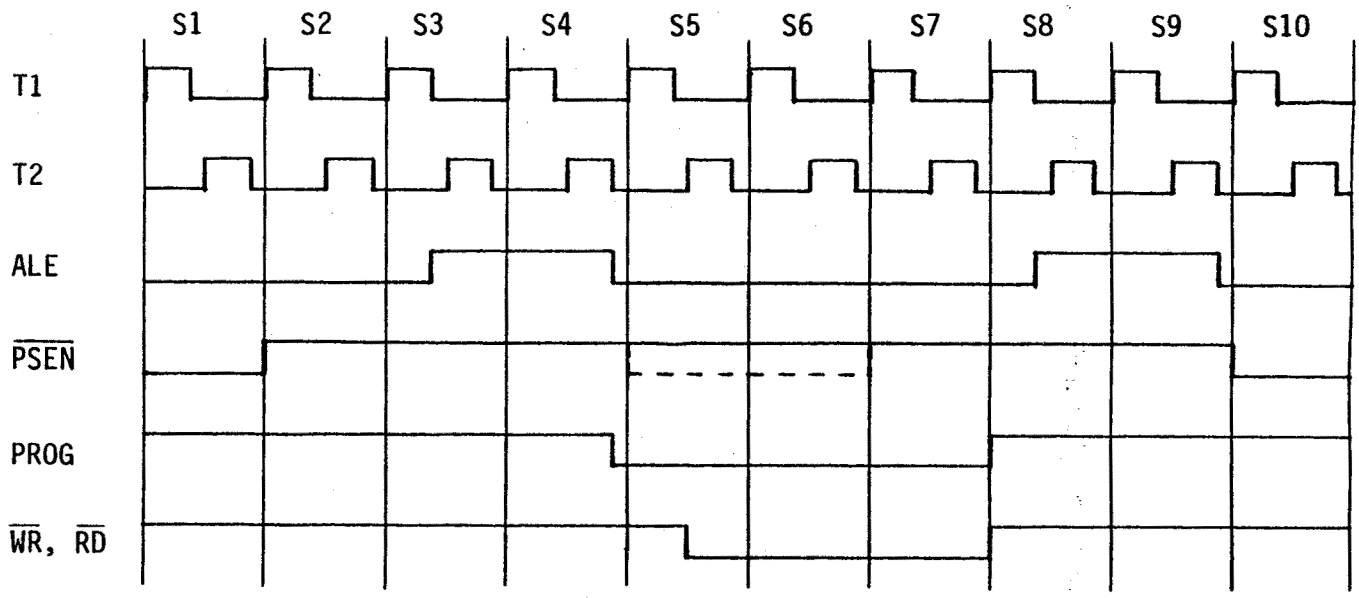


Fig. 2.3- Chronogramme des signaux de contrôle extérieur

D'autre part, un bus externe g, relié au bus interne b de la partie opérative, dessert tous les ports d'entrées-sorties BUS, P1 et P2. Les sorties sont mémorisées dans des registres notés BUS, DP1 et DP2; tandis que les entrées ne le sont pas.
Les ports P1 et P2 peuvent traiter à la fois les entrées pour certains bits et les sorties pour les autres. Une broche programmée à "1", par exemple, travaille en entrée.
Parmi les instructions d'entrées-sorties, quatre permettent d'étendre le système d'entrées-sorties et deux pour travailler avec la mémoire RAM externe.

a) Extension du système d'entrées-sorties

Le système d'entrées-sorties est étendu à l'aide des instructions MOVD A,Pi, MOVD Pi,A, ANLD Pi,A et ORLD Pi,A. Pi désigne les quatre ports (de 4 bits) du microprocesseur 8243 et toute communication avec le processeur maître se fait grâce au signal de sortie PROG et aux quatre bits de poids faible du

port 2, notés DP2L. Ainsi, le registre de sortie DP2 de ce port se trouve divisé en deux parties DP2L et DP2H. Le transfert de données entre le 8048 et le 8243 (voir 1.2.3) est effectué en deux temps:

1- Le premier mot est validé pendant le front descendant du signal PROG, c'est à dire en (S4,T2). Il contient deux bits de code d'instruction et deux autres pour l'adresse du port.
Il est déterminé à partir du code opération de l'instruction en lui ajoutant une constante (voir tableau des constantes ci-dessous). Le résultat est stocké dans les quatre bits de poids faible du port 2, notés DP2L. Cette opération doit être faite à l'état S3, puisque l'état S4 est occupé par l'incrémentation éventuelle du compteur T. Ce qui peut s'écrire:

	T1	T2
(S3)	U1 <--- a:= RI U2 <--- b:= constante	DP2L <--- gb:= U1+U2

Tableau des constantes

Instructions	Code opération	Contenu du port 2	Constante à ajouter*
MOVD A,Pi	0000 11pp	00pp	x4H
MOVD Pi,A	0011 11pp	01pp	x8H
ANLD Pi,A	1001 11pp	11pp	x0H
ORLD Pi,A	1000 11pp	10pp	xCH

* Pour ces instructions, seulement la partie poids faible de ces constantes nous intéresse.

2- Le second mot est validé pendant le front montant du signal

PROG, c'est à dire en (S8,T1). Il représente la donnée de quatre bits à transférer en entrée ou en sortie. On peut lire un port du 8243 ou y écrire grâce aux quatre instructions précédentes.

L'écriture dans le port Pi se fait facilement. On stocke d'abord la donnée dans la partie DP2L du port 2, puis on la transfére vers le port choisi du 8243 en (S8,T1). Quant à la lecture, on doit d'abord positionner les quatre bits de poids faible du port 2 à "1" pour qu'ils puissent travailler en entrée et soient prêts à accepter la donnée du port Pi en (S8,T1).

Prenons par exemple l'instruction ORLD Pi,A. Elle permet de masquer certains bits du port Pi du 8243 en faisant le "OU" logique avec le contenu de l'accumulateur A. Pendant l'état S6, on met à "1" la partie DP2L du port 2 pour y recevoir le contenu du port Pi en (S7,T1). L'opération de masquage est effectuée à travers l'UAL et le résultat est sauvegardé dans DP2L pour être finalement transféré en (S8,T1) vers le port Pi.

	T1	T2
(S6)	U2:= FFH; cin:= 0	DP2L <--- gb:= U1+U2
(S7)	U1 <--- a:= A U2 <---bg:= P2	DP2L <--- gb:= U1+U2

b) Extension de la mémoire RAM

Deux instructions MOVX A,aRr et MOVX aRr,A nous permettent de communiquer avec la RAM externe. On fait d'abord sortir l'adresse de la RAM par l'intermédiaire du registre BUS. Elle est disponible à l'état S3, comme toutes les instructions utilisant le mode d'adressage indirect (voir 2.1.3.1), mais elle n'est validée qu'en (S4,T2) par le signal ALE.

	T1	T2
(S3)	U1 <--- a:= RAM(W); cin:=0	BUS <---gb:= U1+U2
(S4)	U1 <--- a:= T; cin:= (0 ou 1)	T <--- b:= U1+U2 (DB:= BUS)

Les transferts de données sont réalisés à l'aide des signaux de sortie $\overline{RD}$ et $\overline{WR}$ pendant leur front montant en (S8,T1). Pour la première instruction MOVX A,aRr, la donnée est retournée sur les broches DB et validée par le signal $\overline{RD}$. Donc, l'accumulateur A ne peut être chargé qu'à partir de l'état S8.

	T1	T2
(S8)	U2 <--- bg:= DB; cin:= 0	A <--- b:= U1+U2

Pour la seconde instruction MOVX aRr,A, la donnée (le contenu de l'accumulateur) est chargée dans le registre de sortie BUS en S5 pour respecter le côté systématique d'autres instructions. Cependant, elle n'est validée qu'en (S8,T1) par le signal $\overline{WR}$.

	T1	T2
(S5)	U2 <--- b:= A; cin:= 0	BUS <--- gb:= U1+U2
(S6) et (S7):	rien	
(S8)	DB:= BUS	

2.1.4.3- Exemple de l'instruction MOVP3 A,aA

Dans l'instruction MOVP3 A,aA, l'accumulateur adresse une case mémoire (de la page 3) dont le contenu sera transféré dans ce même accumulateur. Le compteur de programme est remis à jour après cette opération.
Il s'agit d'un adressage indirect de la ROM. On charge d'abord le compteur de programme : PCL reçoit le contenu de l'accumulateur et PCH la valeur 011. Puis, le contenu de la case mémoire située à cette adresse est transféré dans l'accumulateur.

L'algorithme de l'instruction MOVP3 A,aA sera transcrit de la façon suivante:

	T1	T2
(S1)	U1 <--- a:= PCL; cin:= (0 ou 1) W, RI <--- b:= 14H ou (b:= ROM(PC) ou bg:= DB) (P2:= AP2L)	RT, PCL <--- b:= U1+U2 (BUS <---gb:= U1+U2)* OVF:= cout (P2:= AP2L)
(S2)	U1 <--- a:= PCH; cin:= OVF W<7:3> <--- b:= (00H ou 18H)	PCH<2:0> <--- b:= U1+U2 AP2L<2:0> <---gb:= U1+U2
(S3)	U2 <--- b:= A; cin:= 0	PCL <--- b:= U1+U2 (BUS <---gb:= U1+U2)* A <--- a:= PCH
(S4)	U2 <--- b:= T; cin:= (0 ou 1)	T <--- a:= U1+U2 PCH<2:0> <--- b:= W AP2L<2:0> <--- gb:= W (DB:= BUS)
(S5)	(P2:= AP2L)	(P2:= AP2L)
(S6)	U1 <--- a:= RT; cin:= 0 W <--- b:= (ROM(PC) ou bg:= DB) (P2:= AP2L)	PCL <--- b:= U1+U2 BUS <---gb:= U1+U2 OVF:= cout (P2:= AP2L)

(S7) U2 <--- b:= A; cin:= OVF PCH<2:0> <--- b:= U1+U2
AP2L<2:0> <---gb:= U1+U2

(S8)

(S9) (DB:= BUS)

(S10) U1 <--- a:= W; cin:= 0 A <--- b:= U1+U2
(P2:= AP2L) (P2:= AP2L)

* Le registre de sortie BUS n'est chargé que si AP2L<3> + AP2L<2> = 1.

Les deux premiers états seront identiques pour toutes les instructions. On en profite du chargement de la nouvelle adresse dans PCL en (S1,T1) pour la sauvegarder en même temps dans le registre temporaire RT. L'accès à la mémoire morte interne est lent et demande au minimum deux états, du fait de son optimisation en surface. Par conséquent, la préparation de l'adresse doit se faire le plus tôt possible. On commence à charger PCL par le contenu de l'accumulateur en S3. Le PCH est d'abord sauvegardé dans l'accumulateur en (S3,T2).
Ensuite, il est forcé à la page mémoire 3 en (S4) par l'intermédiaire du registre W qui contient les trois bits de poids faible du code opération correspondant à la valeur 011 (voir annexe C : code opération des instructions).
Le contenu de la case mémoire pointée par la nouvelle valeur du compteur de programme est retourné en (S6,T1) et stocké momentanément dans W qui joue ici le rôle de registre intermédiaire. Il n'est transféré dans l'accumulateur que pendant l'état S10 pour être cohérent avec d'autres instructions. Entre temps, le compteur de programme doit être remis à jour; son ancien contenu est restauré pendant les états S6 et S7.

2.2- ETABLISSEMENT DE L'ARCHITECTURE

2.2.1- Constitution de la partie opérative

La partie opérative est formée essentiellement d'un bloc arithmétique et logique, de plusieurs registres et de circuits spéciaux.

2.2.1.1- Bloc arithmétique et logique

Ce bloc est constitué de:

- Une unité arithmétique et logique (UAL) permettant d'effectuer quatre opérations (disjonction, addition, "ET" et "OU" logique),
- Un permuteur "SWAP" à la sortie de l'UAL, pour la permutation entre les quatre bits de poids fort et ceux de poids faible,
- Un décaleur "SHIFTER", placé après le permuteur, pour réaliser les rotations à droite et à gauche,
- Deux registres tampons U1 et U2 à l'entrée de l'UAL.

2.2.1.2- Registres de la partie opérative

On a besoin de:

- Un registre d'adresse W pour la RAM,
- Un accumulateur A,
- Un compteur T,
- Un pointeur de programme de 12 bits formé de deux parties:
  une partie poids faible de 8 bits (PCL)
  une partie poids fort de 4 bits (PCH)
- Un registre temporaire RT,
- Un registre d'instruction RI,
- Un registre d'état PSW

Ces deux derniers registres sont implantés dans la partie opérative afin de faciliter la distribution des commandes de la

partie contrôle.

Le registre d'état PSW est composé de huit bascules disposées de la manière suivante:

CY	AC	F0	BS	1	S2	S1	S0

Bit: 7 0

On fait sortir un fil de l'indicateur BS pour servir de compte-rendu à la partie contrôle. D'autre part, le bit 3 de ce registre n'est pas utilisé. On pourra y définir la bascule OVF qui permet de stocker la retenue résultante de l'incrémentation du PCL. Elle sera mise à "1" lors de la lecture du PSW par l'instruction MOV PSW,A.

Quant au registre d'instruction, ses huit sorties sont dirigées vers la partie contrôle.

2.2.1.3- Circuits spéciaux

a) Génération des constantes

Les constantes nécessaires à générer sont au nombre de dix: 00H, 18H, 0CH, 14H, 40H, 03H, 07H, 30H, 06H et 60H.
Elles sont uniquement lues sur le bus b. Nous allons résumer ci-dessous leur utilisation.

CONSTANTES	UTILISATION
00	Instructions DAA et ANLD Pi,A. En (S2,T1) de chaque instruction quand la banque 0 de la RAM est sélectionnée.
18	Instruction MOVD Pi,A. En (S2,T1) de toutes les instructions quand

	la banque 1 de la RAM est sélectionnée.
0C	Instruction ORLD Pi,A.
14	Instruction MOVD A,Pi. En (S1,T1) de toutes les instructions dans le cas de la détection d'une interruption.
40	Instruction CALL.
03	Pendant l'exécution du CALL dans le cas d'une interruption externe.
07	Pendant l'exécution du CALL dans le cas d'une interruption due au compteur T.
30	Instructions RETR ou RET.
06, 60	Instruction DAA.

b) Circuits de test de l'accumulateur

Il y a deux types de test pour l'accumulateur:
- test du zéro
- test d'un bit de l'accumulateur

Ces tests sont effectués pendant l'exécution des instructions de branchement suivantes:

- JZ a	1100 0110	saut à l'adresse a si A = 0.
- JNZ a	1101 0110	saut à l'adresse a si A ≠ 0.
- JBb a	bbb1 0010	saut à l'adresse a si le bit b de l'accumulateur A<b> = 1.

D'autre part, on a besoin de tester le zéro pour le registre de la RAM lors de l'exécution de l'instruction de saut DJNZ Rr,a.

- DJNZ Rr,a 1110 1rrr saut à l'adresse a si RAM(W)-1 ≠ 0, le champ rrr désigne l'adresse du registre de la RAM à tester.

Nous avons placé le circuit de test du zéro pour l'accumulateur à la sortie de l'UAL, afin de pouvoir aussi l'utiliser pour le test des registres de la RAM.

2.2.2- Plan de masse de la partie opérative

De la décomposition des instructions et de la définition précédente des différents blocs constituant la partie opérative, on peut élaborer une première architecture. On en déduit ainsi le schéma suivant:

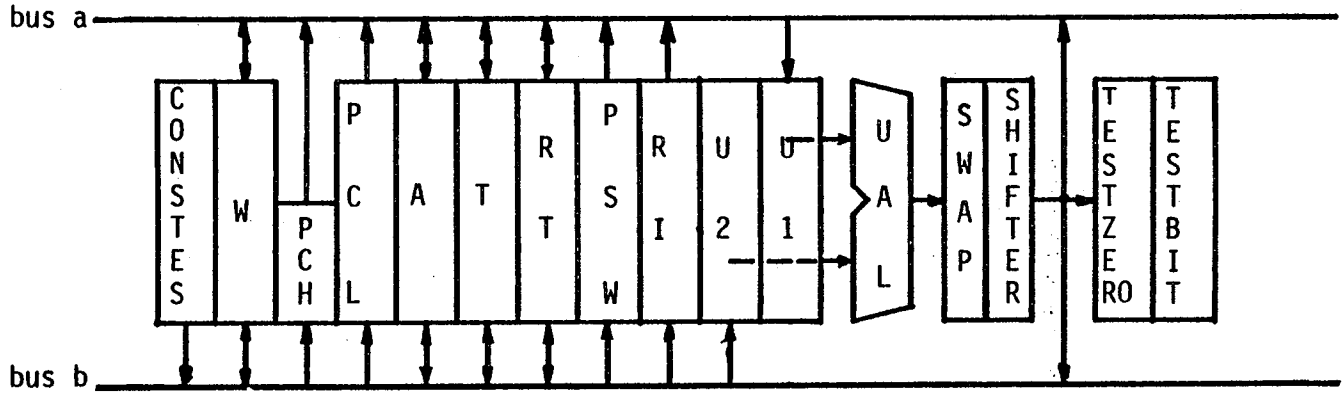


Schéma 1 : Configuration de la partie opérative

2.2.2.1- Placement des registres

Au départ, l'organisation des différentes parties fonctionnelles du micro-ordinateur a été choisi comme suit: la partie contrôle est située au-dessus de la partie opérative, la ROM au-dessous et la RAM à sa gauche. La ROM possède deux décodeurs pour que son décodage soit rapide.

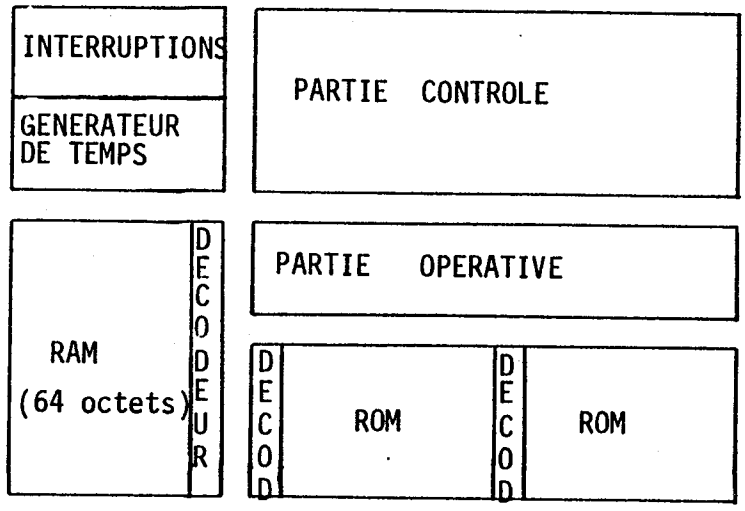


Schéma 2 : Organisation du micro-ordinateur

D'autre part, le placement du registre d'adresse W et celui du compteur de programme PCL et PCH sont assujettis respectivement à la position de la RAM et à celle de la ROM par rapport à la partie opérative. Ainsi, le registre W doit être situé le plus à gauche par rapport aux autres registres. Quant au pointeur de programme PCL, il sera au milieu de la partie opérative pour que ses sorties partent directement vers le décodeur de la ROM qui se trouve au milieu. Par contre, le décodeur de la page mémoire est situé à l'extrémité gauche de la ROM, d'où le choix du placement du PCH à coté du registre W.

Quant au registre d'état PSW, il est placé à la sortie de l'UAL du fait du rôle joué par les deux indicateurs CY et AC qui mémorisent respectivement le débordement et la retenue auxiliaire après une opération d'addition avec l'accumulateur.

2.2.2.2- Disposition des bascules

D'autre part, le registre PCH n'occupe que quatre bits de poids faible. On peut ainsi implanter quatre autres bascules dans sa partie de poids fort. Cependant, on a seulement trois indicateurs qui peuvent figurer dans la partie opérative:

- un indicateur F1 d'usage général, défini par l'utilisateur.

- un indicateur TF qui prend la valeur "1" quand le compteur T passe de la valeur FFH à 00H.
- un indicateur DBF qui permet de sélectionner entre la mémoire ROM interne et externe après exécution de l'instruction de branchement direct JMP ou celle du CALL.

Il reste néanmoins à déterminer la disposition des bascules. Pour cela, nous nous proposons de regarder de près le code opération des instructions de branchement conditionnel.

- JBba bbb1 0010 saut si le bit b de l'accumulateur A<b> = 1.
- JCa 1111 0110 saut si PSW<7> = 1.
- JNCa 1110 0110 saut si PSW<7> = 0.
- JT0a 0011 0110 saut si l'entrée T0 = 1.
- JNT0a 0010 0110 saut si l'entrée T0 = 0.
- JT1a 0101 0110 saut si l'entrée T1 = 1.
- JNT1a 0100 0110 saut si l'entrée T1 = 0.
- JF0a 1011 0110 saut si F0 = 1.
- JF1a 0111 0110 saut si F1 = 1.
- JTFa 0001 0110 saut si TF = 1.
- JNIa 1000 0110 saut si l'entrée $\overline{INT}$ = 0.

Ces instructions de branchement permettent de tester chaque indicateur. En examinant par exemple le code opération de l'instruction JBba, on remarque que ses trois bits de poids fort servent à déterminer le bit de l'accumulateur à tester. Afin de pouvoir tester les indicateurs avec le même circuit de test, il est intéressant de les placer selon leur code opération.

Exemple :
- JF1a 0111 0110 la bascule F1 doit être placée au bit 3.
- JTFa 0001 0110 la bascule TF doit être placée au bit 0.

De plus, cet indicateur TF passe à la valeur "1" lors du débordement du compteur T pendant son incrémentation. Ainsi, il est astucieux de placer cette bascule proche de l'UAL afin d'y stocker la retenue. D'où l'idée de diviser le registre d'état PSW

en deux parties PSH et PSL afin d'avoir la disposition suivante des bascules (voir schéma 3). La bascule T.OVF a été implantée dans la partie opérative, permettant ainsi de combler le vide. Elle permet, comme TF, de mémoriser le débordement du compteur T. Sa sortie est dirigée vers la bascule de demande d'interruption TI RQ (voir fig.4.4).

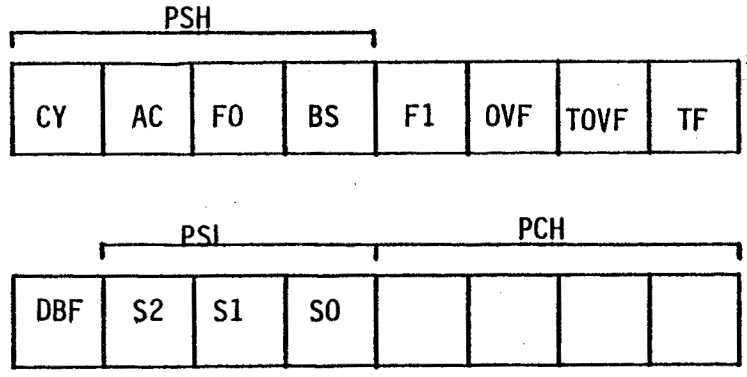


Schéma 3 : Disposition des bascules

2.2.2.3- Architecture de la partie opérative

L'architecture de la partie opérative sera constituée donc :

- d'un bloc arithmétique et logique formé d'une U.A.L. (8 bits) avec deux registres tampons (U1 et U2) à son entrée, d'un permuteur ("SWAPER"), d'un décaleur ("SHIFTER") et des circuits de test pour la correction décimale ("DAA") et pour le zéro d'un registre ou d'un bit.
- de registres tels que le registre d'adresse (W) de la RAM (8 bits), le compteur de programme divisé en deux parties: PCL (8 bits) et PCH (4 bits de poids fort), l'accumulateur A, le compteur T, le registre temporaire RT, le registre d'instruction RI et le registre d'état sectionné en deux parties: PSH (4 bits de poids fort) et PSL (3 bits du pointeur de pile).
- de basculès occupant l'espace vide créé par le PCH qui est de 4 bits seulement (voir 2.2.2.2).

Afin d'optimiser les connexions des registres aux bus, nous avons répertorié tous les transferts parallèles qui se passent pendant une même phase. Par ailleurs, nous nous sommes imposés les contraintes suivantes :

- les registres tampons U1 et U2 sont respectivement connectés au bus a et au bus b (en écriture seulement).
- la RAM est connectée au bus a (lecture et écriture sur le même bus).
- la ROM est lue sur le bus b.

En tenant compte de ces considérations, seulement les connexions des registres PCL, PCH, PSW et RI aux bus ont pu être optimisées. Leur écriture se fait par le bus b et leur écriture sur le bus a (voir fig. 2.4). Les constantes sont aussi lues sur le bus b. La lecture et l'écriture des autres registres (W, RT, A et T) se fait par les deux bus a et b. La sortie de l'U.A.L. est connectée aux deux bus.

Remarque : Les ports d'entrées-sorties ne seront pas implantés dans la partie opérative, mais à côté de leurs broches respectives. Le bus externe g, connecté au bus b, contourne le circuit des blocs fonctionnels du micro-ordinateur pour desservir les entrées-sorties.

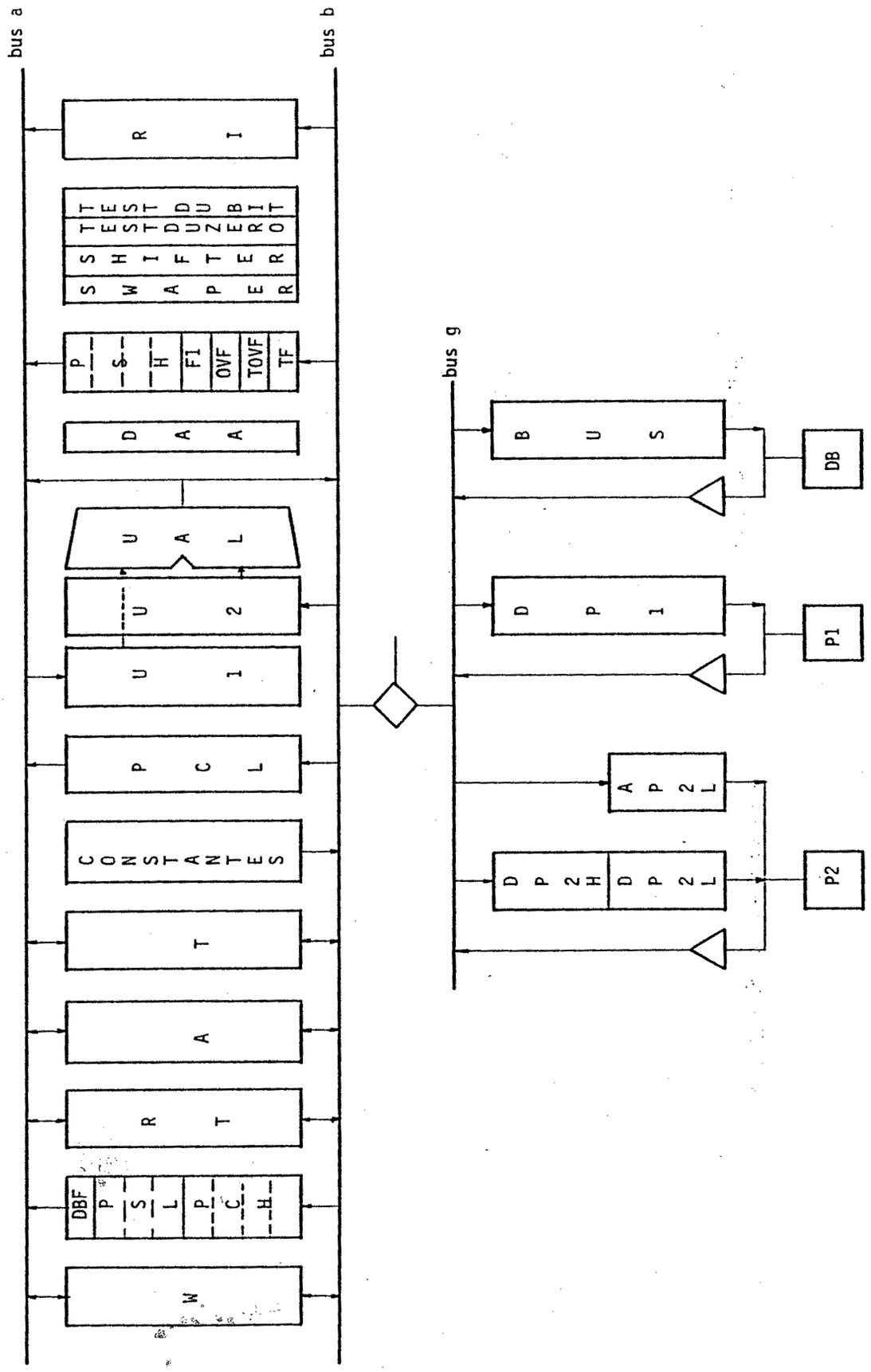


Fig. 2.4- Architecture de la partie opérative

- CHAPITRE 3 : REALISATION DE LA PARTIE OPERATIVE :

3.1- SCHEMAS DES DIFFERENTES PARTIES DE LA P.O.

3.1.1- Schéma d'un registre quelconque de la P.O.

3.1.2- Schéma des registres d'entrée de l'U.A.L. (U1 et U2)

3.1.3- Schéma de l'U.A.L.

3.1.3.1- Définition des opérations de l'U.A.L.

a) Réalisation de la disjonction

b) Réalisation du "OU"

c) Réalisation du "ET"

d) Réalisation de l'addition

3.1.4- Réalisation du permuteur

3.1.5- Réalisation des rotations à droite et à gauche

3.1.6- Schéma des bascules

3.1.6.1- Bascule TF

3.1.6.2- Bascule CY

3.1.6.3- Bascule AC

3.1.6.4- Bascule OVF

3.1.7- Réalisation de l'ajustement décimal

3.1.8- Circuits de test de l'accumulateur

3.1.9- Génération des constantes

3.2- IMPLANTATION EN CMOS

3.2.1- La méthode classique

3.2.1.1- Exemple d'implantation d'un inverseur

3.2.1.2- Evaluation des pas de la grille

a) Pas d'aluminium (PA)

b) Pas de polysilicium (PP)

3.2.1.3- Exemples d'implantation sur grille

a) Portes simples

b) Schéma de l'U.A.L.

3.2.2- Autre méthode d'implantation

3.2.2.1- Exemple d'implantation

a) Portes simples

b) Schéma de l'U.A.L.

3.2.2.2- Assemblage de la partie opérative

a) Présentation du système LUBRICK

b) Résultats de l'implantation

3.2.2.3- Calcul du temps de propagation dans le bus en polysilicium

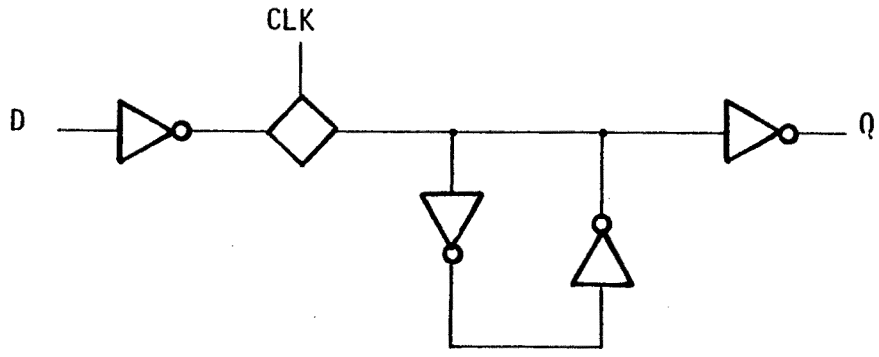
CHAPITRE 3

- REALISATION DE LA PARTIE OPERATIVE -

3.1- SCHEMAS DES DIFFERENTES PARTIES DE LA P.O.

3.1.1- Schéma d'un registre quelconque de la P.O.

On utilise le point mémoire suivant:



L'écriture dans ce point mémoire se fait par l'intermédiaire de la porte de transmission pilotée par l'horloge CLK. En réalité, une porte de transmission en CMOS est réalisée par deux transistors N et P en parallèle afin d'éviter la perte d'un seuil. On aura donc besoin de deux commandes complémentaires. Mais dans notre cas, on peut utiliser un seul transistor, de type N, puisqu'on a un inverseur qui rétablira le niveau logique par la suite.

L'entrée D de ce point mémoire sera connectée à un bus de la partie opérative et la sortie Q à un autre bus, pour former un registre. Cependant, l'inverseur de sortie est remplacé par une porte trois états pour ne lire sur un bus qu'un registre à la fois.
On a besoin de deux commandes complémentées RREG et $\overline{\text{RREG}}$ pour sa lecture. Son écriture se fait à l'aide de la porte de

transmission à l'entrée, pilotée par une commande WREG.
On obtient ainsi le schéma suivant pour un registre REG quelconque.

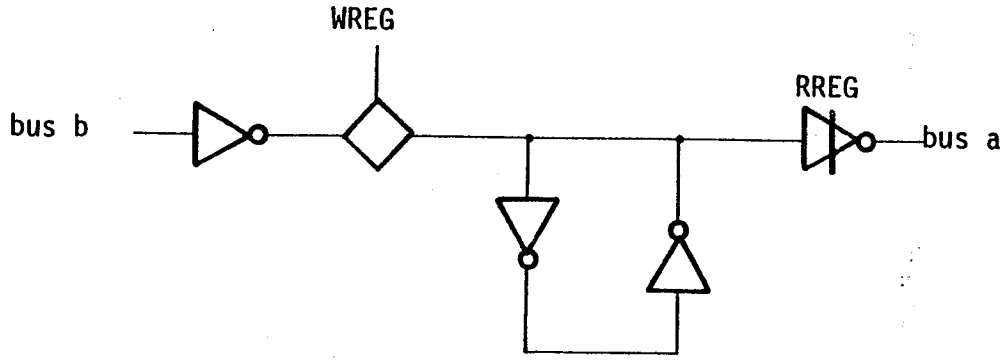


Fig. 3.1- Circuiterie d'un registre de la P.O.

WREG (W comme Write): Cette commande permet l'écriture dans le registre REG lorsqu'elle est à "1".

RREG (R comme Read) : Elle permet la lecture du registre REG lorsqu'elle est à "1". Cela implique que la commande $\overline{\text{RREG}}$ est à "0".

Dans le cas où le registre est connecté aux deux bus de la partie opérative en écriture et en lecture, l'entrée sera multiplexée ainsi que la sortie. On aura le schéma suivant:

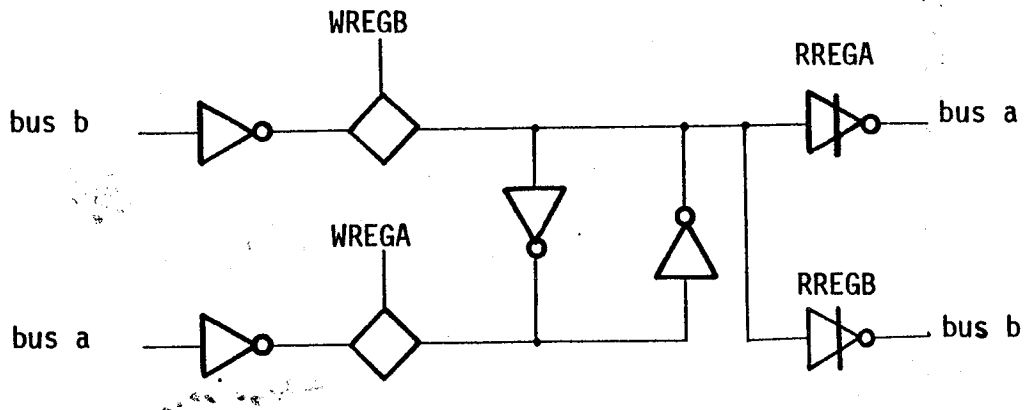


Fig. 3.2- Registre connecté aux 2 bus de la P.O.

3.1.2- Schéma des registres d'entrée de l'U.A.L. (U1 et U2)

Le registre U1 est relié uniquement au bus a. Son écriture se fait donc par l'intermédiaire de ce bus grâce à la commande WU1.

Pour certaines opérations arithmétiques, ce registre est remis à zéro par la commande $\overline{RES1}$ qui est en réalité dédoublée. En effet, on a vu qu'on a besoin de remettre à zéro uniquement les 4 bits de poids faible et le bit 7 de U1 pendant les instructions de sous-routine (CALL, RETR et RET) en (S3,T1) (voir 2.1.4.1). On aura alors:

- une commande $\overline{RES1L}$ pour les 4 bits de poids faible et le bit 7 de U1.
- une autre commande $\overline{RES1H}$ pour les bits 6 à 4 de U1.

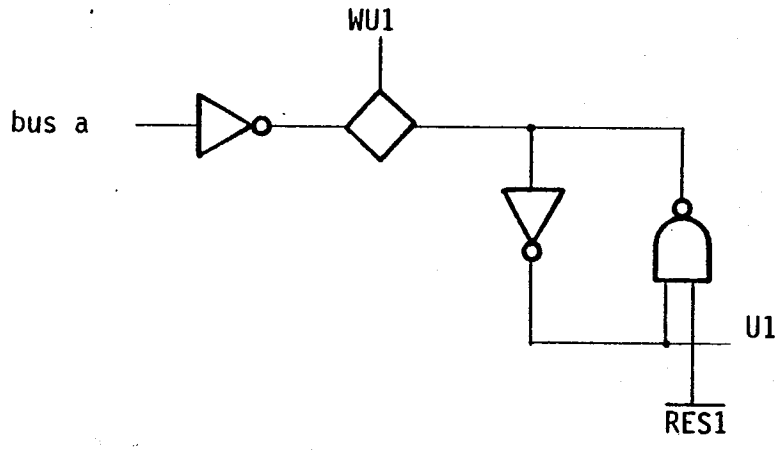


Fig. 3.3- Registre tampon U1

RES1	U1
1	0
0	x

- Mise à zéro du registre U1 quelque soit le positionnement de WU1.
- Mémorisation de l'ancien contenu si WU1 = 0
- Lecture du bus a si WU1 = 1.

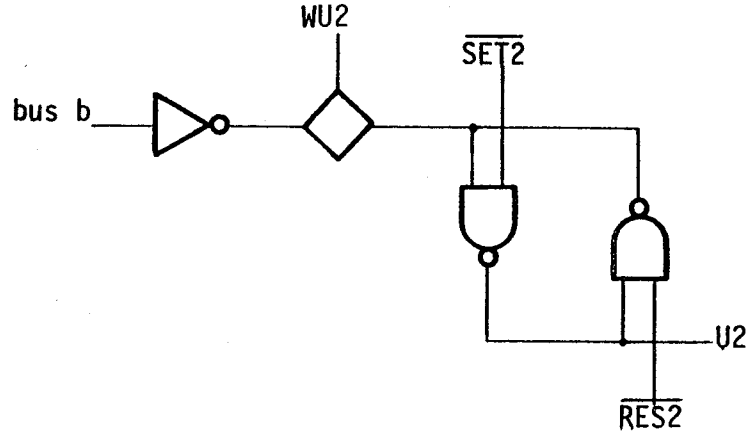


Fig. 3.4- Registre tampon U2

Le registre U2 est réalisé à l'aide d'une bascule RS reliée au bus b par l'intermédiaire d'une porte de transmission pilotée par la commande d'écriture WU2.
Ainsi, la constante hexadécimale FF nécessaire pour décrémenter l'opérande U1 ou calculer son complément (voir 3.1.3.1) sera générée par action de la commande $\overline{SET2}$.

RES2	SET2	U2
0	0	x
x	1	FF
1	0	00

- Mémorisation de l'ancienne valeur de U2 si WU2 = 0, sinon lecture du bus b.

Remarque : Les sorties des deux registres U1 et U2 sont

connectées directement à l'entrée de l'U.A.L.

3.1.3- Schéma de l'U.A.L.

3.1.3.1- Définition des opérations de l'U.A.L.

L'unité arithmétique et logique du 8048 devra nous permettre de faire l'addition, la disjonction, le "ET" et le "OU" logique, l'incrémentation, la décrémentation et le complément (voir 1.1.1).

Cependant, on remarque que la décrémentation d'un opérande U1 revient à faire son addition avec la constante FFH, d'où:

$$U1 - 1 = U1 + FFH$$

Quant au complément, il est obtenu à partir de la disjonction avec la constante FFH, d'où:

$$U1 = U1 \oplus FFH$$

Dans notre cas, l'UAL sera simple à réaliser puisqu'elle ne doit effectuer en réalité que quatre opérations élémentaires entre les opérandes d'entrée U1 et U2.

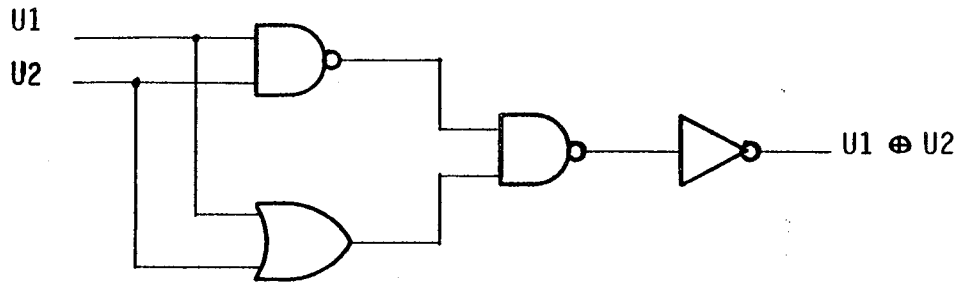
- disjonction,
- "ET" logique,
- "OU" logique,
- addition.

a) Réalisation de la disjonction

$$\overline{U1 \oplus U2} = \overline{U1}.\overline{U2} + U1.U2$$

$$= (\overline{U1+U2}) + U1.U2$$

D'où: $$\overline{U1 \oplus U2} = (U1+U2) \cdot \overline{(U1.U2)}$$

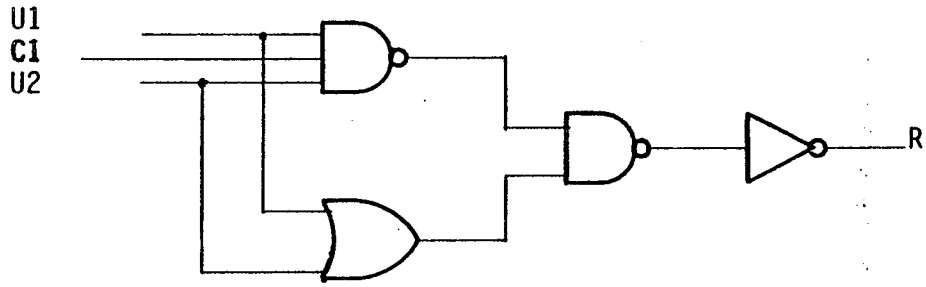


Remarque :

- En CMOS, il est plutôt intéressant de travailler en portes "NAND", à cause de la faible mobilité des trous.
- L'inverseur en sortie de porte est intéressant, comme nous le verrons plus tard.

b) Réalisation du "OU"

A partir du schéma précédent de la disjonction, on peut réaliser le "OU" en "tuant" le "NON-ET" à l'aide d'une commande C1.



C1 = 0 ===> R = U1 + U2 (obtention du "OU")

C1 = 1 ===> R = U1 ⊕ U2

c) Réalisation du "ET"

De même, on fait agir une commande C2 sur le "OU" et on obtient ainsi le "ET". D'où on obtient le schéma 1, permettant de réaliser la disjonction, le "ET" et le "OU" à l'aide des commandes C1 et C2.

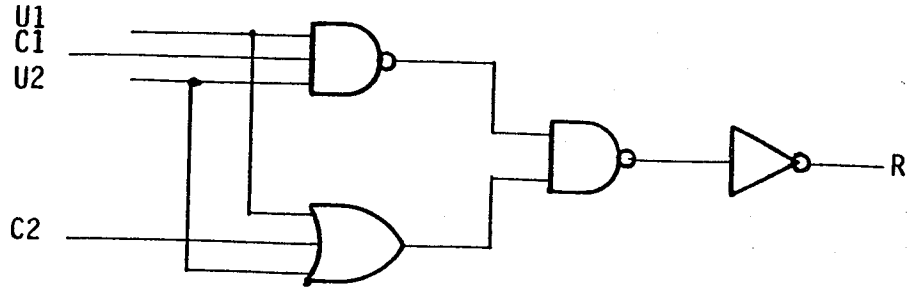


Schéma 1 : Réalisation de la disjonction, "OU", "ET"

C1	C2	R
0	0	U1+U2
1	0	U1⊕U2
1	1	$\overline{U1.U2}$

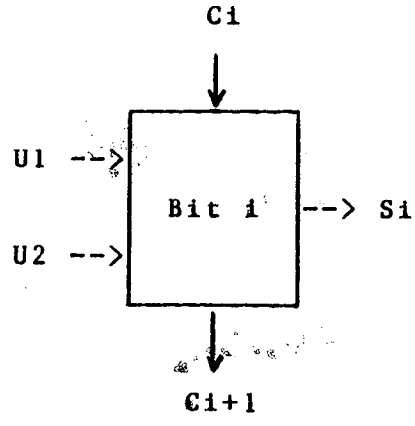
Remarque : on obtient le "ET" complémenté.

d) Réalisation de l'addition

L'addition binaire est obtenue à l'aide des relations suivantes:

$$Si = (U1+U2)i \oplus Ci \qquad (1)$$

$$Ci+1 = (U1.U2)i + Ci.(U1 \oplus U2)i \qquad (2)$$



Si est le résultat de l'addition du bit i.
Ci et Ci+1 désignent respectivement la retenue entrante et la retenue sortante. Elles seront notées par cin et cout.

L'équation (1) peut s'écrire:

$Si = Ri \oplus cin$

où $Ri = (U1+U2)i$ est obtenu par le schéma 1 en faisant C1 = 1 et C2 = 0.

Donc le résultat de l'addition sera la disjonction entre la retenue entrante et la sortie du schéma 1.

En CMOS, la disjonction est simple à réaliser. Elle se fait à l'aide de quatre transistors pour peu que l'on dispose d'une entrée complémentée.

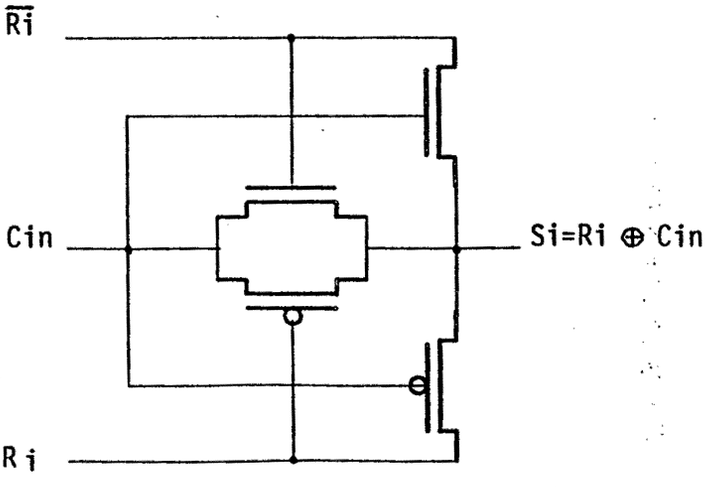


Schéma 2 : Disjonction en CMOS

Calcul de la retenue

L'équation (2) peut s'écrire:

$cout = G + P.cin$

où G désigne le terme de génération de la retenue et P celui de la propagation de la retenue, avec:

$G = U1.U2$

$P = U1 \oplus U2$

U1	U2	G	P	cout	$\overline{cout}$
0	0	0	0	0	1
0	1	0	1	cin	$\overline{cin}$
1	0	0	1	cin	$\overline{cin}$
1	1	1	0	1	0

On constate que lorsque P = 1, alors cout = cin. Autrement dit, lorsque U1 et U2 sont différents, alors la retenue entrante est propagée.
D'autre part, lorsque P = 0, alors $\overline{cout} = \overline{G}$. C'est-à-dire que la retenue est générée seulement quand U1 et U2 sont identiques. On en déduit donc le schéma 3 qui réalise l'équation (2).

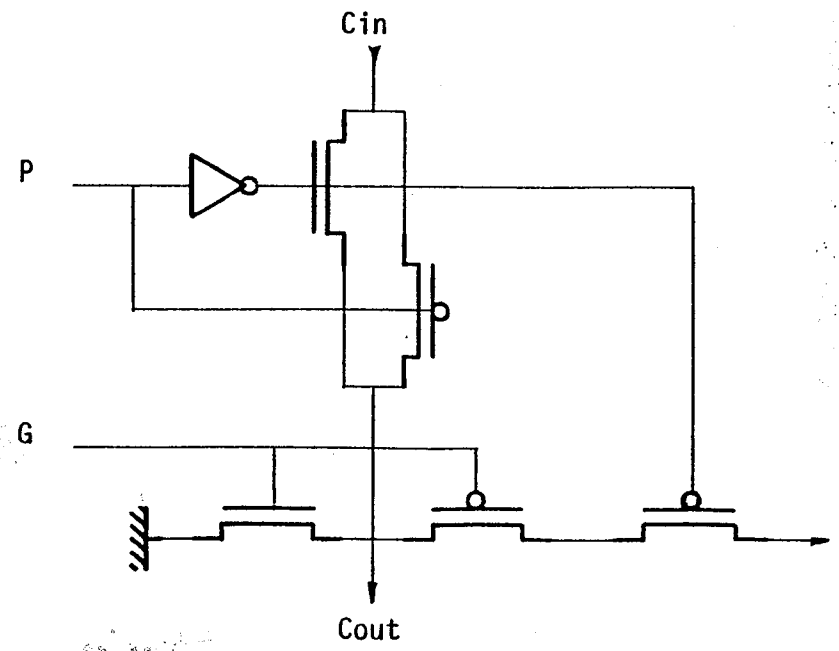
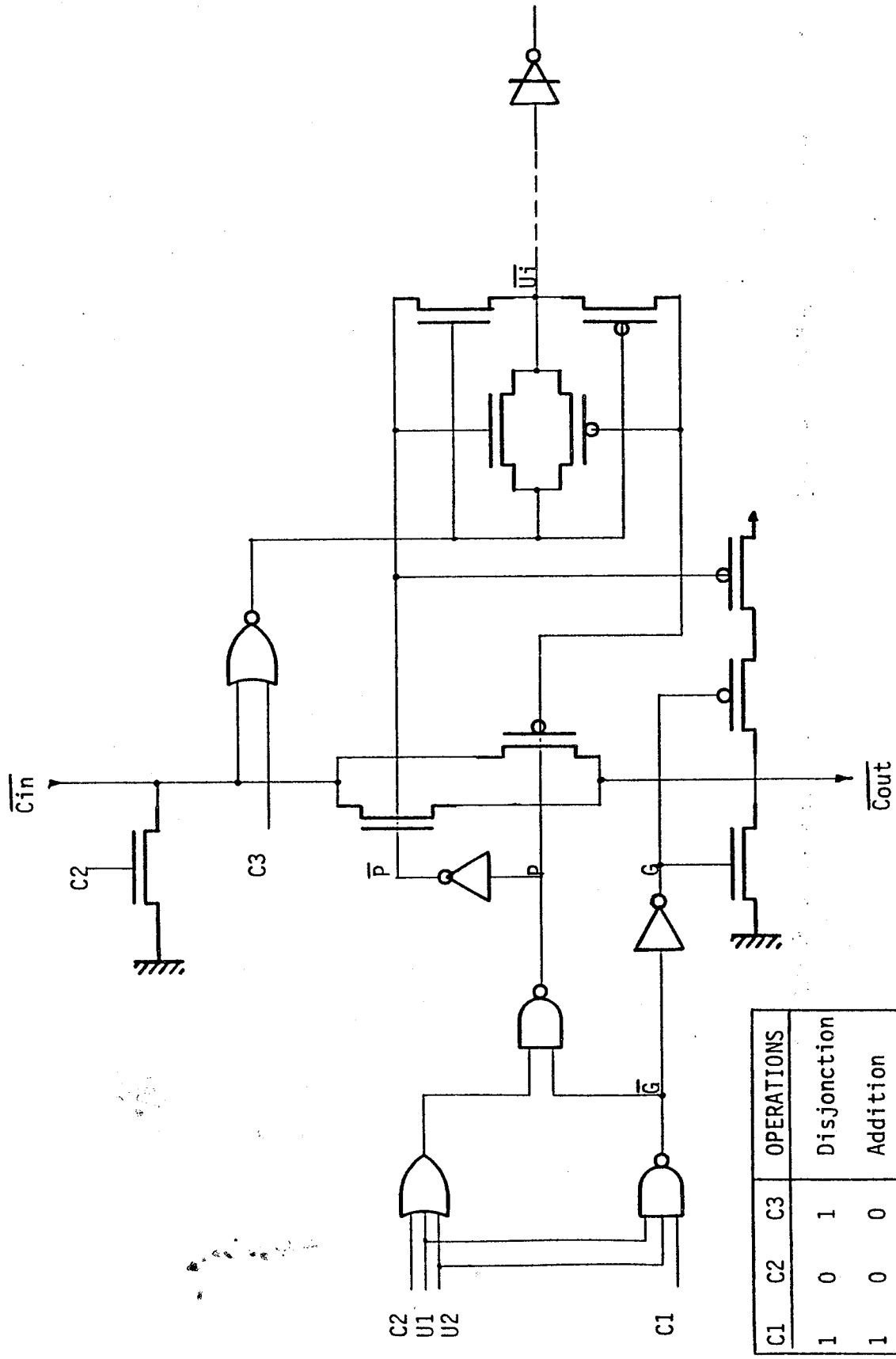


Schéma 3 : Réalisation de la ligne de retenue

Les termes P et G sont réalisés par le schéma 1. D'où, en assemblant les schémas 1, 2 et 3, on obtient notre UAL représentée par la figure 3.5. Cette unité arithmétique et



C1	C2	C3	OPERATIONS
1	0	1	Disjonction
1	0	0	Addition
1	1	0	"ET"
0	0	1	"OU"

Fig. 3.5- Schéma de l'U.A.L.

logique permet de réaliser les quatre opérations élémentaires: la disjonction, l'addition, le "ET" et le "OU".

Remarques :

La sortie Ui de l'UAL est complémentée de façon à utiliser un amplificateur trois états d'attaque du bus pour inverser le résultat.

De même, la ligne de la retenue est complémentée.

La commande C3 permet d'inhiber la retenue pendant la disjonction et le "OU". C'est-à-dire que l'on met la commande C3 à "1" pendant l'exécution de ces deux opérations. Pour les deux autres opérations ("ET", addition) cette commande est à "0".

Nous avons vu que le schéma 1 nous permettait d'obtenir la disjonction, le "OU" et le "ET" complémenté. Il faudra donc inverser le résultat pendant l'opération du "ET". Ceci est réalisé à l'aide d'un circuit faisant la disjonction entre "1" et la sortie. Le "1" est obtenu grâce à la porte "NOR" dont ses entrées, la commande C3 et la ligne de retenue, sont mises à zéro. La commande C2 permet de tirer à la masse la ligne de la retenue.

3.1.4- Réalisation du permuteur

Un circuit spécial est conçu, à la sortie de l'UAL, pour l'exécution de l'instruction SWAP A (A <7:4> $\rightleftharpoons$ A <3:0>).
Cette permutation est effectuée en un seul coup pendant une phase.

La figure 3.6 représente le circuit réalisant la permutation ("swap") entre les bits de poids fort et ceux de poids faible. Il est constitué de quatre portes de passage pilotées par une seule commande SW et son complément.

SW = 1 Permutation

SW = 0 Pas de permutation.

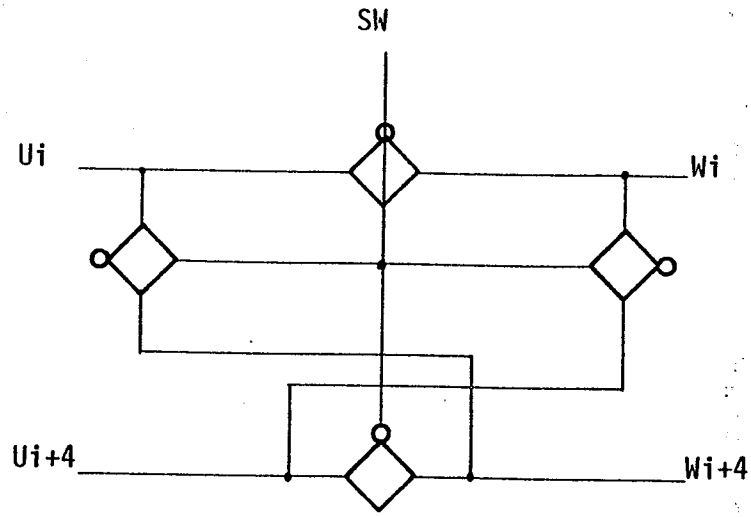


Fig. 3.6- Réalisation du permuteur

3.1.5- Réalisation des rotations à droite et à gauche

Le décaleur ("SHIFTER") est placé à la sortie du permuteur.
Les décalages à droite et à gauche se font à l'aide de portes de passage pilotées par la commande SL et son complément (voir schéma 4).

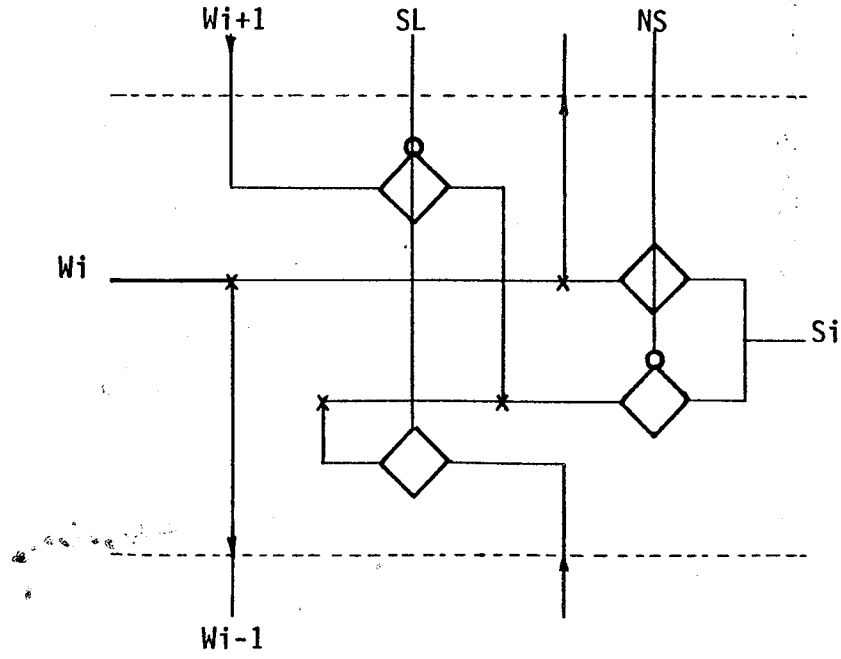


Schéma 4 : Réalisation du décalage à droite et à gauche

SL = 1, Décalage à gauche.
SL = 0, Décalage à droite.

La commande NS permet d'annuler le décalage effectué précédemment.
NS = 1, Pas de décalage (transparence du décaleur : Si = Wi).
NS = 0, Prise en compte du décalage précédent.

Pour la réalisation des rotations, les bits 0 à 7 sont un peu particuliers. Ecrivons d'abord les quatre instructions permettant d'effectuer les rotations à droite et à gauche, avec ou sans retenue.

- Instruction RLA (rotation à gauche sans retenue)
  A<n+1> <--- A<n> avec n ∈ (0,6)
  A<0> <--- A<7>

- Instruction RLCA (rotation à gauche avec retenue)
  A<n+1> <--- A<n>
  A<0> <--- PSW<7>
  PSW<7> <--- A<7>

- Instruction RRA (rotation à droite sans retenue)
  A<n> <--- A<n+1>
  A<7> <--- A<0>

- Instruction RRCA (rotation à droite avec retenue)
  A<n> <--- A<n+1>
  A<7> <--- PSW<7>
  PSW<7> <--- A<0>

On remarque que le bit 0 (A<0>) peut recevoir soit le bit 7 (A<7>) ou la retenue PSW<7>, d'après les instructions RLA et RLCA.
De même pour le bit 7 (A<7>), il peut recevoir soit le bit 0 (A<0>) ou la retenue PSW<7>, d'après les instructions RRA et RRCA.

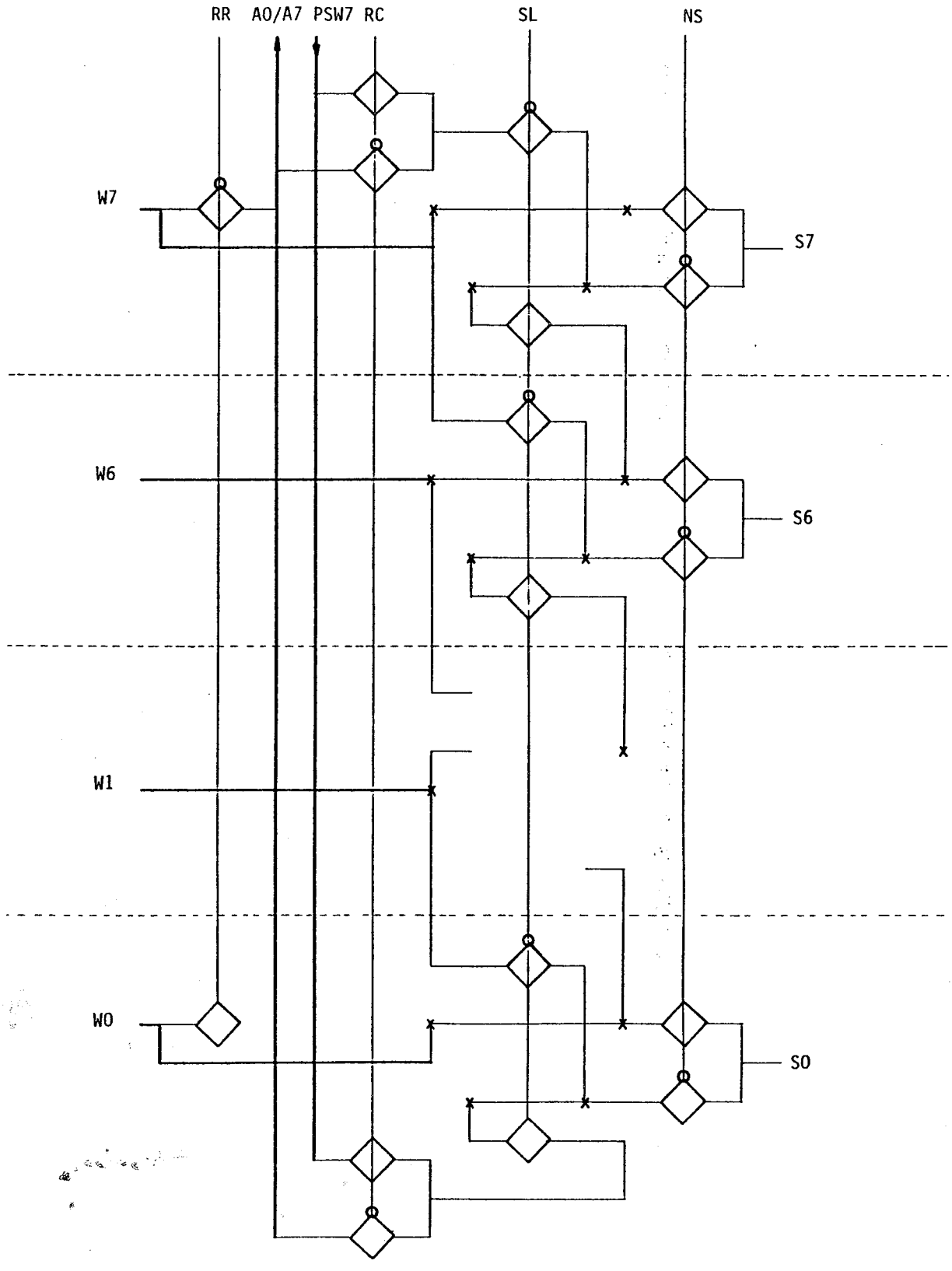


Fig. 3.7- Rotations à droite et à gauche

La sélection des entrées aux bits 0 ou 7 se fait à l'aide d'un multiplexeur piloté par la commande RC. Cette dernière permet de choisir entre les rotations avec retenue et sans retenue.

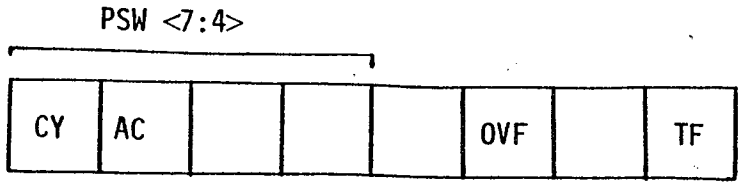
D'autre part, pour les rotations avec retenue, on remarque que la retenue est changée. Le PSW<7> reçoit soit le bit 0 ou le bit 7 de l'accumulateur.
La ligne A0/A7 alimente le PSW<7> pendant la rotation avec retenue.
La commande RR permet de faire la sélection entre la rotation à droite et celle à gauche.
La figure 3.7 représente le décaleur ("SHIFTER") pour faire les rotations à droite et à gauche. Il est situé à la sortie de l'UAL.

Instructions	Commandes			
	RR	RC	SL	NS
RLA	0	0	1	0
RLCA	0	1	1	0
RRA	1	0	0	0
RRCA	1	1	0	0

Remarque : Pour rendre le décaleur transparent (ne pas faire de décalage), il suffit de mettre la commande NS à "1",indépendamment des autres commandes.

3.1.6- Schéma des bascules

Tous les indicateurs s'implantent de la même manière qu'un point registre (voir 3.1.1), à l'exception des quatre bascules suivantes qui sont un peu particulières: TF, CY, AC et OVF.



3.1.6.1- Bascule TF

Elle est formée d'une bascule RS.

Une fois cette bascule à "1" (résultant du débordement du compteur T), elle doit rester dans cet état. Cet indicateur n'est remis à "0" qu'après le "RESET" ou l'exécution de l'instruction JTF.

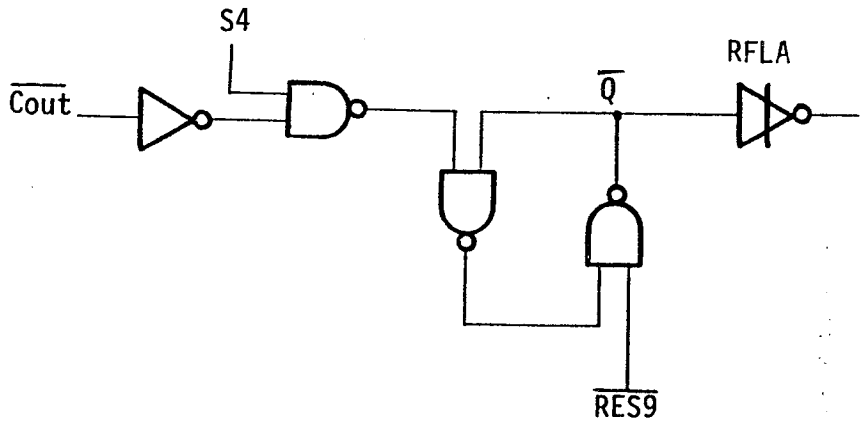


Fig. 3.8- Schéma de la bascule TF

Cette bascule est remise à zéro que par la commande $\overline{RES9}$.

$RES9 = 1 \Longrightarrow \overline{Q} = 1$

Lorsque cette commande passe à "0", alors la bascule mémorise l'ancienne valeur. La bascule n'est mise à "1" que lorsque la retenue sortante est égale à "1", soit $\overline{cout} = 0$.

En effet:

$\overline{cout} = 0 \Longrightarrow \overline{Q} = 0$ (Mise à "1" de la bascule).

$\overline{cout} = 1 \Longrightarrow$ Mémorisation de l'ancienne valeur.

Ainsi, uniquement après l'excécution de l'instruction JTF ou lors

d'un "RESET" qu'on actionne la commande RES9 à "1", ce qui permet de remettre cet indicateur à "0".

3.1.6.2- Bascule CY

Cette bascule sert à mémoriser la retenue résultante d'une addition de l'accumulateur avec un opérande.

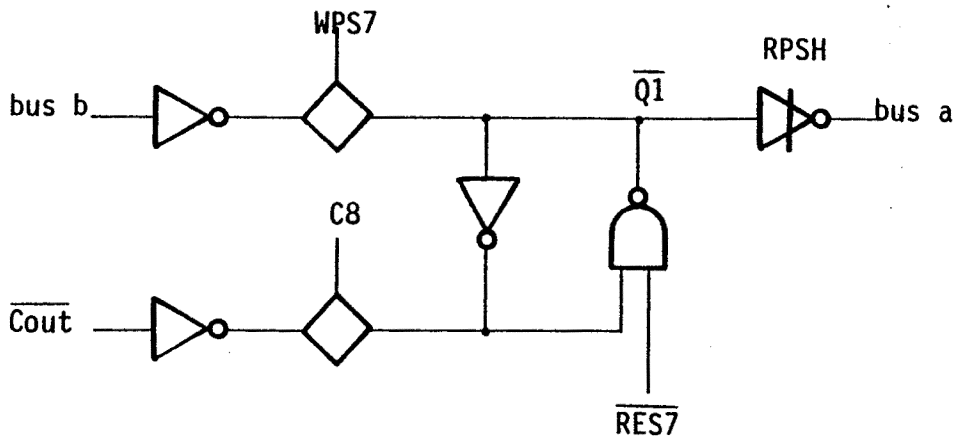


Fig. 3.9- Schéma de la bascule CY (ou PSW<7>)

On doit aussi pouvoir écrire dans cette bascule par l'intermédiaire du bus b, en se servant de la commande WPS7 pendant l'instruction MOV PSW,A.
Elle peut être lue sur le bus a, par l'action de la commande RPSH pendant l'instruction MOV A,PSW.
La commande $\overline{RES7}$ permet la remise à zéro de cet indicateur, soit $\overline{Q1}$ = 1, après l'addition avec retenue de l'accumulateur et d'un opérande (instruction ADDC A,Rr et ADDC A,aRr).
La commande C8 autorise le chargement de la retenue sortante $\overline{cout}$ en (S5,T2) pendant les instructions d'addition de l'accumulateur (ADD A,Rr et ADD A,aRr).

3.1.6.3- Bascule AC

Cette bascule joue le même rôle que la précédente bien qu'elle sert à mémoriser la retenue auxiliaire AC pendant les instructions d'addition de l'accumulateur. La seule différence est que cet indicateur ne posséde pas de commande de remise à

zéro.
Son écriture par le bus b se fait à l'aide de la commande WPS6.

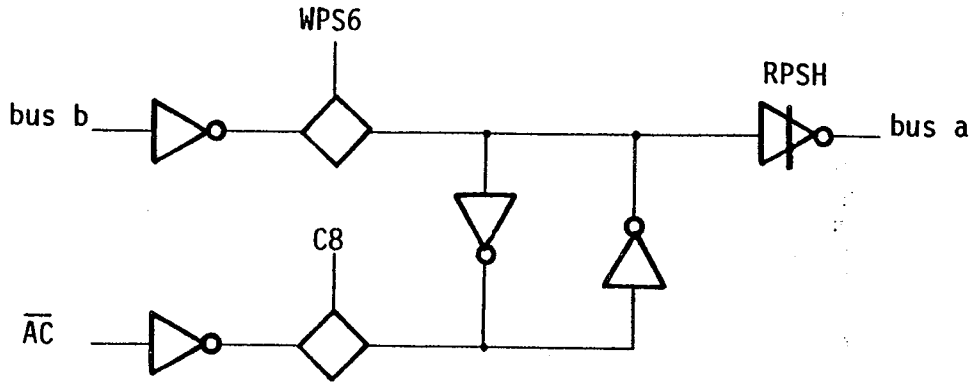


Fig. 3.10- Schéma de la bascule AC (ou PSW<6>)

3.1.6.4- Bascule OVF

Cette bascule est complexe puisqu'elle assure plusieurs fonctions:

- Mémorisation de la retenue sortante cout résultante de l'incrémentation du PCL,
- Mémorisation du bit 7 ou du bit 0 de l'accumulateur (A0/7) pendant l'exécution de l'instruction RLC A ou RRC A en (S3,T2).

Exemple : Instruction RLC A

	T1	T2
(S3)	U1 <--- a:= A; cin:= 0	A <--- b:= SL (U1+U2) A<0>:= PSW<7> OVF:= A<7>
(S3)	(Incrémentation éventuelle du compteur T)	
(S5)	U1 <--- a:= PSH; cin:= 0	PSW<7> <--- b:= SW (U1+U2)

- Génération de la retenue entrante cin (cin = 0, 1 ou OVF)

pendant les opérations à travers l'U.A.L.

- Cette bascule doit être mise à "1" lors de la lecture du PSW par l'instruction MOV A,PSW.

La bascule OVF sera de type RS dont l'entrée est multiplexée. La commande C10 permet de mémoriser le débordement du PCL lors de son incrémentation en (S1,T2) de toutes les instructions. La commande C9 est activée pendant les instructions de rotation avec retenue RLC A et RRC A en (S3,T2) pour la sauvegarde du bit 0 et du bit 7 de l'accumulateur dans cette bascule.

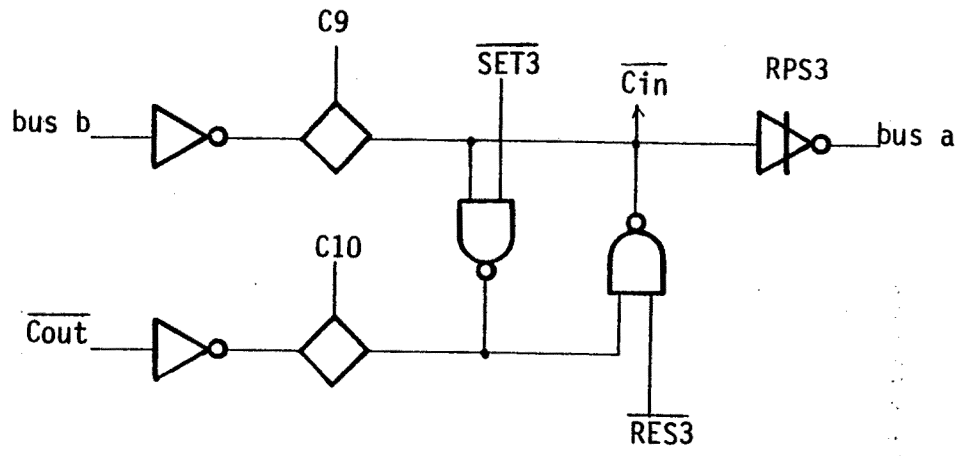


Fig. 3.11- Schéma de la bascule OVF

RES3	SET3	OVF	
0	0	x	- Mémorisation de l'ancienne valeur
x	1	1	- (cin=0)
1	0	0	- (cin=1, utilisé pendant l'incrémentation d'un opérande).

3.1.7- Réalisation de l'ajustement décimal

Pendant l'instruction DA A, on teste d'abord si la retenue auxiliaire AC est à "1" ou si la partie de poids faible de

l'accumulateur (A<3:0>) est supérieure à 9.
Ensuite, on teste la retenue CY (PSW<7>) à "1" ou si la partie de poids fort de l'accumulateur, soit A<7:4> est supérieure à 9.

A<3:0> >9 ===> A<3:0> = (1010 ou 1011)
ou (1100 ou 1101)
ou (1110 ou 1111)

On constate que si la partie A<3:0> est supérieure à 9, alors le bit 3 est égal à "1" et l'un des bits 1 ou 2 est égal à "1". Ce qui peut s'exprimer par:

A<3:0> > 9 ===> A<3> (A<2> + A<1>) = 1
De même, A<7:4> > 9 ===> A<7> (A<6> + A<5>) = 1

Ce qui revient à faire les deux tests suivants pendant la correction décimale. On aura deux comptes-rendus pour la partie contrôle:

1°) on teste si: PSW<6> + A<3> (A<2> + A<1>) = 1
2°) on teste ensuite si: PSW<7> + A<7> (A<6> + A<5>) = 1.

D'où le schéma suivant:

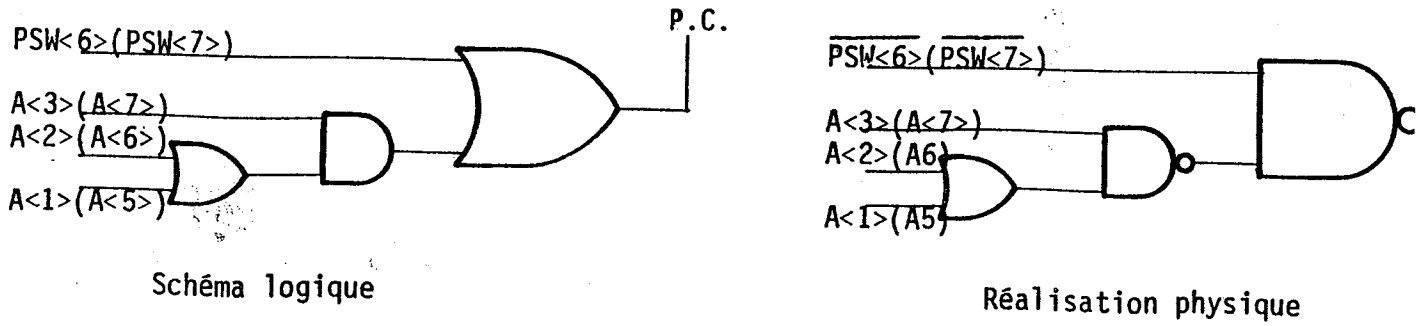


Fig. 3.12- Circuit de test pour la correction décimale

3.1.8- Circuits de test de l'accumulateur

Il y a deux tests à réaliser sur l'accumulateur:

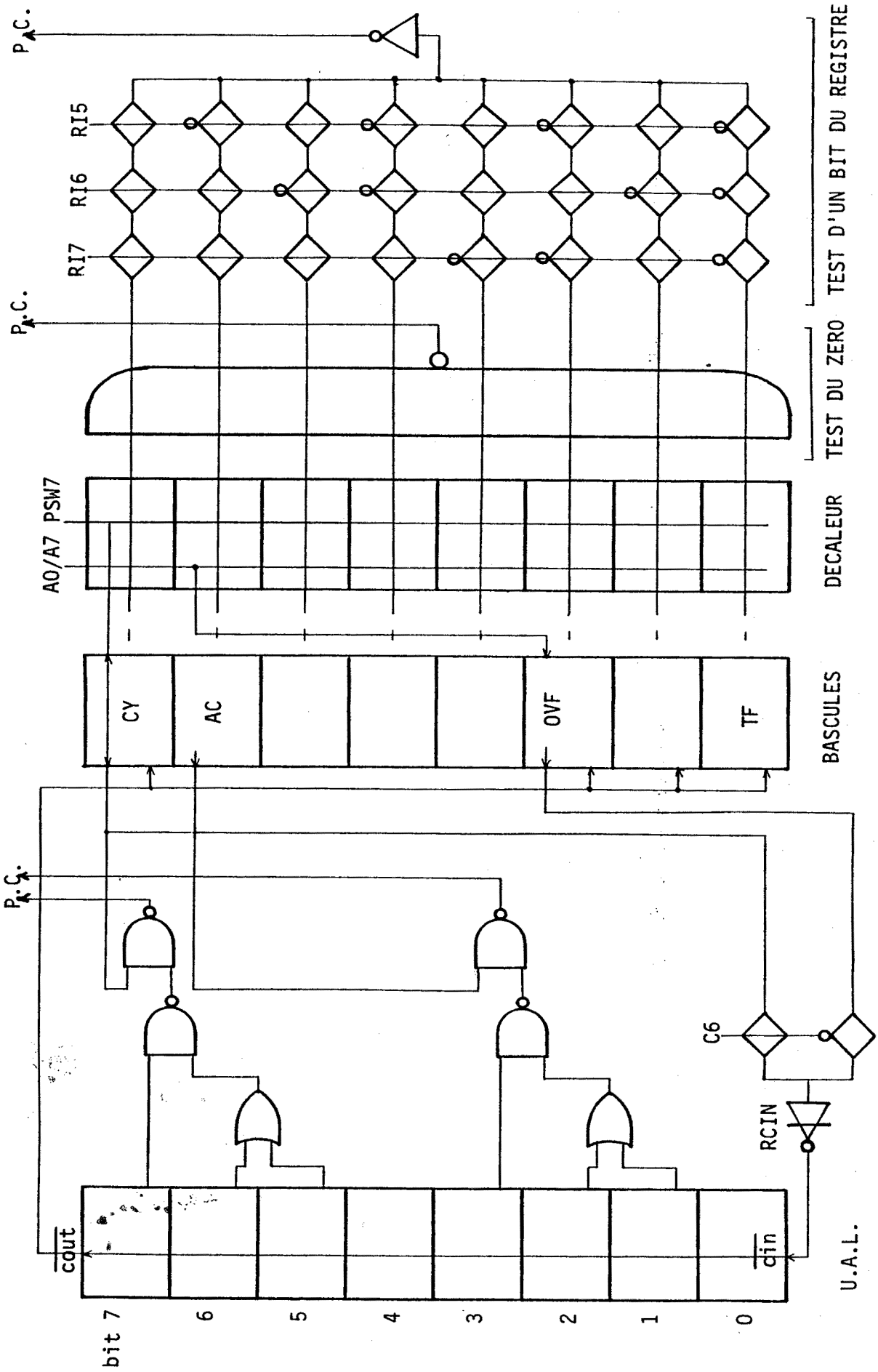


Fig. 3.13- Configuration de l'U.A.L., des bascules et des circuits de test.

- test du zéro,
- test d'un bit.

Le premier est réalisé par une porte "NAND" et le second par un multiplexeur permettant de sélectionner le bit à tester selon le code opération. Ces deux circuits de test sont situées à la sortie du "SHIFTER" et leur compte-rendu est dirigé vers la partie contrôle (voir fig. 3.13).

3.1.9- Génération des constantes

Les constantes sont au nombre de dix (voir 2.2.1.3,a). On peut les générer à l'aide d'un PLA statique formé de portes "NAND". Cette réalisation s'implante facilement dans la partie opérative. Ces constantes ont été codées sur quatre bits.

Commandes A	B	C	D	Génération des constantes
1	1	1	1	00
1	0	0	0	18
0	0	0	0	0C
0	0	0	1	14
1	0	1	0	40
0	1	1	0	03
0	0	1	0	07
1	0	0	1	30
0	0	1	1	06
1	0	1	1	60

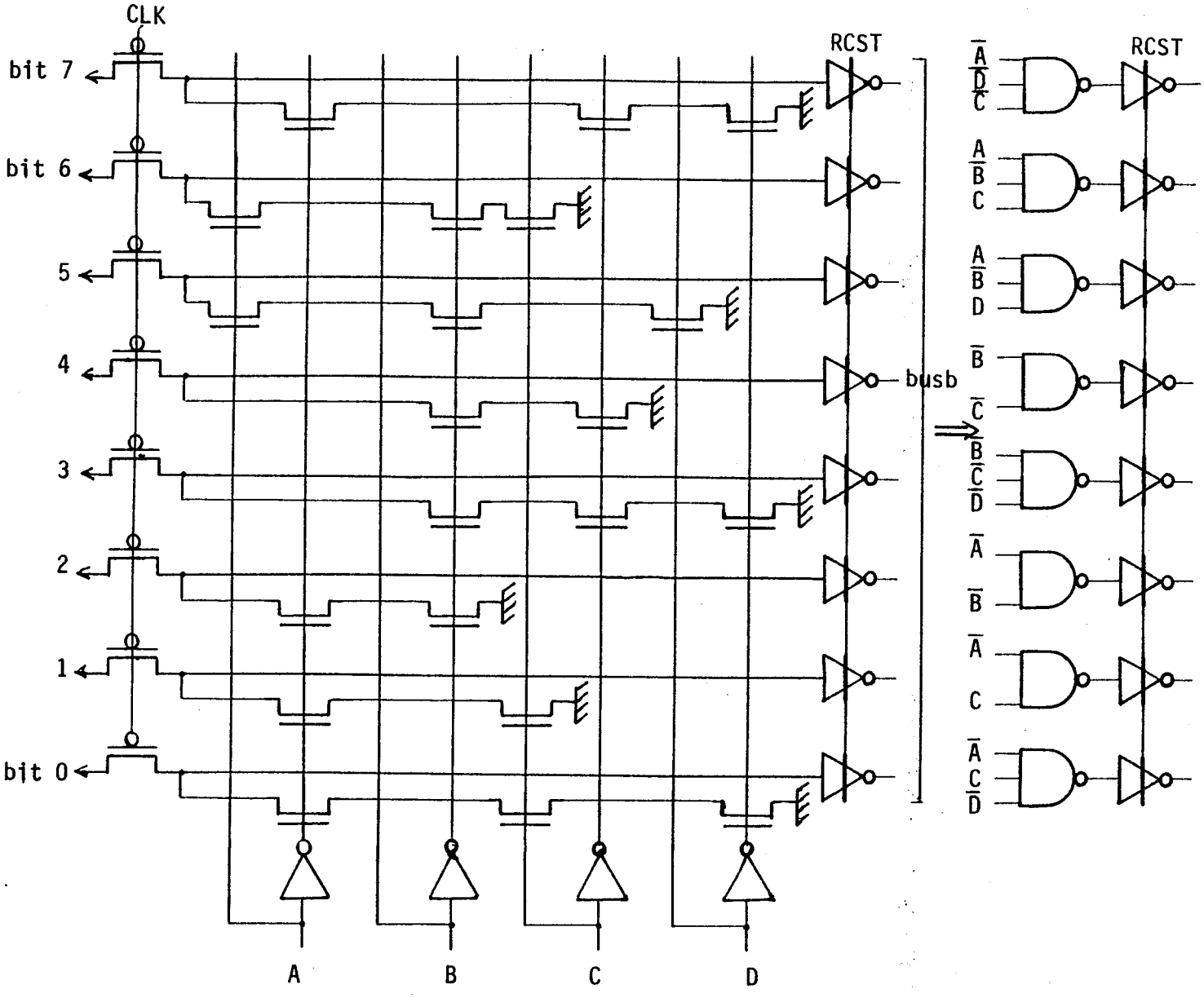


Fig. 3.14- Génération des constantes

3.2- IMPLANTATION EN CMOS

Après avoir établi le schéma logique des différents blocs constituant la partie opérative, il faut définir une stratégie d'implantation de la circuiterie CMOS.

3.2.1- La méthode classique

La méthode classique en NMOS, telle qu'elle a été proposée dans (ref. 16), consiste à dessiner des bandes rectilignes et horizontales en métal au-dessous desquelles est conçue la logique. Ainsi, les bus de données de la partie opérative et les rails d'alimentations seront réalisés en aluminium; tandis que les commandes seront disposées verticalement en polysilicium. Cette dernière couche étant plus résistive que l'aluminium, elle n'est utilisée que pour former un transistor (défini par la superposition du polysilicium sur la diffusion) ou pour établir éventuellement des interconnexions dans le cas où les liaisons en métal seront impossibles à réaliser. Cette topologie semble être une bonne stratégie, mais il reste cependant à définir une hauteur fixe permettant le dessin de toutes les cellules de base. Le choix de ce paramètre est déterminant pour l'optimisation en surface de la partie opérative.
En réalité, pour un système à deux bus complémentés, on n'a besoin que de six lignes d'aluminium par bit (deux bus d'alimentation VDD et VSS et, deux bus de données A et B avec leur complément). Néanmoins, deux lignes de service (notées C et D) ont été ajoutées pour permettre la réalisation des ponts à l'intérieur d'une cellule et la connexion entre les différentes briques. Avec cette technique, il est facile d'utiliser la transparence des blocs en augmentant (si nécessaire) le nombre de lignes de service.
On aura la structure suivante (voir fig. 3.15) qui comporte entre les deux rails d'alimentation VDD, respectivement: deux bus de données A et B, deux lignes de service C et D de part et d'autre de la masse VSS et les compléments des bus de données $\overline{A}$ et $\overline{B}$.

On remarque que la ligne VDD sera partagée entre deux cellules voisines.

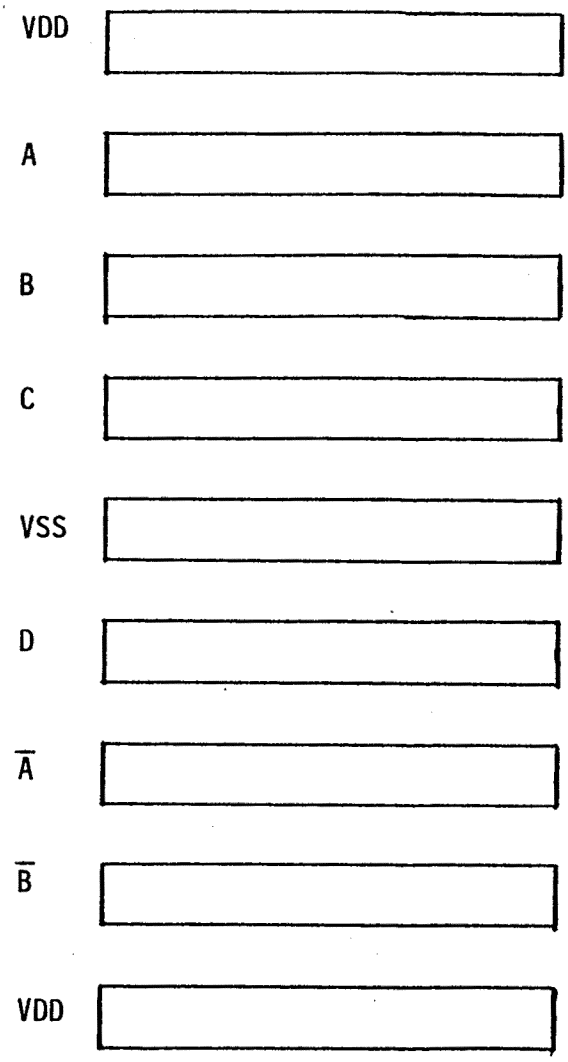


Fig. 3.15- Structure de bandes rectilignes

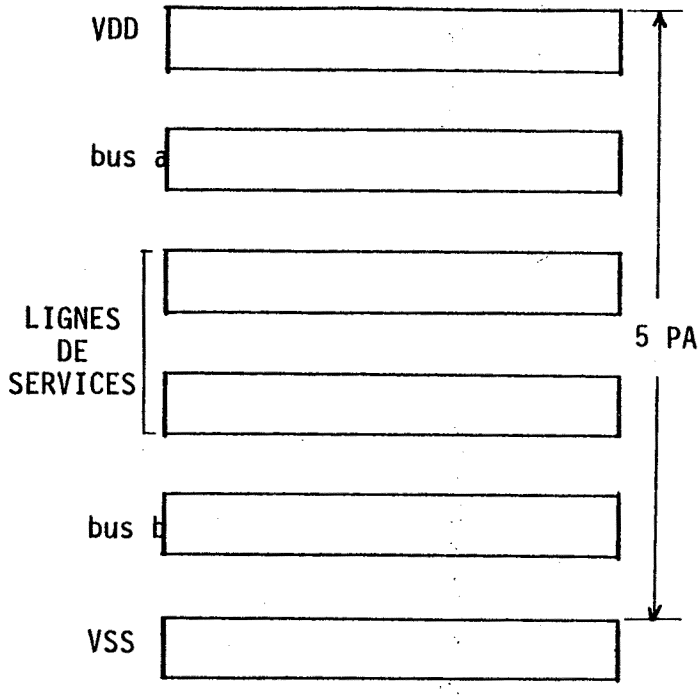


Fig. 3.16- Structure avec deux bus sans complément

On pourra s'inspirer de cette méthode pour l'appliquer à la circuiterie CMOS. Cependant, du fait qu'on travaille avec des bus simples sans complément, on n'aura alors besoin que de la moitié d'une bande. Celle-ci sera définie par les deux rails d'alimentation VDD et VSS à l'intérieur desquels se trouvent deux bus de données a et b et deux lignes de service (voir fig. 3.16). Au total, on aura six bus en métal. La hauteur de la brique sera de cinq pas d'aluminium, puisque les lignes d'alimentation VDD et VSS sont communes aux bits voisins.

Puisque le CMOS possède un caractère symétrique, il sera alors intéressant de représenter les diffusions N+ et P+ horizontalement de part et d'autre des lignes de service, et proche respectivement des lignes VSS et VDD. La topologie consiste donc à avoir une structure de bus horizontaux en aluminium, le polysilicium sera disposé dans le sens vertical tandis que la diffusion lui sera perpendiculaire.

D'autre part, certaines technologies ne permettent pas le contact enterré mais uniquement les contacts simples. Ainsi, un contact diffusion-polysilicium sera réalisé par deux contacts simples diffusion-aluminium et polysilicium-aluminium reliés par un pont métallique.

Dans tout ce qui suivra, il sera représenté:

- l'aluminium en trait mixte (—— - ——)
- le polysilicium en trait interrompu (- - - - -)
- la diffusion en trait continu (————)
- les contacts poly/alu et dif/alu par un point renforcé (.)

Un inverseur est représenté en dessin symbolique par la figure 3.17.

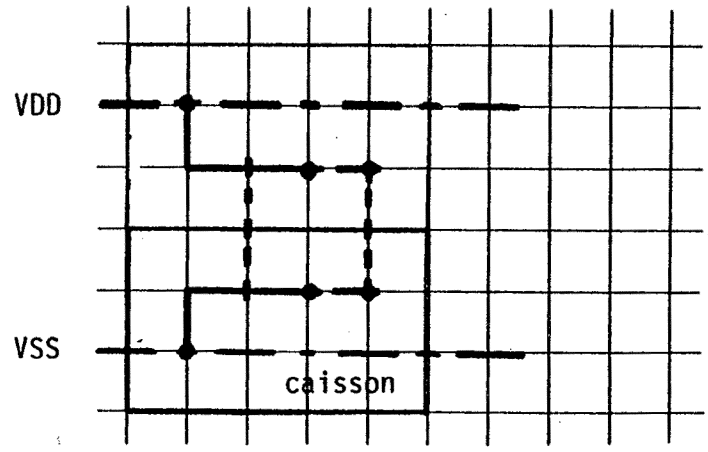


Fig. 3.17- Représentation symbolique d'un inverseur.

On remarque que les ponts métalliques reliant la diffusion au polysilicium prennent de la surface puisqu'on ne peut pas faire passer par dessus des bus en aluminium. Du coup, la structure de

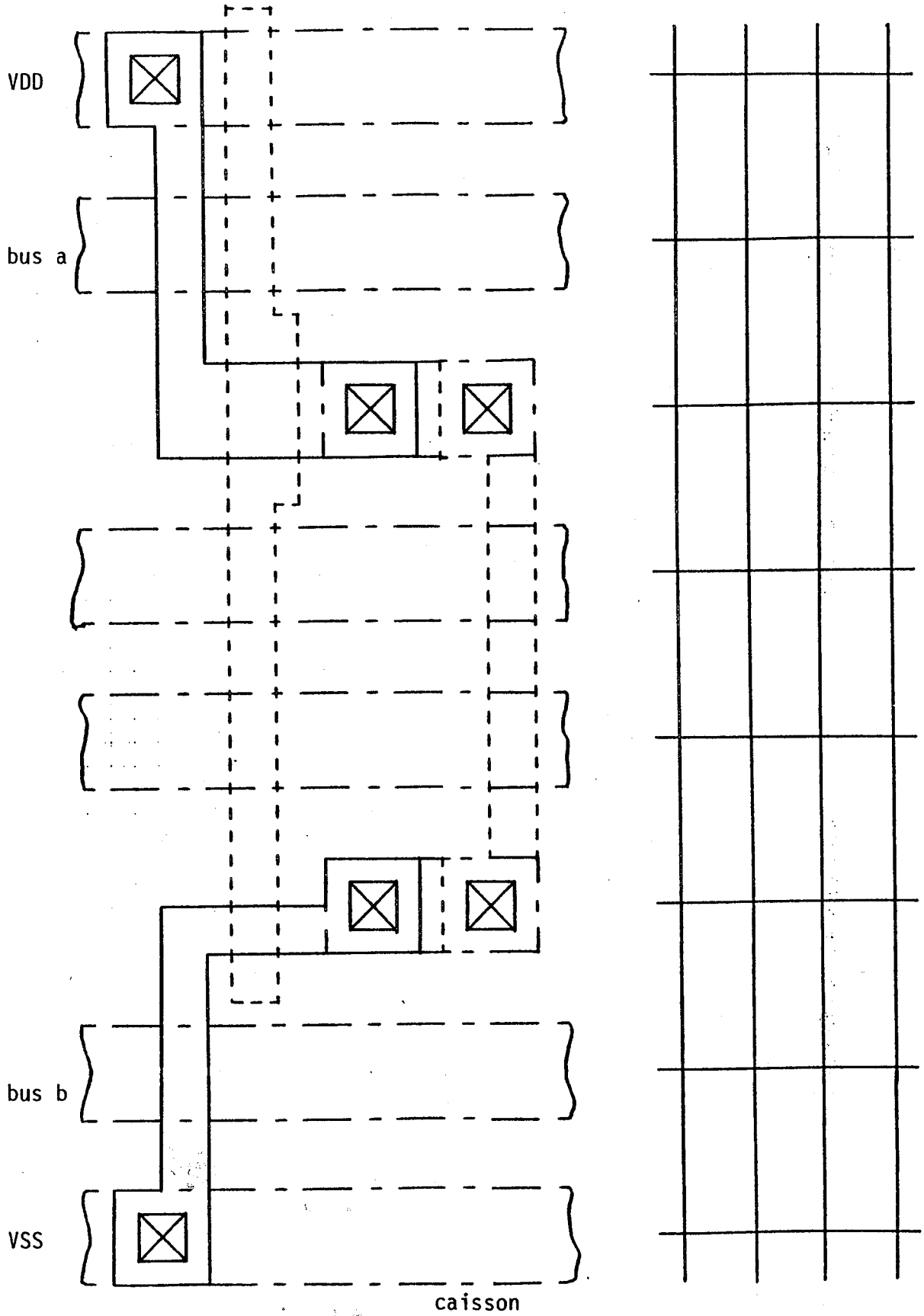
bandes présentée précédemment pour des bus non complémentés (voir fig. 3.16) se trouve élargie de deux pas supplémentaires d'aluminium nécessaires pour la réalisation de ponts et on aura donc une hauteur de 7 pas d'aluminium.

3.2.1.1- Exemple d'implantation d'un inverseur

Un inverseur est dessiné sur bandes avec les dimensions minimales en Lambda (voir fig. 3.18).
De ce dessin, nous vient l'idée de travailler sur une grille formée de colonnes et de lignes avec un pas régulier, comme suggérée dans (ref. 14). Les colonnes de cette matrice sont définies par les contacts et les grilles en polysilicium des transistors, tandis que les lignes le sont par les bus en aluminium et les ponts métalliques.
Dans l'exemple de la figure 3.18, l'inverseur est représenté sur une grille de 4 colonnes et de 8 lignes.

Toutes les cellules de base de la partie opérative auront la même hauteur fixée à 8 lignes. Par contre, la largeur de la grille (ou nombre de colonnes) varie selon le circuit à dessiner.
- les grilles de transistor et les interconnexions en polysilicium se trouvent sur les colonnes.
- les bus en aluminium forment les lignes de la grille.
- les ponts métalliques se trouvent sur les deux lignes situées de part et d'autre des lignes de service.
- les contacts sont placés uniquement aux intersections des lignes et des colonnes.
- les lignes de diffusion se trouvent sur les rangées de ponts métalliques et sous les bus de données. Elles partent aussi dans le sens vertical pour se connecter aux bus d'alimentation.

Avec ces conventions, on peut implanter assez rapidement n'importe quel circuit, sans se soucier des règles de dessin, en travaillant sur cette structure ordonnée. D'autre part, il est facile d'estimer la surface du dessin en fonction du nombre de pas de la grille. Il suffit de définir le pas d'aluminium (PA)



a) Dessin en Lambda

b) Grille correspondante

Fig. 3.18- Implantation d'un inverseur

dans le sens vertical et le pas de polysilicium (PP) dans le sens horizontal et de les évaluer en fonction de la technologie utilisée.

3.2.1.2- Evaluation des pas de la grille

On a deux pas à évaluer, dans le sens vertical (ou pas d'aluminium) et dans le sens horizontal (ou pas de polysilicium).

Par définition, le pas d'une couche conductrice est égal à la somme de sa largeur l et de la distance minimale d séparant deux couches de même niveau.

a) Pas d'aluminium (PA)

D'après les règles de dessin en LAMBDA, on a:

- Distance minimale entre 2 lignes d'aluminium: d(Alu)=3
- Largeur minimale de la couche d'aluminium: l(Alu)=3
- Le contact est de 2 et doit être entouré de 1

Pour loger facilement les contacts sur les bus en aluminium, on prend comme largeur d'aluminium 4 au lieu de 3. D'où l(Alu) = 4 lambda.

PA = d(Alu) + l(Alu) = 3 + 4 = 7

PA = 7 lambda

b) Pas de polysilicium (PP)

Les règles de dessin nous donnent:

- Distance minimale entre 2 lignes de polysilicium: d(Poly)=2
- Largeur minimale de la couche de polysilicium: l(Poly)=2

Puisque le polysilicium est utilisé pour réaliser les grilles de transistor et la longueur du canal P est de 3 lambda, alors on prend comme largeur du polysilicium 3 lambda.

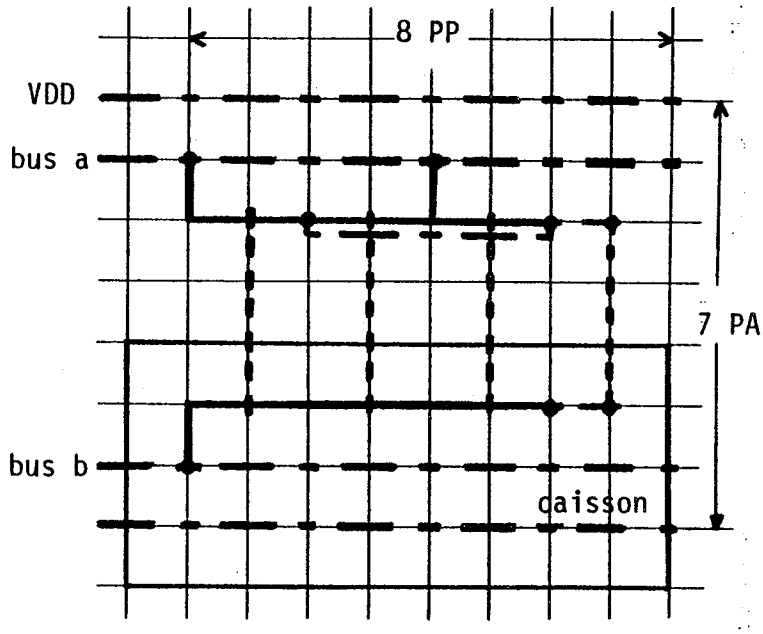
PP = d(Poly) + l(Poly) = 2 + 3 = 5

PP = 5 lambda

3.2.1.3- Exemple d'implantation sur grille

a) Portes simples

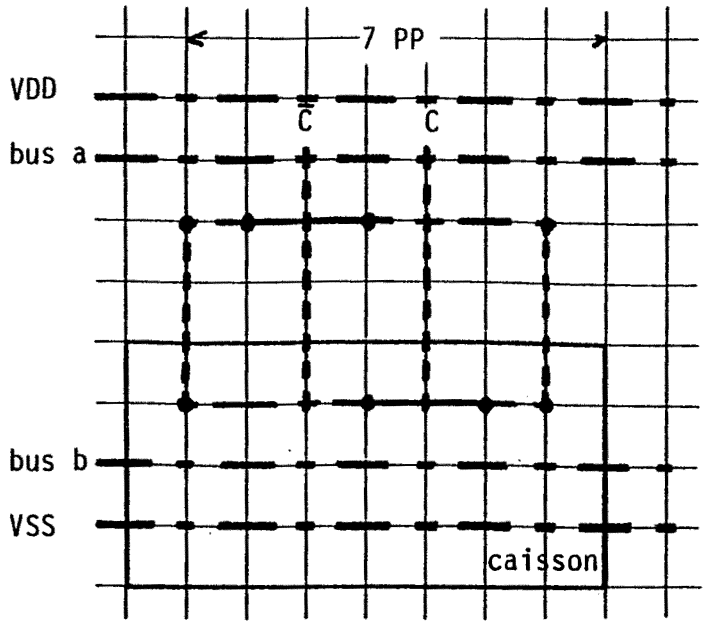
- Porte NAND à trois entrées



Les différentes couches du circuit sont représentées par de simples traits sur cette grille. L'évaluation de la surface de cette porte peut se faire aisément en nombre de pas d'aluminium et de polysilicim.
La hauteur est toujours constante. Elle vaut 7 PA puisque les lignes d'alimentation seront partagées entre les bits voisins.
Dans le sens horizontal, on compte 8 PP en largeur.

- Porte de transmission

Elle est constituée de deux transistors en parallèle, de type N et P, pilotés par deux commandes complémentaires C et $\overline{C}$.



On utilise de grands ponts d'aluminium afin d'y faire passer, à travers, les commandes aux bits voisins. Par conséquent, la longueur du dessin est de 7 PP. Mais si les commandes sont générées localement et parviennent par les lignes de service, alors l'implantation de cette porte de transmission prendra seulement 5 PP dans le sens horizontal.

b) Schéma de l'U.A.L.

L'U.A.L. est implanté en 47 pas de polysilicium (voir fig. 3.19), équivalents à une longueur de 235 lambda. La hauteur étant toujours constante, elle est égale à 49 lambda.
La surface de l'UAL est de: 11.515 lambda-carré.

Avec cette topologie, les circuits sont implantés en longueur et donc on aura des bus très longs. Pour cela, nous nous proposons d'étudier une autre méthode qui utilise toujours la notion de grille mais où les lignes et les colonnes sont interchangées; c'est à dire que la structure précédente subit une rotation de 90°. On aura les lignes en polysilicium et les colonnes en aluminium.

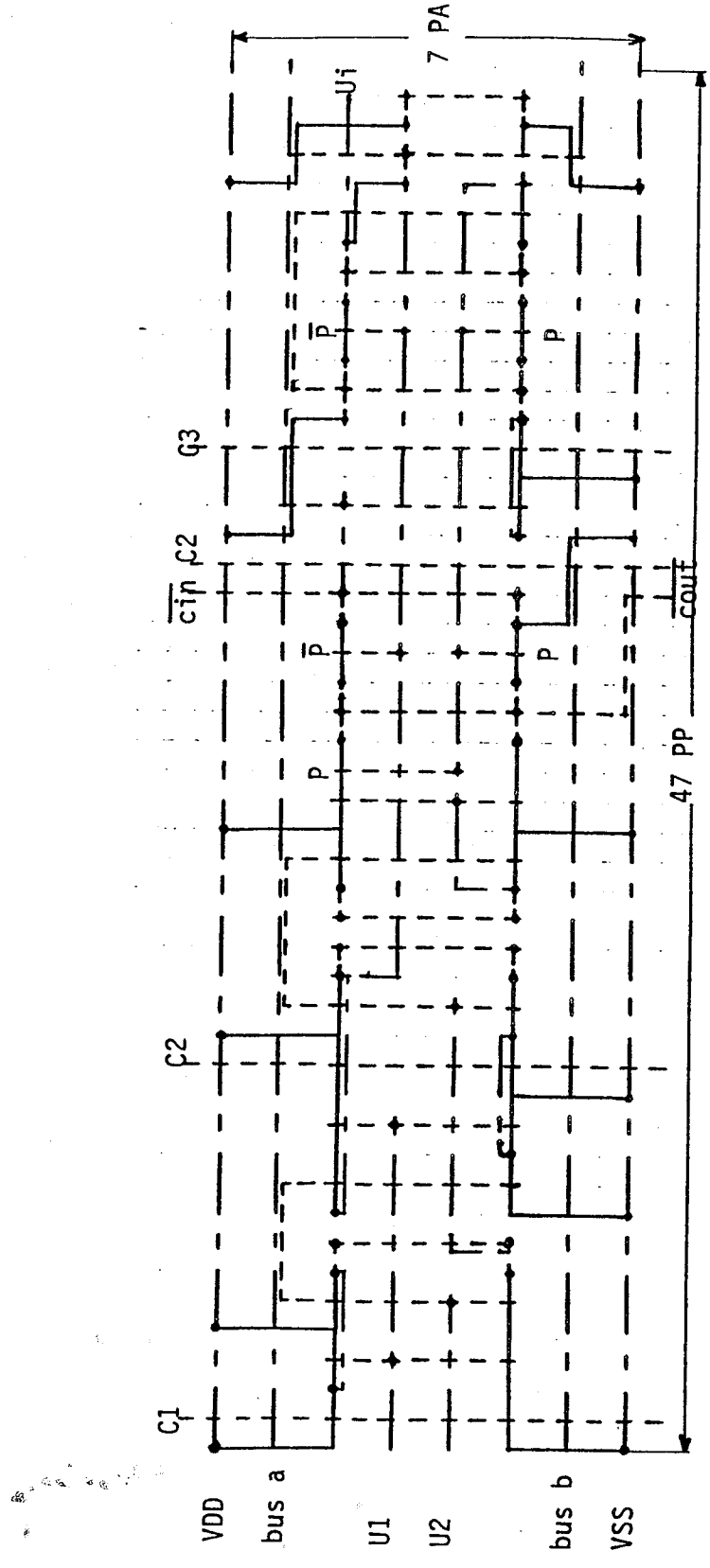


Fig. 3.19- Dessin de l'U.A.L. (Bus en Aluminium)

3.2.2- Autre méthode d'implantation

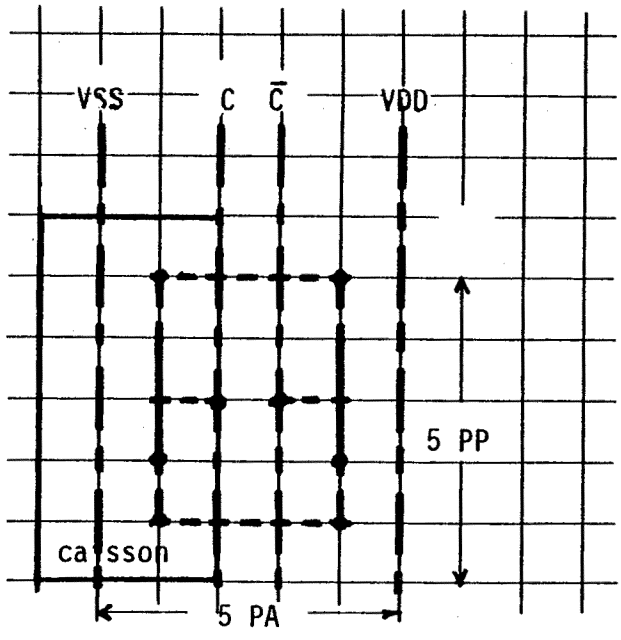
Cette nouvelle stratégie d'implantation consiste à mettre les couches d'aluminium verticalement et celles de polysilicium horizontalement. Dans ce cas là, on aura:

- les bus d'alimentation VDD et VSS en aluminium. Ces deux lignes seront alternées lors de l'implantation d'un circuit.
- les bus de données a et b en polysilicium, ainsi que les interconnexions.
- les commandes en aluminium traversant verticalement les bits.

3.2.2.1- Exemple d'implantation

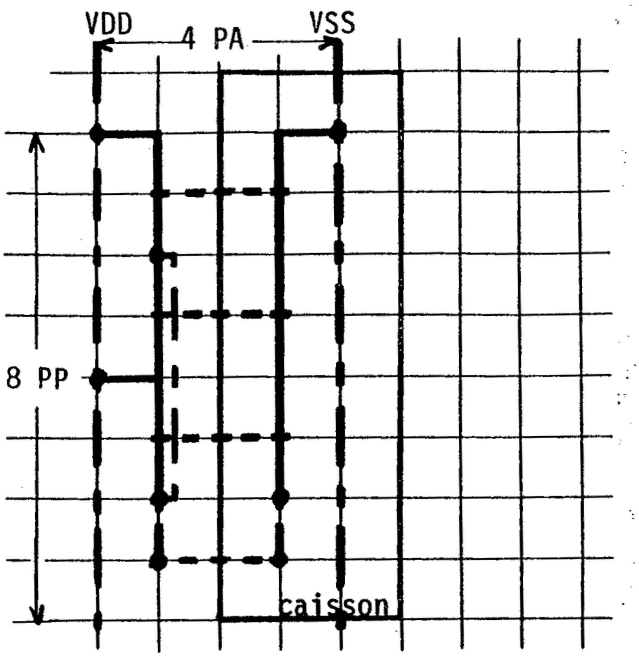
a) Portes simples

- Porte de transmission



La hauteur est évaluée en nombre de pas de polysilicium et vaut 5 PP pour ce dessin. La largeur est de 5 PA.
Les commandes passent verticalement en aluminium, entre les deux rangées de ponts métalliques.

- Porte NAND à trois entrées



La distance entre les lignes d'alimentation VDD et VSS est de 4 PA. C'est la largeur minimale d'une porte élémentaire car il faut laisser un certain espace (équivalent à un pas d'aluminium) entre le caisson des transistors N et la diffusion P+ des transistors P. On peut exploiter ce vide pour faire passer une commande en aluminium, à la limite du caisson, sans modifier la largeur. Mais cette structure se trouve élargie dès que le nombre de commandes augmente.

Quant à la hauteur, elle varie selon le nombre d'entrées d'une porte. Elle vaut 8 PP sur ce dessin.

Remarque : Cette hauteur est de 4 PP pour un inverseur. En général, pour une porte à n entrées, on aura une hauteur de 2(n+1) pas de polysilicium.

b) Schéma de l'U.A.L.

On constate qu'avec cette méthode d'implantation, les deux dimensions (hauteur et largeur du dessin) varient selon le circuit à implanter. Or, nous avons vu que la hauteur est un paramètre important qu'il faut choisir soigneusement pour l'optimisation globale de la surface du dessin.

Pour définir une hauteur convenable, nous avons recherché dans les schémas logiques des différents blocs de la partie opérative, la porte ayant le plus grand nombre d'entrées. Celle-ci se trouve dans l'UAL et possède quatre entrées (voir fig. 3.5). Elle s'implante en 10 pas de polysilicium. Cependant, la hauteur des cellules sera définie à 12 pas de polysilicium, puisqu'on doit ajouter deux pas pour le dessin des bus de données a et b qui sont disposés horizontalement en polysilicium, aux deux extrémités d'une cellule (voir fig. 3.20).

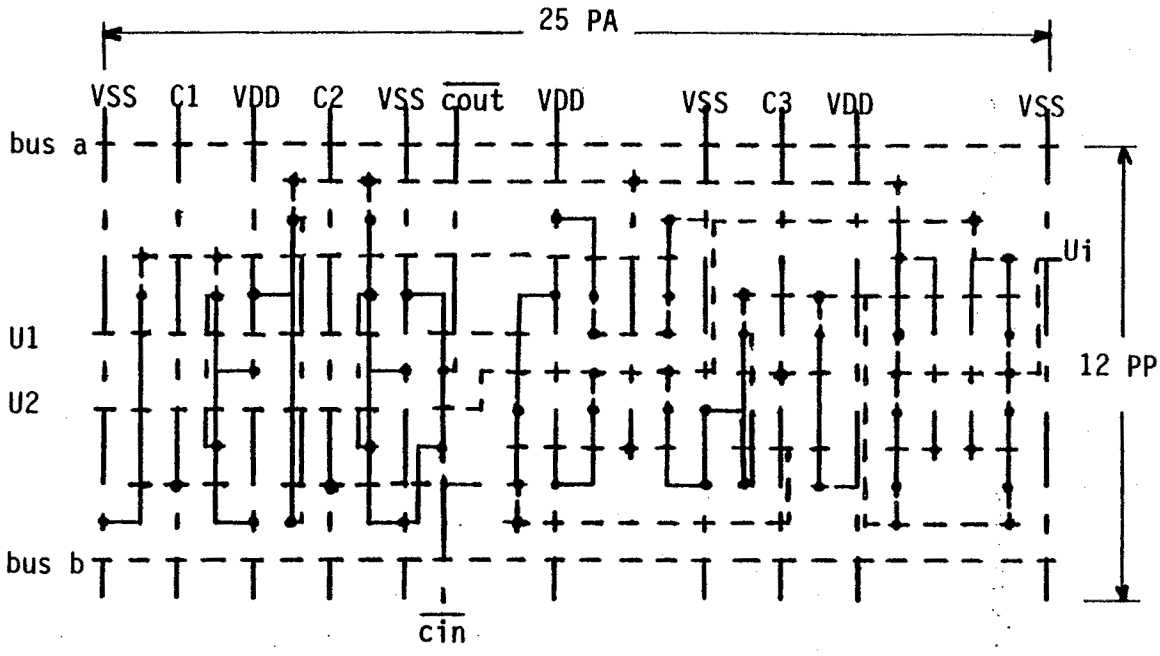


Fig. 3.20- Implantation de l'U.A.L.
(Bus de données en polysilicium)

Ce schéma de l'U.A.L. est dessiné en 25 pas d'aluminium, ce qui donne une longueur de 175 lambda. La hauteur est constante et vaut 12 pas de polysilicium, soit 60 lambda. Sa surface sera donc de 10.500 lambda-carré.

Cette méthode comparée à la précédente s'avère plus efficace puisque la largeur du circuit a été diminuée et la surface optimisée. Elle est aussi intéressante dans le sens qu'elle préfigure une implantation double métal dans laquelle la seconde couche de métal sert à réaliser les bus dans le sens du polysilicium.

Cependant, son seul inconvénient est d'avoir de longues lignes en polysilicium. Ainsi, on se propose d'étudier le temps de propagation à travers les bus de données de la partie opérative que nous allons voir plus loin.

3.2.2.2- Assemblage de la partie opérative

a) Présentation du système LUBRICK

La partie opérative est assemblée, à partir des cellules de base, par programmes écrits en PASCAL qui font appel au programme principal LUBRICK (ref. 2).
Chaque cellule doit avoir un fichier "x" qui décrit ses quatre frontières et éventuellement ses quatre types de connecteurs (dirigés vers l'ouest, l'est, le nord et le sud) avec leurs coordonnées et leurs niveaux de couches.
Le programme LUBRICK se charge d'assembler les figures de base en évitant toute violation des règles de dessin entre les frontières et, de réaliser les connexions entre les différents blocs grâce aux connecteurs déjà déclarés dans leur fichier "x". Il génère à son tour un fichier "x", correspondant à la figure obtenue, qui pourrait être utilisé pour la construction d'une figure plus englobante.

b) Résultats de l'implantation

Les cellules de base sont au nombre de 20 et ont été dessinées sur 12 pas de polysilicium (ou 12 lignes de la grille) définissant leur hauteur. Leur largeur (ou le nombre de colonnes de la grille) dépend fortement du circuit à dessiner.
Les cellules ont d'abord été dessinées en "STICK" et assemblées par programmes à l'aide du système LUBRICK. Les lignes d'alimentation VDD et VSS sont alternées et forment une structure de peigne au niveau de la partie opérative. Les détails techniques concernant son assemblage se trouvent dans le rapport (ref. 11).
La stratégie d'implantation adoptée qui consiste à travailler

sur une structure matricielle de grille, permet d'élargir facilement la largeur W d'un transistor de trois fois sa dimension minimale sans augmenter la surface du circuit.

Ainsi, le passage du dessin en "STICK" de la partie opérative au dessin réel a été fait sans difficultés puisque les programmes d'assemblage pour la version "STICK" ont pu être réadaptés.

Des simulations sur certains points critiques, telle que la propagation de la retenue qui se fait en 5 ns à travers une U.A.L. de 8 bits, ont été effectuées; puis les transistors des cellules de base ont été redessinés avec leurs propres dimensions.

Cette étape du travail a été réalisée par C.MATHERON et P.OBJOIS. Le circuit a été fabriqué dans le cadre CMP CMOS 84. Des plots ont été rajoutés pour permettre le test sous pointes du circuit.

On dénombre un total de 1888 transistors pour la partie opérative. Sa surface est de 1,74 mm2 (2520um x 690um) pour une technologie de 2,5um. Nous obtenons ainsi une densité de 1085 T/mm2 pour la partie opérative. Cela constitue un bon résultat. Toutefois, il reste à évaluer le temps de propagation du bus en polysilicium qui pourrait être un handicap pour cette configuration.

3.2.2.3- Calcul du temps de propagation dans le bus en polysilicium

D'après l'architecture de la partie opérative (voir fig. 2.4), neuf registres sont écrits par l'intermédiaire du bus b alors que seulement cinq registres par le bus a. Ainsi, le temps de propagation à travers le bus b sera plus important. Celui-ci est connecté à neuf inverseurs (entrée des registres) et est attaqué par un amplificateur trois états. On peut le représenter électriquement par la figure 3.21.

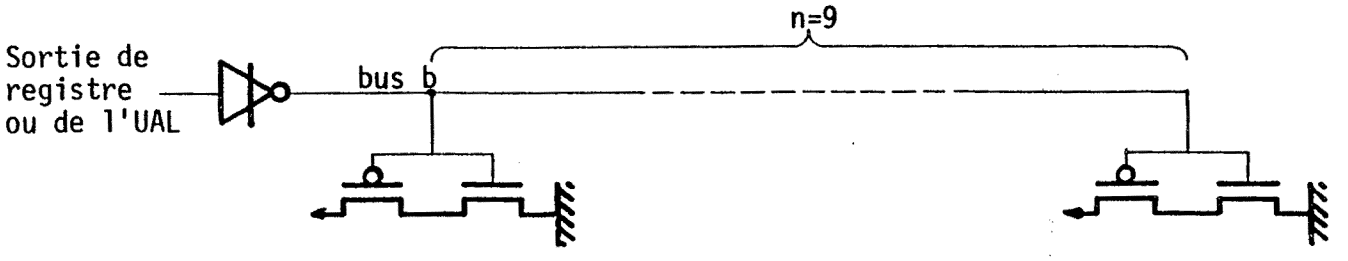


Fig. 3.21- Schéma électrique du bus b

La porte 3 états qui alimente les inverseurs des registres de la partie opérative, est remplacée par une résistance équivalente RO.
Le bus b est considéré comme ayant à la fois une résistance R et une capacité C.
L'inverseur, à l'entrée du registre, présentera une capacité CO. Sa résistance est confondue avec celle du bus.

On suppose que le bus est une ligne à résistances et capacités réparties et que les inverseurs des registres lui sont connectés à égale distance. De cette hypothèse, le bus peut être schématisé de la manière suivante:

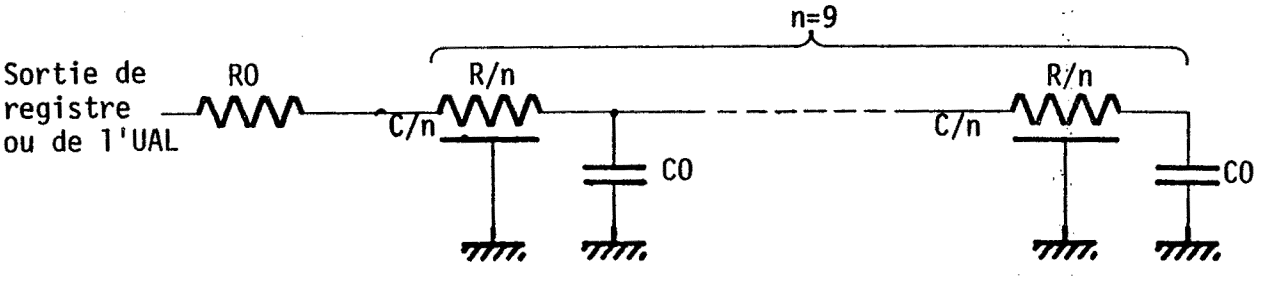


Fig. 3.22- Schéma équivalent du bus b

Dans cette figure :
RO désigne la résistance équivalente à l'amplificateur 3 états CMOS,
R et C étant respectivement la résistance et la capacité du bus,
CO désigne la capacité d'un inverseur et n étant le nombre des inverseurs.

Le temps de propagation est donné par la formule déterminée dans (ref. 7):

$$t = \sum_{i=1}^{n} \sum_{j=1}^{i} 2\ Rj\ Ci$$

où n est le nombre de cellules RC (n=9),

Ci désigne la capacité de la cellule i,

$\sum_{j=1}^{i} Rj$ représente la somme de toutes les résistances du début de la ligne jusqu'à la cellule i.

On a: $\sum_{j=1}^{i} Rj = RO + i\ R/n$

D'autre part, $Ci = CO + C/n$

En remplaçant ces valeurs dans l'équation du temps de propagation, on trouve:

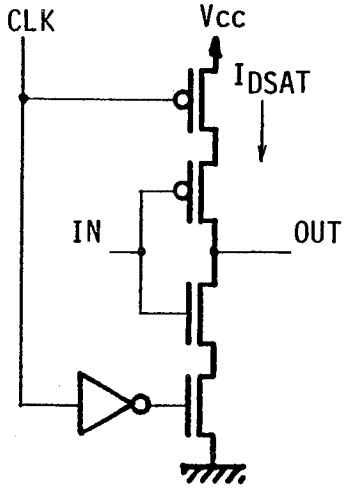
$$t = 2 \sum_{i=1}^{n} Ci \sum_{j=1}^{i} Rj = 2 \sum_{i=1}^{n} (CO + C/n)\ (RO + i\ R/n)$$

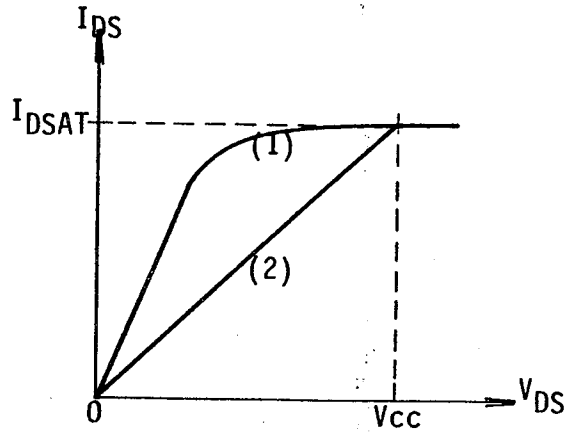
$$t = 2\ (CO + C/n) \sum_{i=1}^{n} (RO + i\ R/n)$$

$$= 2\ (CO + C/n)\ (nRO + n(n+1)\ R/2n)$$

Finalement, $t = 2\ RO\ (C + n\ CO) + R\ (\ (n+1)\ C/n + (n+1)\ CO\ )$ (1)

- Evaluation de RO :





(1) caractéristique du transistor MOS
(2) approximation du transistor à une résistance RO.

Lorsque la commande CLK est activée, on a selon l'entrée IN soit la conduction des deux transistors du type P, soit celle du type N. Pour être dans des conditions défavorables, on considère que ce sont les transistors du type P qui conduisent.
D'autre part, en approximant la caractéristique (1) du transistor MOS à la droite (2) d'une résistance RO (voir figure ci-dessus), on se place dans un cas critique. En effet, le courant considéré est plus faible que la normale et donc, la résistance équivalente évaluée RO sera plus importante.

$$RO = VCC / IDSAT \qquad (2)$$

où VCC est la tension d'alimentation et IDSAT le courant de saturation dont son expression est donnée par la formule suivante:

$$IDSAT = K' . (W/L) . (VGS - VT)^2$$

où K' = u Cox/2 (transconductance),
W/L est le rapport des géométries du transistor équivalent aux deux transistors P en série (Wp/2Lp),
VGS est la tension de grille par rapport à la source et,

VT est la tension de seuil.

En considérant la conduction des transistors de type P, on a:

$$IDSAT = K'p \cdot (Wp/2Lp) \cdot (VGS - VT)^2$$

D'où en remplaçant dans l'équation (2), on trouve:

$$RO = VCC / (K'p \cdot (Wp/2Lp) \cdot (VGS - VT)^2)$$

On prend les dimensions minimales d'un transistor P pour se placer dans le cas défavorable. Avec Wp = Lp = 3 lambda, on trouve:

$$RO = 67,8\ k\Omega$$

- Calcul de CO :

La capacité CO d'un inverseur est la somme des capacités de grille des transistors N et P.

$$CO = Cox\ (Wn.Ln + Wp.Lp)$$

où Cox désigne la capacité de grille par unité de surface,
Wn.Ln exprime la surface de la grille du transistor N et,
Wp.Lp celle du transistor P.

Avec Wn = Wp = 3 lambda,
Ln = 2 lambda et Lp = 3 lambda,
on trouve:

$$CO = 0,015\ pF$$

- Calcul de R et C du bus :

Le bus est en polysilicium; ses deux paramètres R et C seront donc plus importants que dans le cas du bus en aluminium.

* Calcul de R :

La partie opérative a été dessinée sur une largeur de 274 pas d'aluminium. La longueur L du bus de données sera donc de:

$$L = 274 \times 7 = 1918 \text{ lambda}$$

Les bus ont une largeur l = 3 lambda et donc le nombre de carrés sera de:

$$N = L / l = 1918 / 3 = 639$$

La résistance carré du polysilicium est: r = 17 Ω/□ (valeur typique)

D'où: $R = 17 \times 639 = 10,8\ k\Omega$

* Calcul de C :

Surface du bus: $S = L \times l = 1918 \times 3 = 5754$ lambda-carré

On trouve pour la capacité C du bus en polysilicium:

$$C = 0,43\ pF$$

- calcul du temps de propagation t :

Il suffit de remplacer dans la formule (1), les paramètres par leur valeur calculée précédemment:

Application numérique:

R0 = 67,8 kΩ

R. = 10,8 kΩ

C0 = 0,015 pF

C = 0,43 pF

n = 9

on trouve: $t = 83$ ns

- Remarque :

Il est à noter que le temps de propagation calculé reste inférieur à la durée d'une phase (200 ns) de l'horloge bien qu'on ait pris des hypothèses défavorables telles que le courant de charge plus faible (lors de l'évaluation de R0) et les dimensions minimales des transistors. De ce fait, il représente le temps maximum de propagation.

La formule (1) peut s'écrire sous la forme:

$$t = t1 + t2$$

où $t1 = 2\ R0\ (C + nC0)$

$t2 = R\ (\ C\ (n+1)/n + (n+1)\ C0\ ) = (\ (n+1)/n\ )\ R\ (C + n\ C0)$

On constate que les termes t1 et t2 sont identiques à un facteur près. Il suffit de faire le rapport des résistances (2 R0) et ((n+1)/n) R pour les comparer.

$$t1 / t2 = 2\ R0 / ((n+1)/n)\ R = 11,3$$

t1 est onze fois plus important que t2. Donc, il faut agir surtout sur le premier terme afin de diminuer le temps de propagation t. Pour cela, on doit augmenter le rapport des géométries des transistors de la porte 3 états d'attaque du bus pour que la résistance R0 diminue.

Par exemple, si on double la largeur W des transistors, la résistance R0 sera divisée par 2 et le temps t se trouvera presque diminué de moitié.

Application numérique :

Wp = 6 lambda, $R0 = 67,8/2 = 33,9\ k\Omega$

$t1 = 67,8\ (0,43 + 0,135) = 38,307$ ns

$t2 = 10,8\ (0,473 + 0,165) = 6,78$ ns

D'où: $t = 38,307 + 6,78 = 45,087$ ns

$$t = 45 \text{ ns}$$

- Cas du bus en aluminium :

Dans le cas d'un bus en aluminium, t2 est négligeable devant t1 et on aura alors:

$$t = 2RO\ (C + n\ CO)$$

D'autre part, on suppose que les deux méthodes d'implantation, l'une utilisant les bus en polysilicium et l'autre les bus en aluminium, donnent la même longueur pour la partie opérative. Ainsi, en prenant comme largeur du bus 4 lambda, sa surface sera de 7672 lambda-carré et sa capacité C de 0,3 pF.

Pour RO = 33,9 kΩ, on a: $t = 67,8\ (0,3 + 0,135) = 29,491$ ns

$$t = 29,5 \text{ ns}$$

- CHAPITRE 4 : SCHEMAS DES CIRCUITS AUXILIAIRES ET CONCEPTION DE LA PARTIE CONTROLE -

4.1- SCHEMAS DES CIRCUITS AUXILIAIRES

4.1.1- Les entrées-sorties

4.1.1.1- Ports d'entrées-sorties

a) Ports 1 et 2

b) Bus de données

4.1.2- Circuit d'interruptions

4.1.3- Compteur d'instants et d'évènements externes

4.2- CONCEPTION DE LA PARTIE CONTROLE

4.2.1- Notions sur MACSIM

4.2.2- Architecture de la partie contrôle

4.2.3- Réalisation du générateur de temps

4.2.4- Evaluation de la partie contrôle

4.2.4.1- Evaluation d'un PLA

4.2.4.2- Application aux PLA de la partie contrôle

a) PLA de propriétés

b) PLA de commandes

c) Petits PLA

4.2.4.3- Surface de la partie contrôle

4.3- RESULTATS FINAUX

CHAPITRE 4
SCHEMAS DES CIRCUITS AUXILIAIRES ET CONCEPTION DE LA PARTIE CONTROLE

4.1- SCHEMAS DES CIRCUITS AUXILIAIRES

4.1.1- Les entrées-sorties

Le micro-ordinateur 8048 possède trois ports d'entrées-sorties (Port1, Port2 et BUS) et trois lignes de test (T0, T1 et $\overline{INT}$) en entrées.

4.1.1.1- Ports d'entrées-sorties

a) Ports 1 et 2

Ces deux ports ont le même schéma électrique.

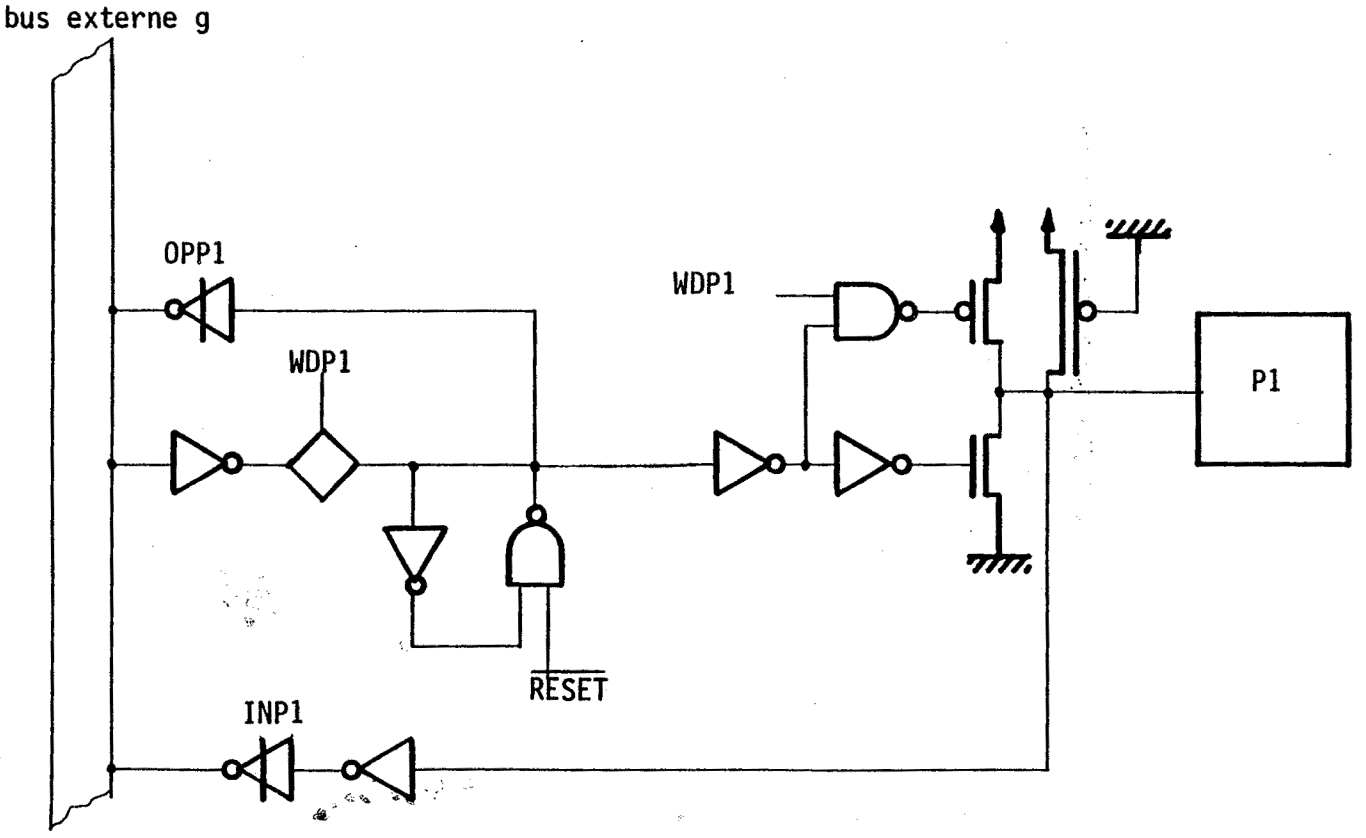


Fig. 4.1- Circuit du port 1

Cette structure permet d'avoir à la fois l'entrée et la sortie sur la même broche.
L'écriture dans les ports 1 et 2 s'effectue grâce à la commande WDP1 (WDP2 pour le port2). Leurs sorties demeurent inchangées jusqu'à leurs réécritures par l'instruction OUTL Pp,A. La commande OPP1 (OPP2 pour le port2) est activée pour effectuer les opérations logiques ("OU", "ET") des ports 1 et 2 à travers l'UAL de la partie opérative, pendant l'exécution des instructions ORL Pp,A et ANL Pp,A.
D'autre part, pour utiliser une broche des ports 1 et 2 comme entrée, il faut d'abord la programmer à l'état "1" par l'intermédiaire de l'instruction OUTL Pp,A. De cette manière, l'utilisateur peut définir certains bits du port en entrées et d'autres en sorties. Toutefois, l'entrée n'est pas mémorisée et sera donc validée par la commande INP1 (INP2 pour le port2) lors de l'exécution de l'instruction IN A,Pp.

Remarques

1- Pendant le "reset", toutes les broches des ports 1 et 2 sont initialisées à l'état logique "1". Ceci est réalisé grâce à la commande $\overline{\text{RESET}}$ et au transistor P dont la grille est reliée à la masse. Ce transistor est long pour éviter un fort débit de courant.

2- L'inverseur de sortie (ou "buffer") est piloté par la commande WDP1 (WDP2 pour le port2) afin d'empêcher une conduction permanente des transistors P en parallèle.

3- Les broches de poids faible (P20 à P23) du port2 peuvent être utilisées comme moyen de communication avec le microproceseur 8243 pour étendre le nombre des entrées-sorties (voir 1.2.3). Cependant, l'information contenue auparavant dans cette partie du port2 est perdue lors de l'exécution des instructions MOVD A,Pi, MOVD Pi,A, ANLD Pi,A et ORLD Pi,A.

Cependant, les quatre broches de poids faible du port2, outre leur utilisation avec le microprocesseur 8243 pour l'extension du

système d'entrées-sorties, servent à faire sortir les quatre bits de poids fort d'adresse pour la mémoire ROM externe. Puisque l'information contenue dans le port2 doit être restituée après l'acquisition de l'instruction (ou la donnée) de l'extérieur, alors il faut prévoir quatre autres bascules pour recevoir l'adresse de la mémoire externe. Leur sortie vers les broches P20 à P23 sera multiplexée avec celle des bascules contenant la donnée du port2 (4 bits de poids faible) et pilotée par la commande SAD2.

Le schéma électrique du port2 sera donc légèrement modifié dans sa partie de poids faible.

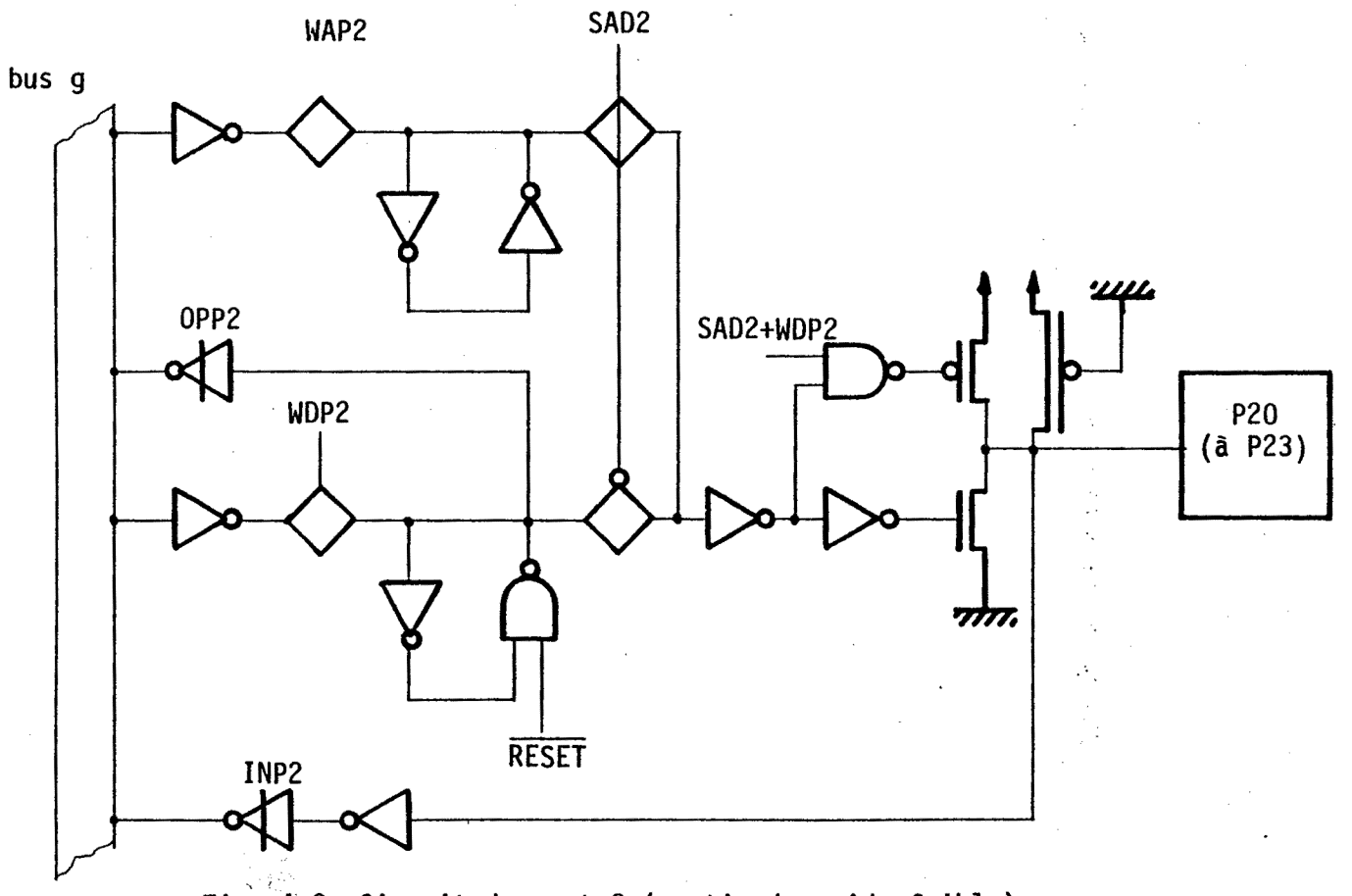


Fig. 4.2- Circuit du port 2 (partie de poids faible)

b) Bus de données

Le bus de données est aussi un port bidirectionnel de 8 bits. Les sorties sont mémorisées, tandis que les entrées ne le sont

pas. Contrairement aux ports 1 et 2, les entrées et sorties ne peuvent pas être programmées sur ce port BUS. Il travaille en entrée ou en sortie selon les signaux de contrôle $\overline{RD}$ ou $\overline{WR}$ générés respectivement pendant les instructions INS ou OUTL.

Ce bus est aussi utilisé pour sortir l'adresse (8 bits) pour la mémoire ROM ou RAM externe. Cependant, l'instruction OUTL ne doit jamais être utilisée dans un système employant une mémoire morte externe puisqu'elle modifie le contenu du registre de sortie BUS.

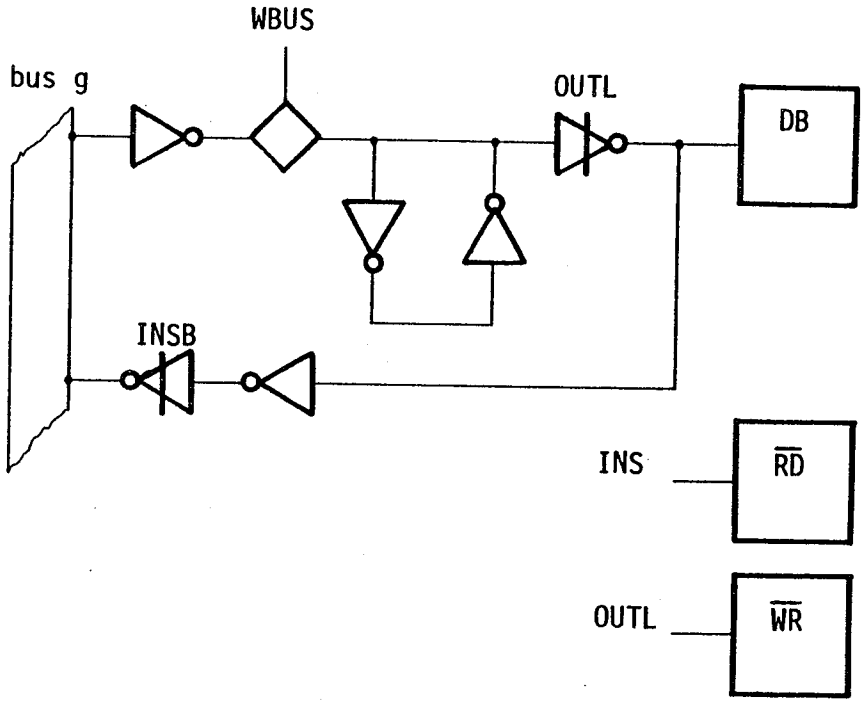


Fig. 4.3- Circuit du BUS de données

4.1.2- Circuit d'interruptions

On a deux types d'interruption:

- interruption externe dont la demande se fait par l'intermédiaire de la broche $\overline{INT}$ qui est échantillonnée au dernier cycle de l'instruction en cours durant le signal ALE.

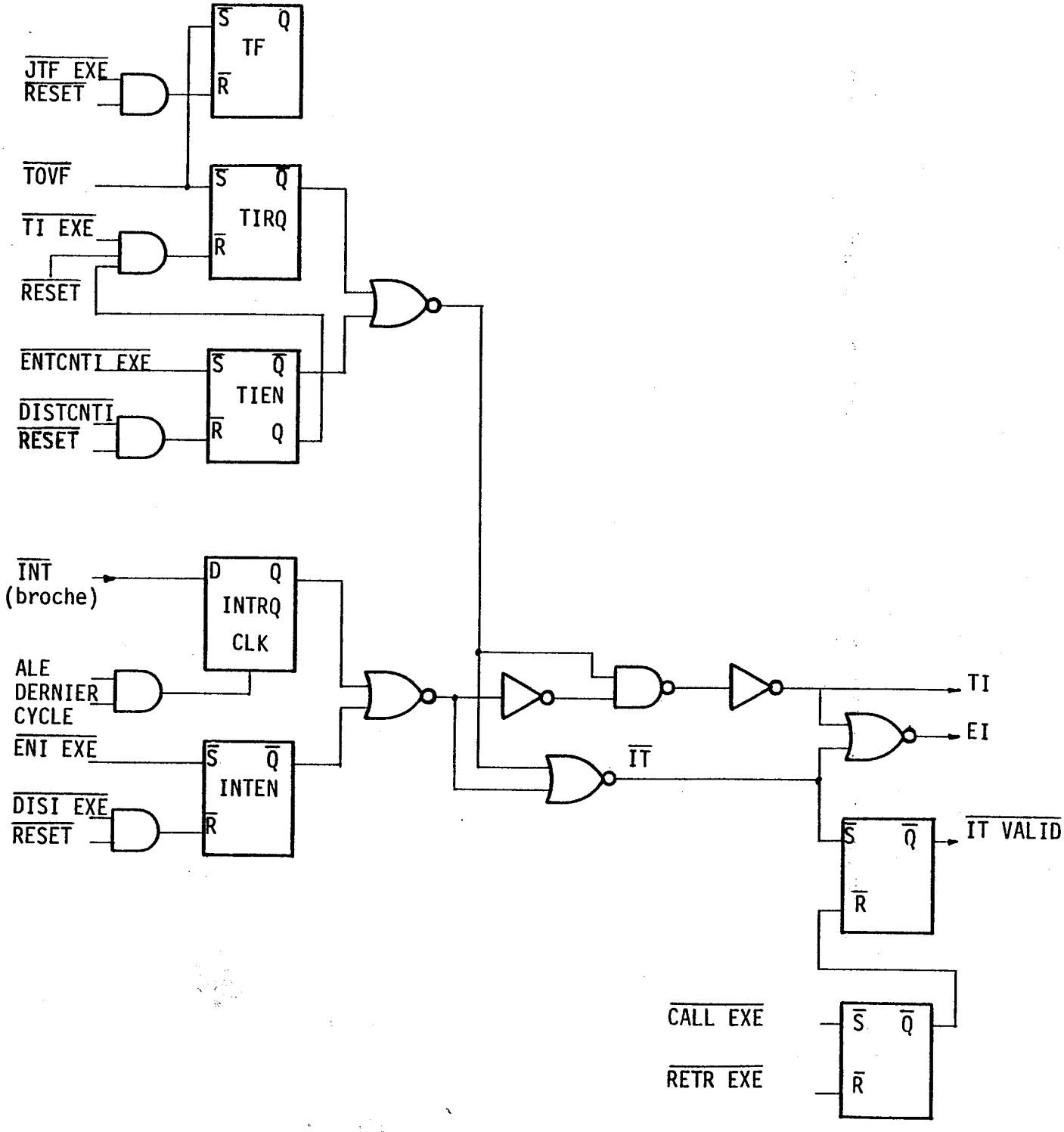


Fig. 4.4- Circuit d'interruptions

- interruption due au compteur d'instants et d'évènements externes dont la demande est sollicitée par un débordement du compteur T lors de son incrémentation.

Ces deux demandes d'interruption ne seront prises en compte qu'après exécution des instructions EN I et EN TCNTI.
Dans le cas où les deux interruptions sont détectées au même moment, alors celle qui vient de l'extérieur est prioritaire. L'autre interruption ne sera exécutée qu'après le retour au programme principal par l'intermédiaire de l'instruction RETR.

- Description du circuit d'interruption.

Le circuit est formé essentiellement de bascules.
- Une bascule D, notée INT RQ, permet la mémorisation de l'état de la broche $\overline{INT}$ échantillonné par le signal ALE, au dernier cycle de l'instruction.
- Une bascule RS, notée INT EN, est mise à "1" lors de l'exécution de l'instruction EN I et remise à "0" pendant l'instruction DIS I ou l'initialisation "RESET".

Il est évident que l'interruption externe n'aura lieu que si ces deux bascules sont simultanément à "1".

Deux autres bascules RS sont nécessaires pour l'interruption due au compteur.
- Une bascule, notée TI EN, est mise à "1" pendant l'exécution de l'instruction EN TCNTI et remise à "0" par l'instruction DISTCNT I ou lors de l'initialisation "RESET".
- Une bascule, notée TI RQ (demande d'interruption du compteur T), est mise à "1" s'il en résulte un débordement lors de l'incrémentation du compteur et remise à "0" après l'exécution de l'instruction DISTCNT I ou de la routine d'interruption (due au compteur) ou enfin pendant l'initialisation "RESET".
Cette bascule est implantée dans la partie opérative ainsi que l'indicateur TF (voir 2.2.2.2).

D'autre part, l'interruption due au compteur T n'est validée que si ces deux dernières bascules sont à "1" simultanément.

A la sortie des bascules, on a le circuit de détection des interruptions qui génère le signal $\overline{IT}$.

IT = 1 interruption détectée.

IT = 0 pas d'interruption.

Le circuit de priorité des interruptions, formé de portes logiques, délivre deux commandes TI et EI. Celles-ci sont exclusives et ne peuvent donc être simultanément à "1". En effet, s'il y a une demande d'interruption externe, alors le signal TI (validation de l'interruption due au compteur T) serait nul.

Par exemple:

- si TI = 1, alors EI = 0 On aura une interruption générée par le compteur T.
- si EI = 1, alors TI = 0 L'exécution du sous-programme d'interruption externe aura lieu.

Finalement, il nous faut une bascule RS pour mémoriser toute demande d'interruption. Elle délivre la commande $\overline{\text{IT VALID}}$ qui permet de signaler qu'on est en attente d'une interruption. Celle-ci sera exécutée à la fin de l'instruction en cours.

- si $\overline{\text{IT VALID}}$ = 0, alors le registre d'instruction RI est forcé, pendant l'état S1, à la valeur héxadécimale 14 qui correspond au code opération de l'instruction CALL. Durant l'exécution de celle-ci, le contenu du compteur de programme (PC) courant est sauvegardé dans la pile, ainsi que celui de la partie de poids fort du registre PSW. La nouvelle valeur du PC sera soit 03H ou 07H selon que le signal EI ou TI soit égal à "1", respectivement.
- si $\overline{\text{IT VALID}}$ = 1, on aura alors le déroulement normal du programme.

Cependant, une autre bascule RS est nécessaire pour inhiber le signal $\overline{\text{IT VALID}}$. En effet, ce signal nous sert uniquement à

forcer le code opération à la valeur 14, afin de faire exécuter l'instruction CALL et de rentrer dans un sous-programme d'interruption. Ce signal doit être remis à zéro pendant l'instruction CALL. La sortie du sous-programme est faite à l'aide de l'instruction RETR pendant laquelle le signal $\overline{IT}$ de détection d'interruption est validé.

4.1.3- Compteur d'instants et d'évènements externes

Le compteur T peut travailler en générateur d'instants précis ou en compteur d'évènements externes selon l'instruction exécutée STRT T ou STRT CNT respectivement. Il est arrêté par l'instruction STOP TCNT ou lors de l'initialisation "RESET".

Le générateur d'instants est incrémenté chaque 32 cycles machine; tandis que l'incrémentation du compteur d'évènements s'effectue, à une cadence d'une fois tous les trois cycles au maximum, à chaque transition du niveau haut au niveau bas sur la broche d'entrée T1.

Le débordement du compteur T, résultant de son incrémentation, est mémorisé dans deux bascules TF et TI RQ (voir fig. 4.4).

Ce compteur T est implanté dans la partie opérative puisqu'il est constitué de 8 bits. L'opération sur T se fait systématiquement à l'état S4 à travers l'U.A.L.

(S4,T1) U1 <--- a:= T; cin:= (0 ou 1);

(S4,T2) T <--- b:= U1+U2

On remarque que la retenue entrante cin de l'UAL peut prendre deux valeurs 0 ou 1. Son calcul est effectué par le circuit suivant:

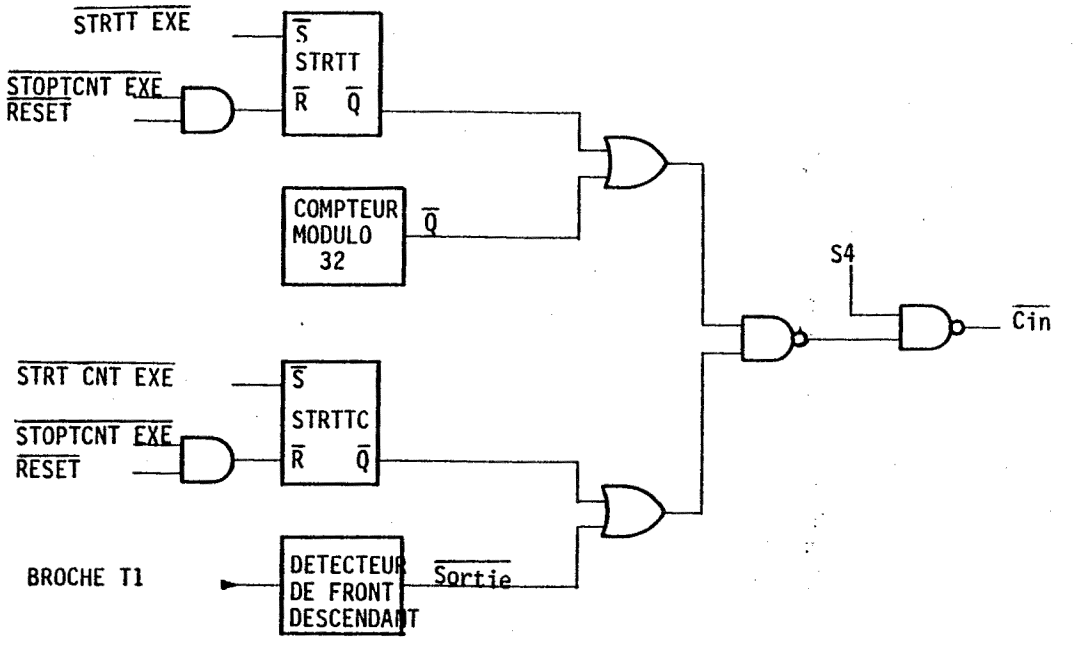


Fig. 4.5- Schéma de la génération de la retenue Cin pour le compteur T

Le circuit est constitué de deux bascules RS, d'un compteur modulo 32 et d'un détecteur de front descendant.

- Les bascules, notées STRT T et STRT TC , sont mises respectivement à "1" lors de l'exécution des instructions STRT T et STRT CNT.

Ces deux bascules sont remises à "0" par l'instruction STOP TCNT et, de ce fait, la retenue cin générée par le circuit sera nulle.

- Le compteur modulo 32 compte le nombre de cycles machine. Il permet, tous les 32 cycles, l'incrémentation du compteur T lorsque celui-ci travaille en mode compteur d'instants.

Il est remis à zéro pendant l'instruction STRT T.

- Le détecteur de front descendant permet l'incrémentation du compteur T, en mode compteur d'évènements externes, à chaque transition du niveau "1" au niveau "0" sur la broche T1. Celle-ci doit rester au niveau haut pendant au moins 500 ns.

Ce détecteur est réalisé à l'aide d'une porte "NOR" devant laquelle sont placées des bascules maître-esclave pour la synchronisation du signal à l'entrée de la broche T1 (voir fig. 4.6).

La sortie S est mémorisée dans une bascule RS qui est mise à "1" lors de la détection d'un front descendant. Elle est remise à zéro après l'incrémentation du compteur T.

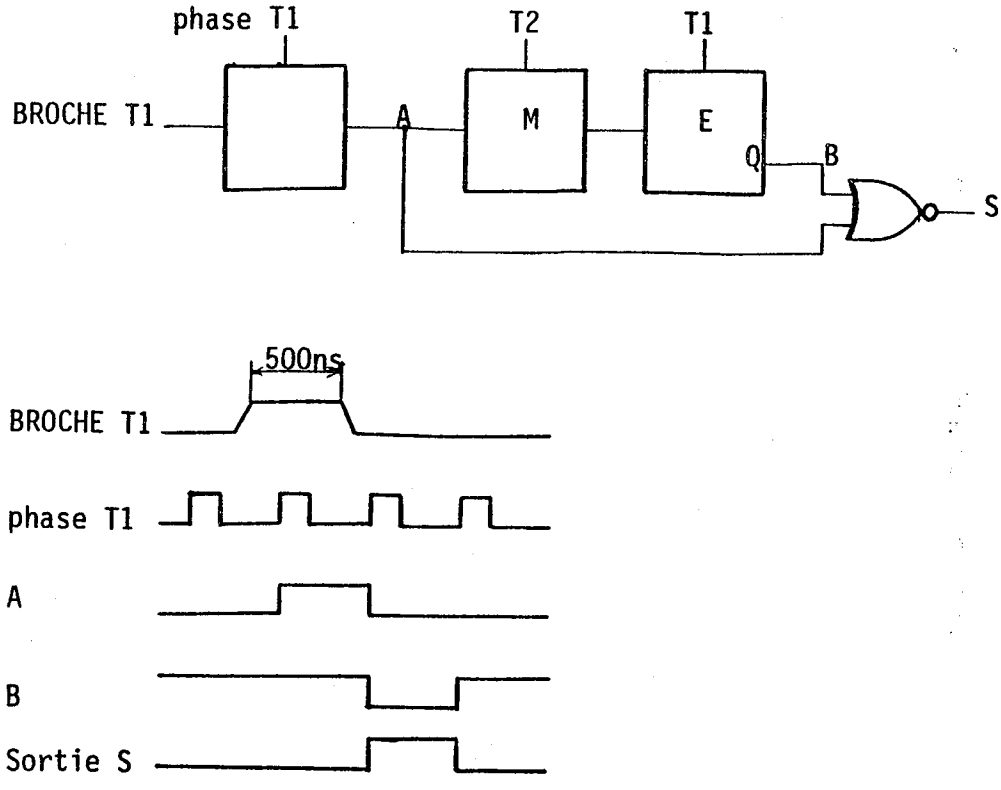


Fig. 4.6- Détection du front descendant

4.2- CONCEPTION DE LA PARTIE CONTROLE

L'algorithme d'interprétation des instructions qui a servi pour l'établissement de la partie opérative est utilisé pour l'extraction des caractéristiques de la partie contrôle. Il est décrit en langage PASCAL conjointement avec le langage IRENE (ref. 10) sous forme de commentaires PASCAL pour lever certaines ambiguités dues au problème de correspondance entre le langage PASCAL et la réalisation matérielle du circuit.

L'outil MACSIM (ref. 3) permet d'extraire de cette description (sous une forme directement traduisible en matériel) les contenus des PLA de la partie contrôle.

4.2.1- Notions sur MACSIM

Cet outil a pour but d'une part, de valider par des simulations le chemin de données défini pendant la conception de la machine informatique et, d'autre part, de dégager par la compilation de la description les caractéristiques de la partie contrôle.

Son principe est basé sur l'analyse des mots-clé. On utilise deux types de mots-clé dans la description des algorithmes:

a) mots-clé de séquencement :

On fait appel à la procédure CYCLE pour désigner le début d'un cycle (un cycle étant la phase élémentaire d'une horloge).

Le mot-clé CODOP suivi du code opération se trouve au début de chaque instruction.

Le mot FININST indique la fin de l'instruction.

b) mots-clé opératifs :

Chaque mot correspond à un champ de commandes codées ou à une commande simple de la partie opérative. Il est représenté par une chaîne de caractères suivi du signe d'affectation en PASCAL.

Exemple : Description des deux premiers états communs à toutes les instructions (voir 2.1.4.3) pour être compilée par MACSIM.

```
CYCLE(param,1);
busa:=PCL; U1:=busa; U2:=0; opual:=U1+U2;
busg:=DB;
if ITVALID then begin cin:=0; busb:=14 end;
           else begin cin:=1;
                      if AP2L3+AP2L2=1 then busb:=busg
                                       else busb:=ROM(PC)
                end;
(* cin:=0 if ITVALID else 1 *)
(* busb:=14 if ITVALID else busg if AP2L3+AP2L2=1 else ROM(PC) *)
RI:=busb; W:=busb;
P2:=AP2L;
CYCLE(param,2);
busb:=opual; PCL:=busb; RT:=busb;
```

```
busg:=busb;
if AP2L3+AP2L2=1 then BUS:=busg;
(* BUS:=busg if AP2L3+AP2L2=1 *)
OVF:=cout;
P2:=AP2L;
CYCLE(param+1000,1);
busa:=PCH; U1:=busa; U2:=0; cin:=OVF; opua1:=U1+U2;
if BS then busb:=18
      else busb:=00;
(* busb:=18 if BS else 0 *)
W73:=busb;
CYCLE(param+1000,2);
busb:=opua1; PCH20:=busb;
busg:=busb; AP2L20:=busg;
```

Remarque : On utilise en commentaires le langage IRENE.

L'analyse de la description se fait donc par la recherche de mots-clé en parcourant le texte de gauche à droite. La démarche utilisée consiste à regrouper les cycles élémentaires qui ont une caractéristique commune et qui permettent de former ainsi une étape englobante définie par cette caractéristique.

4.2.2- Architecture de la partie contrôle

L'outil MACSIM est fait dans l'esprit d'accueillir une structure multi-PLA avec un générateur de temps qui a été retenue comme étant la meilleure approche pour la conception des parties contrôles (ref. 15). En effet, cette solution est la mieux adaptée pour le 8048 puisque toutes les instructions se déroulent en 5 ou 10 états.

L'architecture de la partie contrôle sera constituée de deux PLA avec un générateur de temps. On aura alors:

- Un PLA de propriétés qui est en réalité un décodeur. Il reçoit en entrées les 8 bits du registre d'instruction, 3 signaux ($\overline{\text{IT VALID}}$, EI et TI) provenant du circuit

d'interruptions et 6 compte-rendus de la partie opérative dont:

* 2 pour la correction décimale (voir 3.1.7)
* 1 pour le test du zéro d'un registre (voir 3.1.8)
* 1 pour le test d'un bit d'un registre (voir 3.1.8)
* 1 pour la bascule BS (voir 2.1.2)
* 1 pour l'extension de la ROM (voir 2.1.3.3)

La matrice "ET" génére 200 monômes et la matrice "OU" 117 propriétés en sorties dont un fil servira au séquencement et indiquera s'il s'agit d'une instruction à un cycle ou à deux cycles. Il sera directement connecté au générateur d'instants.

- Un PLA de génération des commandes de MEALY dans lequel les propriétés sont combinées avec les instants pour donner des commandes. Il aura donc en entrées 116 propriétés et les 10 sorties du générateur d'instants. La matrice "ET" génére 171 monômes pour la matrice "OU" qui donne 121 commandes en sorties.

Les contenus de ces deux PLA sont générés par le système MACSIM sous une forme exploitable par le système PAOLA (ref. 4) qui permet le dessin des PLA avec une optimisation topologique.

D'autre part, certaines commandes reçues du second PLA sont paramétrées et donc doivent être encore traitées à travers des PLA ou des décodeurs avant de les envoyer vers la partie opérative. Cette partie sera constituée de plusieurs petits PLA et recevra les 8 bits du code opération latéralement. Elle est conçue pour le moment manuellement.

Finalement, les commandes sont validées par les signaux d'horloge T1 et T2 au niveau des amplificateurs de commandes qui constituent l'interface entre la partie opérative et la partie contrôle.

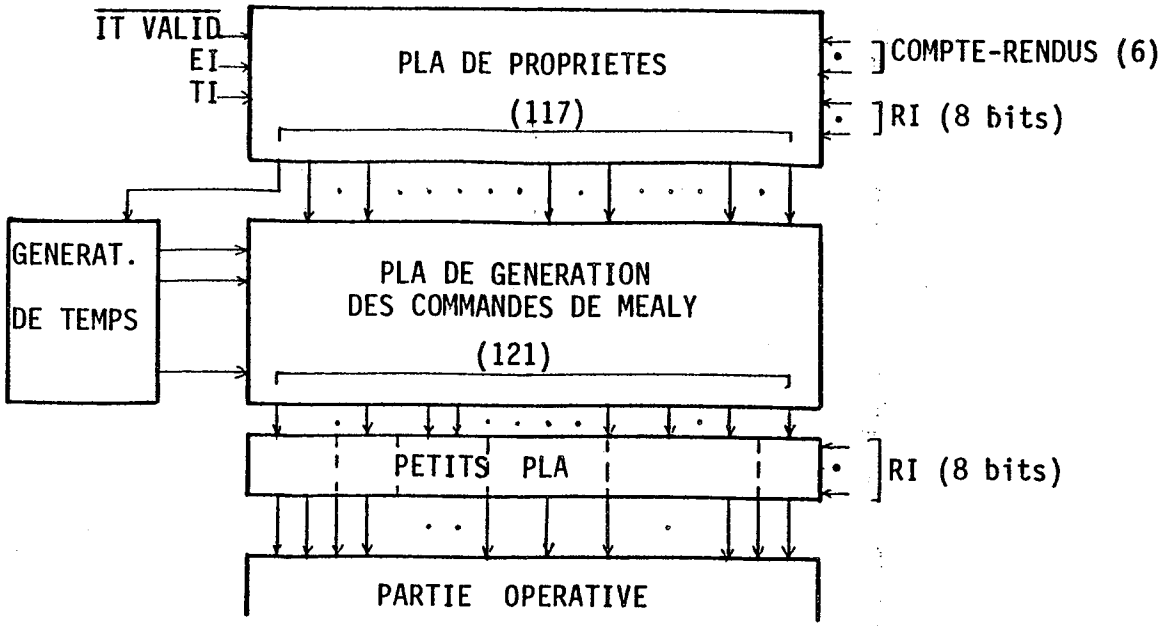


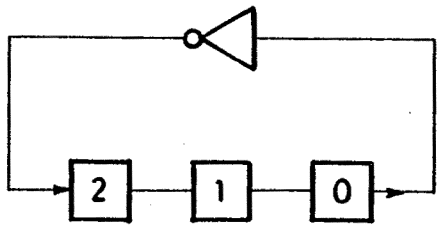
Fig. 4.7- Architecture de la partie contrôle

4.2.3- Réalisation du générateur de temps

Le générateur d'instants sera réalisé à l'aide d'un compteur JOHNSON. Celui-ci est constitué essentiellement d'une chaîne de bascules bouclée par un inverseur. Pour n bascules, on pourra définir 2n états. Chaque état est déterminé par le décodage de 2 bits parmi n.

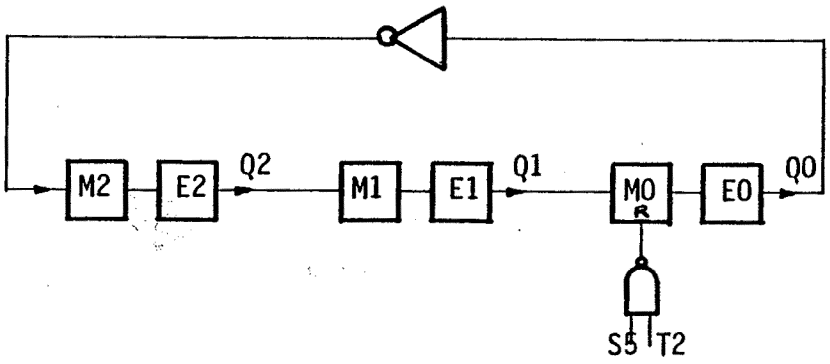
Dans notre cas, on aura besoin de trois paires de bascules maître-esclave pour générer cycliquement cinq états successifs S1, S2, S3, S4 et S5. Pour une instruction à deux cycles, on doit combiner ces états avec le fil extrait du PLA de propriétés (qui permet de différencier entre les instructions à un et à deux cycles) pour former les cinq états S6, S7, S8, S9 et S10 du second cycle.

A la fin de l'état S5, on revient à l'état S1 par une remise à zéro des bascules. En effet, il suffit d'envoyer une commande de "reset", issue du fil signalant l'état S5, à la bascule maître du bit 0 pendant la phase T2 de cet état.



Etat 1	0	0	0	$\overline{Q0}$ $\overline{Q2}$	$\overline{S1}$
Etat 2	1	0	0	Q2 $\overline{Q1}$	$\overline{S2}$
Etat 3	1	1	0	Q1 $\overline{Q0}$	$\overline{S3}$
Etat 4	1	1	1	Q0 Q2	$\overline{S4}$
Etat 5	0	1	1	$\overline{Q2}$ Q1	$\overline{S5}$
Etat 6	0	0	1	Etat non utilisé	

a) Principe du compteur Johnson (3 bascules)



b) Réalisation physique (3 paires de bascules maître-esclave)

Fig. 4.8- Génération des états S1 à S5

4.2.4- Evaluation de la partie contrôle

On utilise des PLA en CMOS semi-statique pour la partie contrôle. Comme dans un PLA NMOS, les matrices "ET" et "OU" seront constituées d'un réseau de transistors N en parallèle. La seule différence est que leurs lignes de sortie sont préchargées chacune par un transistor P et évaluées grâce au transistor N relié à la masse. De plus, elles possèdent un circuit de maintien du "1" réalisé par un système bouclé d'un inverseur et d'un transistor P (voir fig. 4.9).

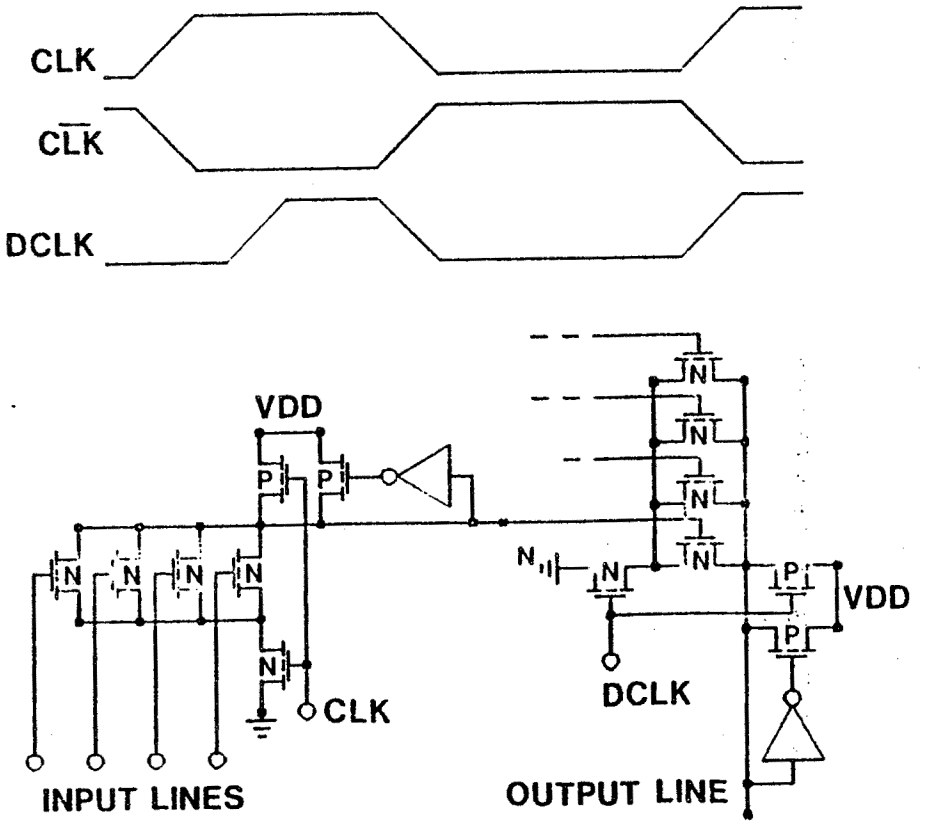


Fig. 4.9- Schéma d'un PLA CMOS semi-statique (ref. 13)
(CLK = phase T1 ou T2)

4.2.4.1- Evaluation d'un PLA

- Dessin d'un point PLA

On définit un pas d'entrée PE (pour des entrées complémentées) et un pas de sortie PS pour un point PLA (voir fig. 4.10).

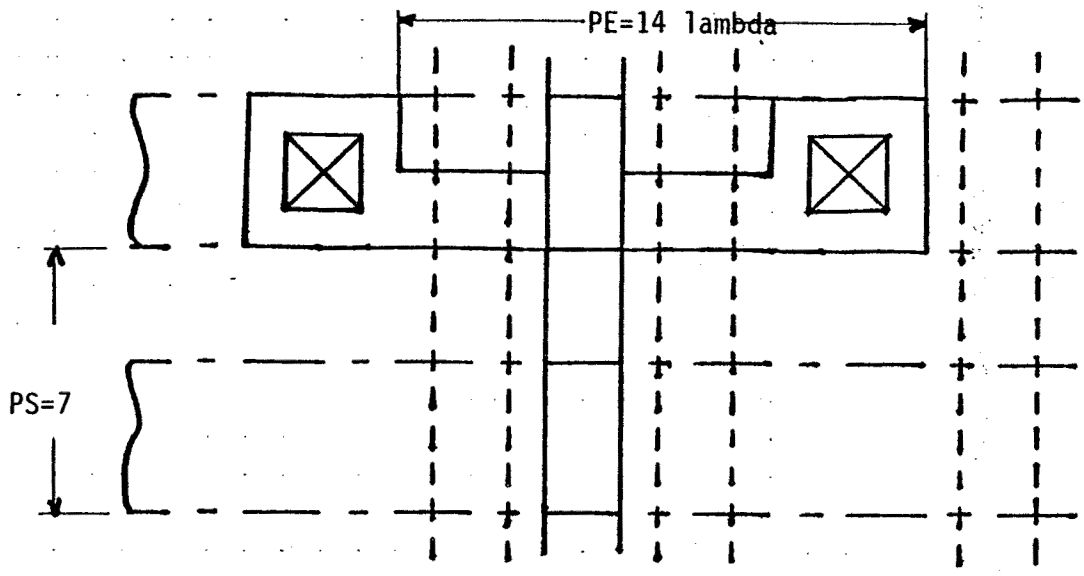
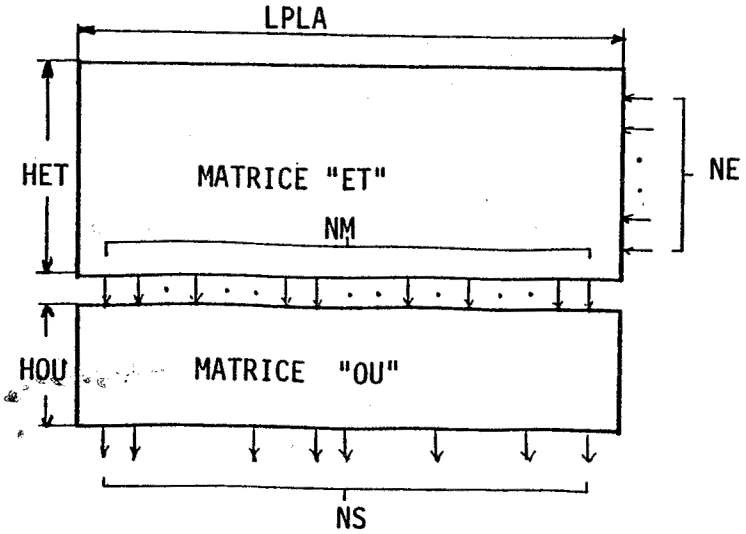


Fig. 4.10- Point PLA (dessin en lambda)

Ces deux paramètres dépendent des règles de dessin. Pour une technologie de 2,5 microns, on a PE = 22 um et PS = 11um.

Soient NE, NS et NM respectivement les nombres d'entrées, de sorties et de monômes d'un PLA.
A l'aide de ces paramètres, on peut exprimer facilement les dimensions des matrices "ET" et "OU".



- Dimensions de la matrice "ET"

Hauteur : HET = NE x PE
Largeur : LPLA = NM x PS + (NM/10) x PS
Le second terme de cette dernière équation tient compte des rapppels de masse qu'on doit ajouter toutes les 10 sorties (de la matrice "ET"). On a:

$$LPLA = 1,1 \times NM \times PS$$

- Dimensions de la matrice "OU"

Hauteur : HOU = NS x PS + (NS/10) x PS = 1,1 x NS x PS
Le coefficient 1,1 représente le facteur de rappel de masse.

Dans un PLA à sorties brisées, la hauteur de la matrice "OU" est exprimée en fonction de son nombre de niveaux NN qui est généralement inférieur à son nombre de sorties.

$$HOU = 1,1 \times NN \times PS$$

Remarque : la largeur de la matrice "OU" est identique à celle de la matrice "ET".

4.2.4.2- Application aux PLA de la partie contrôle

a) PLA de propriétés : (NE=17, NS=117, NM=200)

HET1 = 17 x 22 um = 374 um
LPLA1 = 1,1 x 200 x 11 um = 2420 um

Le PLA de propriétés est en quelque sorte un décodeur et donc sa matrice "OU" est réduite à un seul niveau. D'où:

HOU1 = 1,1 x 1 x 11 um = 12 um

Remarque : la largeur du PLA est du même ordre de grandeur que la longueur de la partie opérative (2520 um).

b) PLA de commandes : (NE=116+10=126, NS=121, NM=171)

On peut optimiser la matrice "ET" en hauteur en brisant les lignes d'entrées qui proviennent du PLA précédent. A chaque ligne brisée on associe ses monômes.

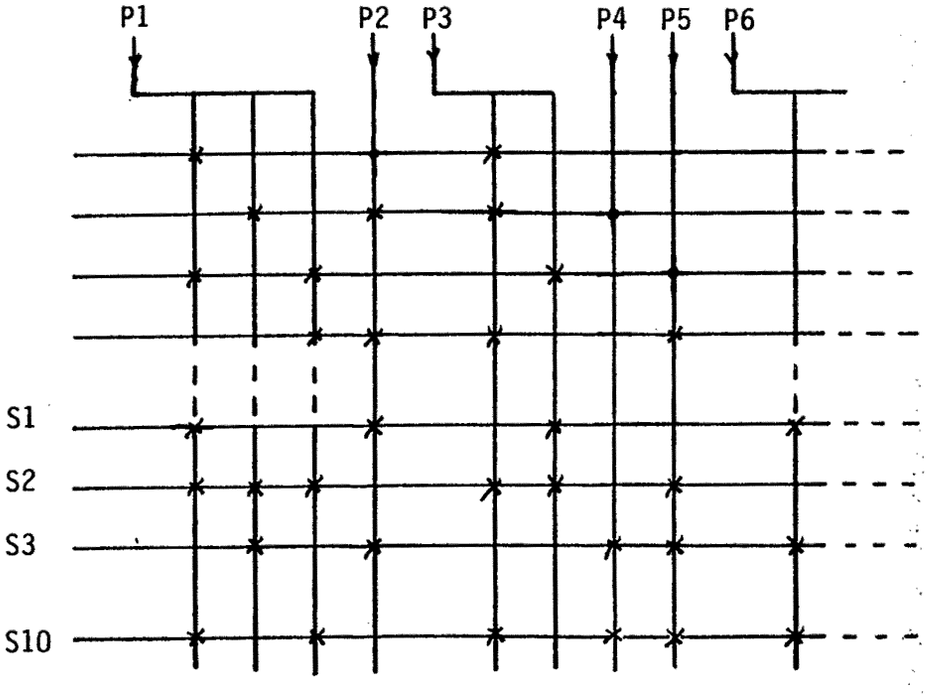


Fig. 4.11- Matrice "ET" du second PLA

La largeur du PLA sera donc proportionnelle à la somme des monômes et des lignes brisées.

LPLA2 = 1,1 x (NM + NB) x PS
où NB désigne le nombre d'entrées briseés.

Pour obtenir une largeur de même taille que la longueur de la partie opérative (2520 um), on est limité à avoir l'équivalent de

208 monômes en sorties; c'est à dire que :

NM + NB = 208
D'où : NB = 208 - 171 = 37

On pourra donc avoir 37 entrées brisées qu'il faut considérer comme une seule entrée lors de l'évaluation de la matrice "ET". D'autre part, ses entrées n'étant pas complémentées, on prendra la moitié du pas d'entrée PE. On aura alors :

NE = (116 - 37) + 1 + 10 = 90

HET2 = 90 x 22/2 = 990 um

La matrice "OU" aura des sorties brisées. On peut supposer que son nombre de niveaux sera égal à 30% de son nombre de sorties. En effet, c'est le pourcentage le plus élevé, obtenu (ref. 17) sur des PLA optimisés par le système PAOLA. La hauteur de cette matrice sera donc :

HOU2 = 1,1 x 121 x 0,3 x 11 = 439 um

La largeur du second PLA est :
LPLA2 = 1,1 x 208 x 11 = 2517 um

c) Petits PLA

On considère ces petits PLA comme un grand PLA unique qui a des entrées brisées, provenant du PLA des commandes, sur un seul niveau. Il reçoit aussi les 8 bits du code opération latéralement. Sa largeur étant la même que celle du second PLA, il nous reste à évaluer la hauteur de la matrice "ET".

HET3 = 8 x 22 + 1 x 22/2 = 187 um

La matrice "OU" est à un seul niveau puisque les petits PLA sont des décodeurs.

HOU3 = 12 um

4.2.4.3- Surface de la partie contrôle

On évalue la surface des PLA de la partie contrôle (SPC) sans les périphériques.

SPC = (HET1 + HOU1 + HET2 + HOU2 + HET3 + HOU3) x LPLA2
SPC = (374 + 12 + 990 + 439 + 187 + 12) x 2517 = 2014 x 2517

SPC = 5,07 mm2

4.3- RESULTATS FINAUX

La surface totale comprenant la partie opérative et la partie contrôle sera:

STOT = 1,74 + 5,07 = 6,81 mm2

Cette surface a été évaluée pour une technologie de 2,5 microns ayant les caractéristiques suivantes:

- Pas d'aluminium x Pas de polysilicium = 56 um2
- Surface du point PLA = 242 um2 (22umx11um)

Afin de pouvoir comparer la surface obtenue précédemment avec celle du circuit original d'Intel 8048 réalisé avec une technologie NMOS de 6 microns, il va falloir extrapoler ces résultats à une technologie équivalente.
Les analyses microphotographiques du 8748 ont montré que :

- Pas d'aluminium = 14 um (13,8 à 14,2um)
- Pas de polysilicium = 14 um (d'après ref. 5)
- Surface du point PLA = 648 um2 (d'après ref. 12)

Par extrapolation, on obtient les résultats suivants :

- Surface de la P.O. = 1,74 x (14 x 14) / 56 = 6,09 mm2
- Surface de la P.C. = 5,07 x 648 / 242 = 13,58 mm2
- Total des surfaces = 19,67 mm2

On remarque que la surface de la partie contrôle est le double de celle de la partie opérative. Elle pourra être réduite car le second PLA (PLA de génération des commandes de MEALY) étant creux, il peut être optimisé davantage à l'aide du système PAOLA (ref. 4).

Sur le circuit original d'Intel, la partie opérative et la partie contrôle ont été dessinées sur une surface de 9,51 mm2 (4,03mm x 2,36mm) (voir fig. 4.12, partie hachurée).

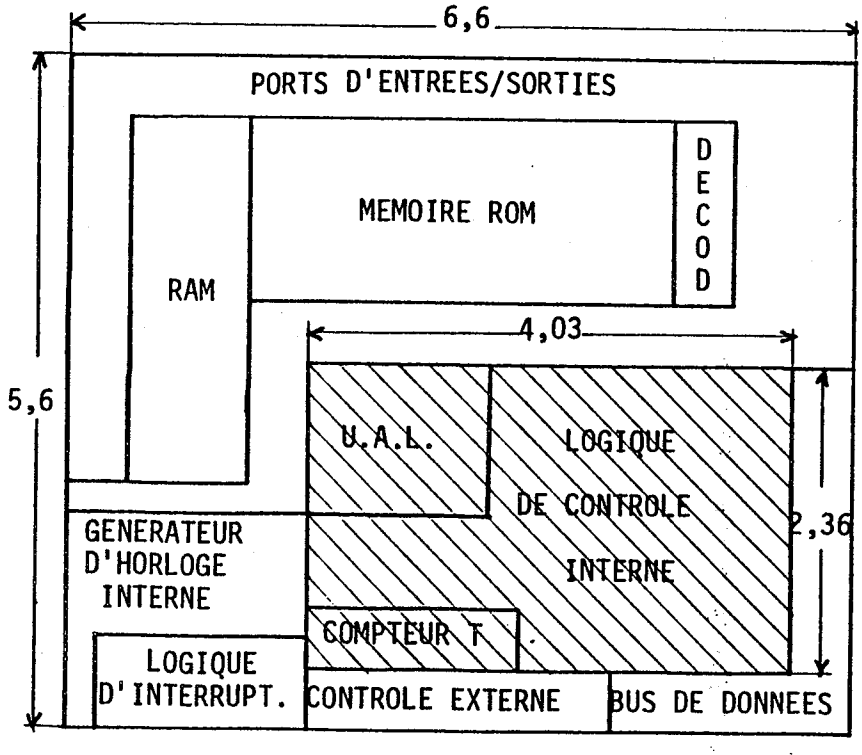


Fig. 4.12- Topologie du 8048 original

La différence de surface entre notre réalisation et celle d'Intel ne s'explique pas uniquement par le fait que la circuiterie CMOS exige plus de transistors que celle du NMOS, mais aussi par la régularité de la conception. Cependant, ceci ne constitue pas un si mauvais résultat puisque notre méthodologie de conception est basée sur une optimisation globale de la surface totale en

minimisant la surface des connexions entre différents modules. On pourra aussi estimer que notre circuit sera réalisable avec la même surface que la version originale du 8048.
En effet, d'après (ref. 12) :

- Surface totale du circuit = 36,96 mm2 (6,6mm x 5,6mm)
- Surface mémoire morte = 3,19 mm2
- Surface mémoire RAM = 1,55 mm2

Surface restante = 36,96 - (19,67 + 3,19 + 1,55) = 12,55 mm2

Cette surface disponible est suffisante pour dessiner les périphériques des PLA de contrôle, le générateur de temps, le circuit d'interruptions, l'horlogerie, les ports d'entrées-sorties et la logique de contrôle externe.

Par ailleurs, la densité du 8048 d'Intel est de 541 transistors par mm2 (pour un total de 20.000 transistors) due essentiellement à un grand pourcentage de ROM et de RAM.
La densité obtenue pour notre partie opérative est de 1085 transistors par mm2 pour une technologie de 2,5 microns. Elle représente le double de celle de la version originale et sera certainement plus élevée pour l'ensemble du circuit puisqu'on a un nombre important de points de PLA, de ROM et de RAM.

- CONCLUSION -

Notre travail a porté sur la conception du micro-ordinateur 80C48. L'intérêt de concevoir un circuit déjà existant sur le marché, réside dans la comparaison que l'on peut faire entre les deux conceptions, en tenant compte du facteur technologique. Nous pouvons ainsi évaluer notre méthode de conception et valider nos outils de CAO tels que LUBRICK, MACSIM et PAOLA.

Nous avons montré comment la décomposition des instructions peut se faire par l'introduction d'une structure à deux bus et d'une horloge à deux phases. La démarche utilisée était manuelle et nous a pris beaucoup de temps en raison de nombreuses redéfinitions de la décomposition, avant d'arriver à la description finale. L'obtention de l'algorithme d'interprétation est primordiale pour la définition de l'architecture de la partie opérative et la réalisation de la partie contrôle.

La partie opérative obtenue posséde une structure régulière de "bit-slice" (ref. 8) et a été implantée en CMOS.

Nous avons proposé une stratégie d'implantation des circuits CMOS qui consiste à avoir des bus de données en polysilicium dans le sens horizontal et des bus d'alimentation ainsi que les commandes en aluminium dans le sens vertical. Cette méthode nous a permis d'adopter une représentation symbolique en "STICK" et de travailler sur une structure ordonnée. L'introduction de la grille, formée de lignes de polysilicium et de colonnes d'aluminium, facilite le passage au dessin réel. En effet, on peut augmenter la largeur W des transistors, jusqu'à trois fois la dimension minimale, sans modifier le squelette de la grille.
Cette stratégie d'implantation semble être un handicap pour la performance du circuit puisque nous avons dessiné les bus de données en polysilicium. Toutefois, des calculs ont donné des temps de propagation de 45 ns pour le bus en polysilicium et de 29,5 ns pour le bus similaire en aluminium. Dans notre cas, ceci ne constitue pas un mauvais résultat puisqu'une phase d'horloge

dure 200 ns pour un cristal de 6 MHz. Cependant, on peut conclure que cette méthode est limitée à des parties opératives qui ne sont pas très longues. Elle est facilement adaptable avec succès à une technologie utilisant deux couches d'aluminium. Dans ce cas, les bus de données ainsi que les connexions entre les deux types de diffusion seront conçus avec la seconde couche de métal.

La partie contrôle a été automatiquement générée à partir de l'algorithme d'interprétation des instructions qui a servi à définir l'architecture de la partie opérative. En effet, l'outil MACSIM permet d'extraire de la description du microprocesseur les contenus des PLA.
Le 80C48 est bien adapté pour accueillir une structure de deux PLA (PLA de propriétés et PLA de commandes) avec un générateur de temps puisque chaque instruction s'exécute en 10 ou en 20 instants. L'interface entre la partie contrôle et la partie opérative se fait à l'aide de plusieurs petits PLA et décodeurs puisque certaines commandes sont paramétrées. Elles seront donc traitées avant d'être envoyées vers la partie opérative.

Finalement, cette étude nous a permis de préciser notre besoin en outils de CAO que l'on doit développer pour faciliter la conception et donc réduire son coût. Cela nous amène à faire les propositions finales suivantes:

1- La décomposition des instructions en un algorithme d'interprétation constitue une étape décisive dans la conception d'un microprocesseur. Elle doit être automatisée pour optimiser les connexions des registres aux bus de la partie opérative, car la démarche manuelle est éprouvante et consomme trop de temps au concepteur.

2- On doit construire une bibliothèque de cellules de base décrites en représentation symbolique pour qu'elles puissent être valables pour n'importe quelle technologie. Il suffit en effet de changer le fichier technologique.

3- La structure multi-PLA avec générateur de temps est la mieux adaptée pour la partie contrôle du 80C48. Mais, pour éviter de rester dans un cadre particulier, il va falloir prévoir la génération automatique d'autres styles de parties contrôles telle que la solution microprogrammée afin de laisser au concepteur le libre choix de conception.

- REFERENCES -

(1) F.ANCEAU
"CAPRI: A Design Methodology and a Silicon compiler for VLSI circuits Specified by algorithms", Third Caltech Conference on VERY LARGE SCALE INTEGRATION, Pasadena, March 1983, pp.15-31.

(2) J.P.SCHOELLKOPF
"LUBRICK: A Silicon Assembler and its Application to Data-path Design for FISC", VLSI 83, Trondheim, NORWAY, August 1983, pp.435-445.

(3) J.P.SCHOELLKOPF
"SILICIEL: Compilation du silicium et Architecture des Circuits Intégrés", Thèse d'état I.N.P.Grenoble en préparation.

(4) S.CHUQUILLANQUI, R.KRASICKI, T.PEREZ SEGOVIA
"A VLSI Topological Optimization Strategy Applied to PLA Design", Proc. of the ICCAD Conference, September 1983, Santa Clara, pp.184-189.

(5) R.REIS
"TESS: Evaluateur Topologique Prédictif pour la Génération Automatique des Plans de masse de Circuits VLSI", Thèse de docteur-ingénieur, I.N.P.Grenoble, Janvier 1983.

(6) INTEL MCS-48TM
"Family of single chip microcomputer user's manual", June 1978.

(7) M.BONNET
"Optimisation de temps de propagation sur les lignes dans les circuits intégrés", Rapport de recherche IMAG N°398, Grenoble, Juin 1983.

(8) C.MEAD, L.CONWAY
"Introduction to VLSI Systems", California, Addison Wesley, 1980.

(9) F.BERTRAND, M.D.SAHBATOU
"Application d'une démarche descendante au 8748", Rapport de DEA, U.S.M.Grenoble, Juin 1982.

(10) S.MARINE, F.ANCEAU, K.JAHIDI
"IRENE: Un Langage de Description pour les Circuits Intégrés", Rapport de recherche IMAG N°356, Grenoble, Mars 1983.

(11) M.D.SAHBATOU
"Rapport technique sur l'implantation de la partie opérative CMOS", Rapport interne, Grenoble, 24 Septembre 1983.

(12) J.M.COSTA ALVES MARQUES
"MOSAIC: Une méthodologie de conception pour les circuits système VLSI", Thèse de docteur-ingénieur, I.N.P.Grenoble, Septembre 1980.

(13) W.D.PRITCHARD
"A CMOS PLA generator", ESSCIRC 82, Brussells, pp.97.

(14) A.D.LOPEZ, H.S.LAW
"A Dense Gate Matrix Layout Method for MOS VLSI", IEEE Journal of Solid States, vol. N°15, pp.736-740, August 1980.

(15) M.OBREBSKA
"Etude Comparative de Différentes Méthodes de Conception des Parties Contrôles des Microprocesseurs", Thèse de docteur-ingénieur, I.N.P.Grenoble, 25 Juin 1982.

(16) A.A.SUZIM
"Etude des Parties Opératives à Eléments Modulaires pour

Processeurs Monolithiques", Thèse de docteur-ingénieur, I.N.P.Grenoble, 6 Novembre 1981, pp.88-90.

(17) T.PEREZ SEGOVIA
"Optimisation en surface des PLA", Rapport de DEA, ENSIMAG/I.N.P.Grenoble, Juin 1980.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to +150°C
Voltage On Any Pin With Respect to Ground ..... -0.5V to +7V
Power Dissipation ..... 1.5 Watt

*COMMENT:
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

D.C. AND OPERATING CHARACTERISTICS T_A = 0°C to 70°C, $V_{CC} = V_{DD}$ = +5V ±10%*, V_{SS} = 0V

Symbol	Parameter	Limits Min.	Typ.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except XTAL1, XTAL2)	-.5		.8	V	
V_{IH}	Input High Voltage (All Except XTAL1,XTAL2,$\overline{RESET}$)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage ($\overline{RESET}$,XTAL1)	3.0		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS, $\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.45	V	I_{OL} = 2.0mA
V_{OL1}	Output Low Voltage (All Other Outputs Except PROG)			.45	V	I_{OL} = 1.6mA
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0mA
V_{OH}	Output High Voltage (BUS, $\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	2.4			V	I_{OH} = 100μA
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = 50μA
I_{IL}	Input Leakage Current (T1, EA, $\overline{INT}$)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OL}	Output Leakage Current (Bus, T0) (High Impedance State)			-10	μA	$V_{CC} \geq V_{IN} \geq V_{SS}$ +.45
I_{DD}	V_{DD} Supply Current		10	20	mA	
$I_{DD} + I_{CC}$	Total Supply Current		65	135	mA	

A.C. CHARACTERISTICS T_A = 0°C to 70°C, $V_{CC} = V_{DD}$ = +5V ±10%*, V_{SS} = 0V

Symbol	Parameter	8048/8748 8035/8035L Min.	Max.	8748-8 8035-8 Min.	Max.	Unit	Conditions (Note 1)
t_{LL}	ALE Pulse Width	400		600		ns	
t_{AL}	Address Setup to ALE	150		150		ns	
t_{LA}	Address Hold from ALE	80		80		ns	
t_{CC}	Control Pulse Width ($\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$)	900		1500		ns	
t_{DW}	Data Set-Up Before $\overline{WR}$	500		640		ns	
t_{WD}	Data Hold After $\overline{WR}$	120		120		ns	C_L = 20pF
t_{CY}	Cycle Time	2.5	15.0	4.17	15.0	μs	6 MHz XTAL (3.6MHz XTAL for -8)
t_{DR}	Data Hold	0	200	0	200	ns	
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In		500		750	ns	
t_{AW}	Address Setup to $\overline{WR}$	230		260		ns	
t_{AD}	Address Setup to Data In		950		1450	ns	
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$	0		0		ns	

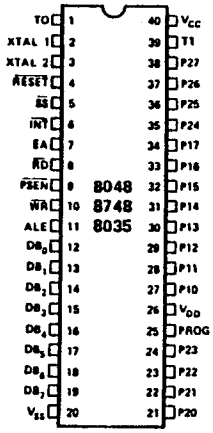
Note 1: Control Outputs: C_L = 80 pF
BUS Outputs: C_L = 150 pF

t_{CY} = 2.5μs

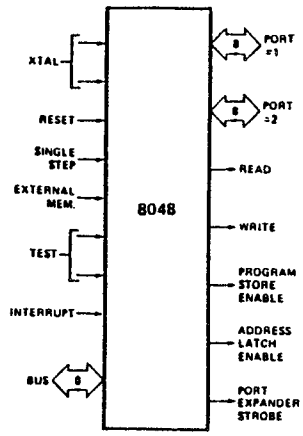
*Standard 8748 and 8035 ±5%, ±10% available.

- ANNEXE B : DESCRIPTION DES BROCHES -

PIN CONFIGURATION



LOGIC SYMBOL



PIN DESCRIPTION

Designation	Pin #	Function
V_{SS}	20	Circuit GND potential
V_{DD}	26	Programming power supply; +25V during program, +5V during operation for both ROM and PROM. Low power standby pin in 8048 ROM version.
V_{CC}	40	Main power supply; +5V during operation and programming.
PROG	25	Program pulse (+25V) input pin during 8748 programming. Output strobe for 8243 I/O expander.
P10-P17 Port 1	27-34	8-bit quasi-bidirectional port.
P20-P27 Port 2	21-24 35-38	8-bit quasi-bidirectional port. P20-P23 contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243
DB_0-DB_7 BUS	12-19	True bidirectional port which can be written or read synchronously using the $\overline{RD}$, $\overline{WR}$ strobes. The port can also be statically latched. Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{PSEN}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{RD}$, and $\overline{WR}$.
T0	1	Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction. T0 is also used during programming.
T1	39	Input pin testable using the JT1, and JNT1 instructions. Can be designated the timer/counter input using the STRT CNT instruction.
$\overline{INT}$	6	Interrupt input. Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction. (Active low)

Designation	Pin #	Function
$\overline{RD}$	8	Output strobe activated during a BUS read. Can be used to enable data onto the BUS from an external device. Used as a Read Strobe to External Data Memory. (Active low)
$\overline{RESET}$	4	Input which is used to initialize the processor. Also used during PROM programming verification, and power down. (Active low) (Non TTL V_{IH})
$\overline{WR}$	10	Output strobe during a BUS write. (Active low) Used as write strobe to External Data Memory.
ALE	11	Address Latch Enable. This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
$\overline{PSEN}$	9	Program Store Enable. This output occurs only during a fetch to external program memory. (Active low)
$\overline{SS}$	5	Single step input can be used in conjunction with ALE to "single step" the processor through each instruction. (Active low)
EA	7	External Access input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification. (Active high)
XTAL1	2	One side of crystal input for internal oscillator. Also input for external source. (Non TTL V_{IH})
XTAL2	3	Other side of crystal input.

- ANNEXE C : CODE OPERATION DES INSTRUCTIONS -

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	INC &R0	XCH A,&R0	XCHD A,&R0	ORL A,&R0	ANL A,&R0	ADD A,&R0	ADDC A,&R0	MOVX A,&R0	MOVX &R0,A	MOV &R0,A	MOV &R0,#DTA		XCL A,&R0		MOV A,&R0
1		INC &R1	XCH A,&R1	XCHD A,&R1	ORL A,&R1	ANL A,&R1	ADD A,&R1	ADDC A,&R1	MOVX A,&R1	MOVX &R1,A	MOV &R1,A	MOV &R1,#DTA		XRL A,&R1		MOV A,&R1
2	OUTL BUS,A	JB0 ADD		JB1 ADD	MOV A,T	JB2 ADD	MOV T,A	JB3 ADD		JB4 ADD		JB5 ADD		JB6 ADD		JBZ ADD
3	ADD A,#DTA	ADDC A,#DTA	MOV A,#DTA		ORL A,#DTA	ANL A,#DTA			RET	RETR	MOVP A,@A	JMPP @A		XRL A,#DTA	MOV A,PSW	
4	JMP PG 0	CALL PG 0	JMP PG 1	CALL PG 1	JMP PG 2	CALL PG 2	JMP PG 3	CALL PG 3	JMP PG 4	CALL PG 4	JMP PG 5	CALL PG 5	JMP PG 6	CALL PG 6	JMP PG 7	CALL PG 7
5	EN INT	DIS INT	EN TINT	DIS TINT	STRT CNT	STRT T	STOP TCNT	EN TCNT	CLR F0	CPL F0	CLR F1	CPL F1	SEL RB0	SEL RB1	SEL MB0	SEL MB1
6		JTF	JNT0	JT0	JNT1	JT1		JF1	JNI	JNZ		JZE	JZ		JNC	JC
7	DEC A	INC A	CLR A	CPL A	SWAP	DAA	RRC A	RR A		CLR CY	CPL CY		MOV A,PSW	MOV PSW,A	RLA	RLC
8	INS A,BUS	INC @R0	XCH A,@R0		ORL A,@R0	ANL A,@R0	ADD A,@R0	ADDC A,@R0	ORL P0,#DTA	ANL P0,#DTA	MOV P0,A	MOV P0,#DTA	DEC @R0	XRL A,@R0	DJNZ R0	MOV A,P0
9	INS A,P1	INC @R1	XCH A,@R1	OUTL P1,A	ORL A,P1	ANL A,P1	ADD A,@R1	ADDC A,@R1	ORL P1,#DTA	ANL P1,#DTA	MOV P1,A	MOV P1,#DTA	DEC R1	XRL A,@R1	DJNZ R1	MOV A,R1
A	INS A,P2	INC R2	XCH A,R2	OUTL P2,A	ORL A,R2	ANL A,R2	ADD A,R2	ADDC A,R2	ORL P2,#DTA	ANL P2,#DTA	MOV R2,A	MOV R2,#DTA	DEC R2	XRL A,R2	DJNZ R2	MOV A,P2
B		INC P3	XCH A,P3		ORL A,P3	ANL A,P3	ADD A,R3	ADDC A,R3			MOV P3,A	MOV R3,#DTA	DEC R3	XRL A,R3	DJNZ P3	MOV A,R3
C	MOVD A,P4	INC R4	XCH A,R4	MOVD P4,A	ORL A,R4	ANL A,R4	ADD A,R4	ADDC A,R4	ORLD P4,A	ANLD P4,A	MOV R4,A	MOV R4,#DTA	DEC R4	XRL A,R4	DJNZ R4	MOV A,R4
D	MOVD A,P5	INC R5	XCH A,R5	MOVD P5,A	ORL A,R5	ANL A,R5	ADD A,R5	ADDC A,R5	ORLD P5,A	ANLD P5,A	MOV R5,A	MOV R5,#DTA	DEC R5	XRL A,R5	DJNZ R5	MOV A,R5
E	MOVD A,P6	INC R6	XCH A,R6	MOVD P6,A	ORL A,R6	ANL A,R6	ADD A,R6	ADDC A,R6	ORLD P6,A	ANLD P6,A	MOV R6,A	MOV R6,#DTA	DEC R6	XRL A,R6	DJNZ R6	MOV A,R6
F	MOVD A,P7	INC R7	XCH A,R7	MOVD P7,A	ORL A,R7	ANL A,R7	ADD A,R7	ADDC A,R7	ORLD P7,A	ANLD P7,A	MOV R7,A	MOV R7,#DTA	DEC R7	XRL A,R7	DJNZ R7	MOV A,R7

AUTORISATION de SOUTENANCE

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

VU les rapports de présentation de Messieurs

. F. ANCEAU, Professeur

. J.P POTET, Ingénieur

Monsieur SAHBATOU Mohammed

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

Fait à Grenoble, le 26 octobre 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,