Conception descendante appliquée aux microprocesseurs VLSI

François Bertrand

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
«Informatique»

par

BERTRAND François

CONCEPTION DESCENDANTE APPLIQUEE

AUX MICROPROCESSEURS VLSI.

Thèse soutenue le 27 septembre 1985 devant la commission d'examen.

G. MAZARE	Président
F. ANCEAU	
G. NOGUEZ	Examinateurs
J.P. POTET	
J.P. SCHOELLKOPF	

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

****

à mes parents,

à Pascale.

Résumé :

La nécessité d'une conception sûre et descendante de circuits intégrés V.L.S.I. (Very Large Scale Integration) est largement reconnue. Nous nous intéressons ici aux premières étapes d'une telle conception à savoir le passage des spécifications initiales à la définition de l'architecture du circuit. La méthode proposée est une méthode de conception par affinements successifs de spécifications; on distingue trois choix fondamentaux dans la conception de circuits:

- le choix des algorithmes,
- le choix du chemin de données associé aux blocs fonctionnels,
- le choix de la structure de la partie contrôle.

Les validations partielles de conception se feront par analyse et simulation, l'optimisation des choix de conception repose en partie sur une évaluation prédictive.

Abstract:

The objectives of a top-down design methodology of V.L.S.I circuits (Very Large Scale Integration) is to minimize the probability of design error and to optimize the design choices. Such an approach based on classical specification and refinement method is presented here.

It illustrates the architectural choice of complex circuits. The main tool allowing design validation and criteria optimization is the interpret control flowchart. Three main design steps are considered:

- choice of the computation algorithm,
- choice of the data path,
- choice of the control structure.

Analysis and simulation give validation of design steps. Choices are realised upon valuation.

Avant propos:

A l'heure où l'on voit apparaître sur le marché des circuits intégrés de plus en plus complexes, la mise en oeuvre d'outils d'aide à la conception est une nécessité; les circuits logiques complexes voient leur coût se réduire avec l'amélioration des méthodes de conception.

L'objet de cette étude est de valider la méthode de conception descendante CAPRI par la réalisation d'un circuit d'application: le micro-ordinateur 80C48 d'INTEL, en technologie CMOS.

D'autre part, il est nécessaire de préciser que cette étude est le fruit d'un travail d'équipe:

- les outils informatiques (LUCIE, LUBRICK, MACSIM, PAOLA) ont été élaborés au sein de l'équipe de recherche en architecture des ordinateurs de l'IMAG,
- l'implantation du 80C48 a été réalisée par Messieurs M. SAHBATOU, C. MATHERON et P. OBJOIS de la même équipe.

Je ne voudrais pas terminer cet avant propos sans exprimer toute ma reconnaissance à

- Monsieur F. ANCEAU, professeur INPG détaché, qui a bien voulu m'acceuillir au sein de l'équipe de Recherche en Architecture des Ordinateurs de TIM3/IMAG, et sans qui, ce projet n'aurait jamais vu jour. Monsieur ANCEAU est aussi à l'origine des idées de base ennoncées dans ce document,

- Monsieur J.P. SCHOELLKOPF, ingénieur chez BULL Système, pour sa patience et son aide de tous les instants,

- Monsieur J.P.POTET, ingénieur chez MHS, pour la confiance et l'aide apportée durant la réalisation du projet,

- Messieurs G. MAZARE (professeur ENSIMAG/IMAG) et G. NOGUEZ (professeur à l'Université de Paris VI) d'avoir accepté de faire parti de ce jury,

- la societé MHS (Matra Harris Semiconducteur) pour le sujet et l'aide au projet,

- la société BULL Système pour m'avoir permis de rédiger ce document,

- Monsieur IGLESIAS, et avec lui toute l'équipe de reprographie de l'IMAG, pour son travail soigné et sa gentillesse,

- et enfin, tous les membres de l'équipe, et en particulier Monsieur S. CHUQUILLANQUI, pour les nombreuses discussions et encouragements au cours de ces années.

SOMMAIRE:

CHAPITRE I:

ASPECTS METHODOLOGIQUES

CHAPITRE I: Aspects méthodologiques

I 1 Vers une méthodologie de conception structurée

Un circuit intégré monolithique est un assemblage de composants électroniques fabriqués et reliés sur une seule tranche de silicium.

La microélectronique est une activité multi-disciplinaire. Le schéma de la figure I.1 détermine les principales étapes qui constituent l'élaboration d'un circuit VLSI. (Very Large Scale Integration)

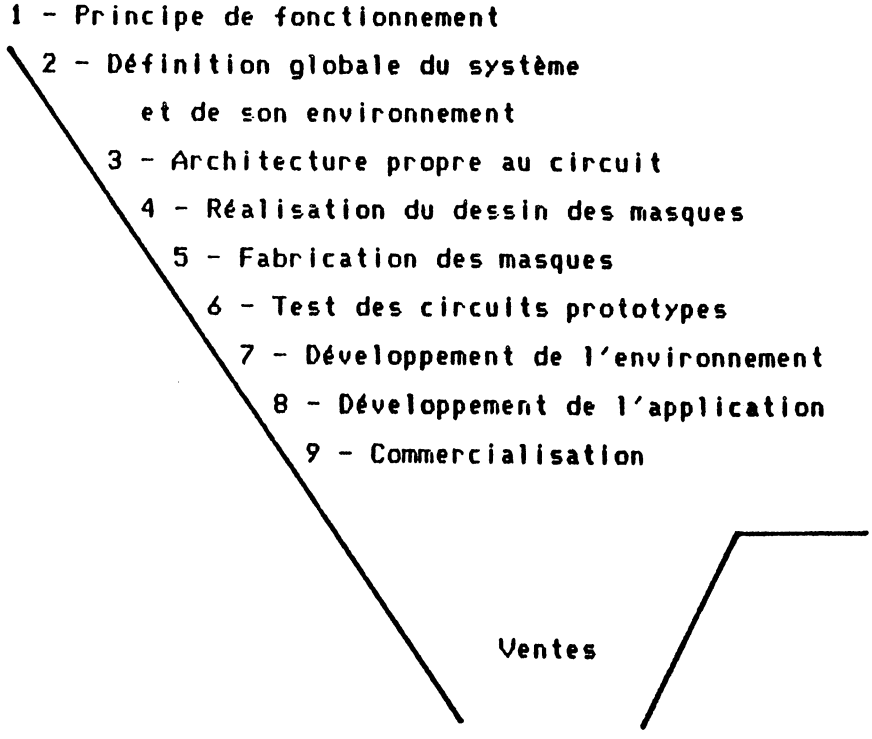


fig:I.1 Principales étapes d'élaboration d'un circuit intégré

Une telle séquence s'appliquerait à beaucoup de fabrications industrielles. Ce qui est remarquable, par contre, c'est l'interaction accentuée entre les différentes étapes et le niveau d'automatisation que l'on peut y appliquer. (simulations, saisies de données, générations automatiques...)

Plusieurs technologies permettent la réalisation de tels circuits.

Elles ont en commun le fait que chaque étape du processus de fabrication est appliquée simultanément à tous les éléments de même nature sans influencer ceux de nature différente; cela implique que l'ensemble des opérations de fabrication requises est indépendant du nombre de composants réalisés.

Les circuits logiques complexes voient leur coût se réduire avec l'amélioration des techniques d'intégration. En fait, le coût de fabrication par porte logique est presque inversement proportionnel à la densité d'intégration. Cela suscite un effort très important de la part des constructeurs pour augmenter cette densité.

Evidemment, cet ensemble de portes doit être organisé pour exécuter la fonction désirée: c'est ce que l'on appelle la conception du circuit. Le travail nécessaire à la conception d'un circuit n'est pas proportionnel au nombre de portes mais à sa complexité logique.

Le besoin d'augmenter la densité d'intégration nécessite de faire appel à une technologie de plus en plus performante. Son développement se traduit:

- d'une part, par l'augmentation régulière de la surface des puces fabriquées; il est actuellement envisageable d'étudier des circuits représentant une surface globale de 1cm2. (On note qu'en parallèle, le nombre de transistors intégrés à surface constante double tous les cinq ans.)

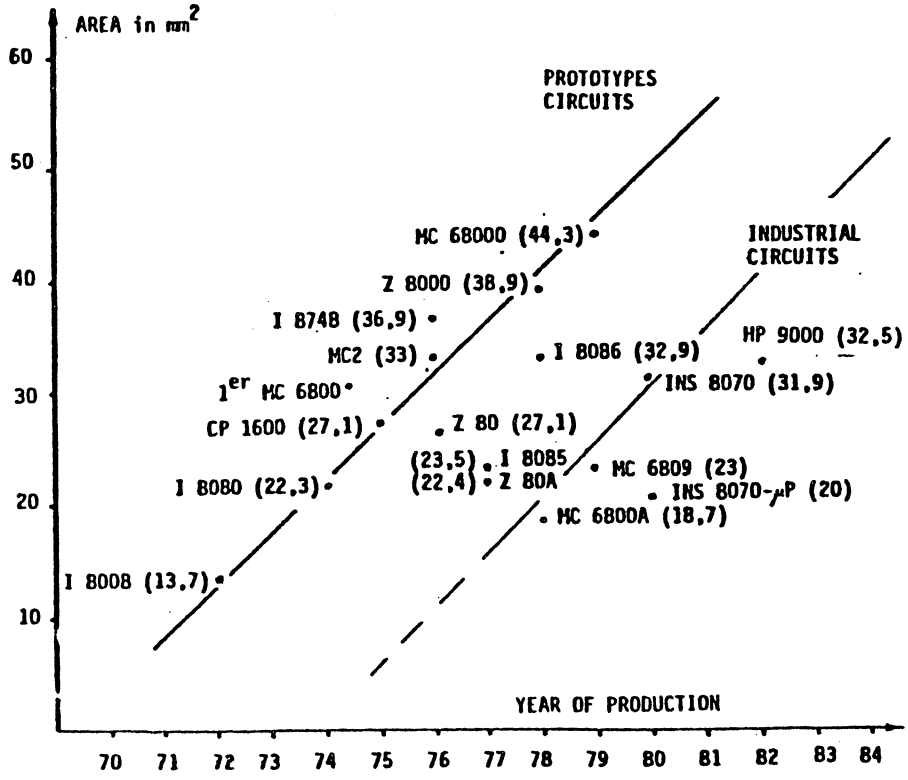


fig:I.2 Evaluation de la surface des microprocesseurs (OBR82)

- d'autre part, par la diminution relative des élements de base; certaines technologies autorisent des grilles de 1 u permettant ainsi d'augmenter la densité de transistors au mm carré.

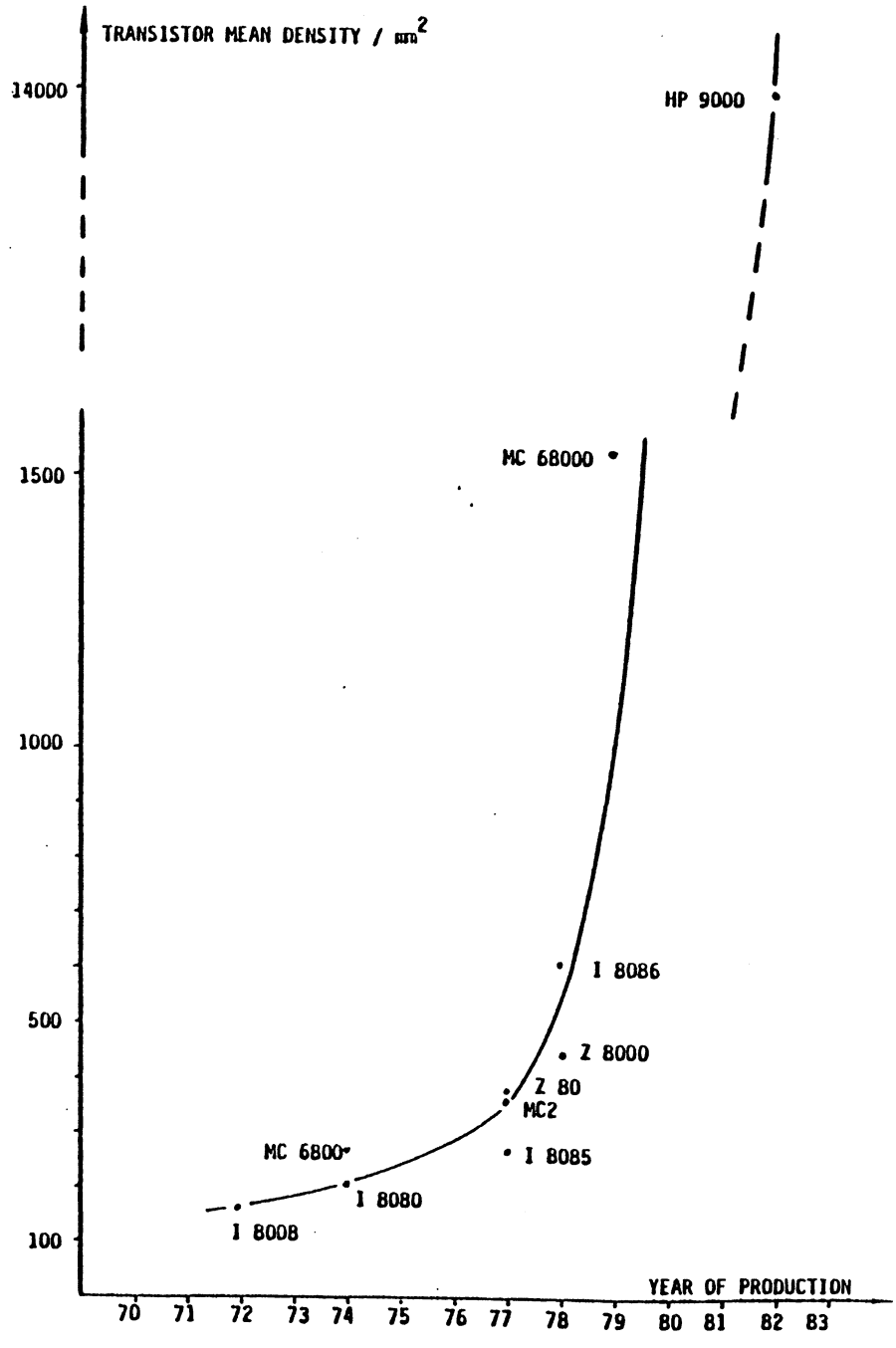


fig:I.3 Evaluation de la densité moyenne (OBR82)

L'intégration d'un nombre de plus en plus grand d'éléments permet une augmentation de la complexité globale des circuits réalisés.

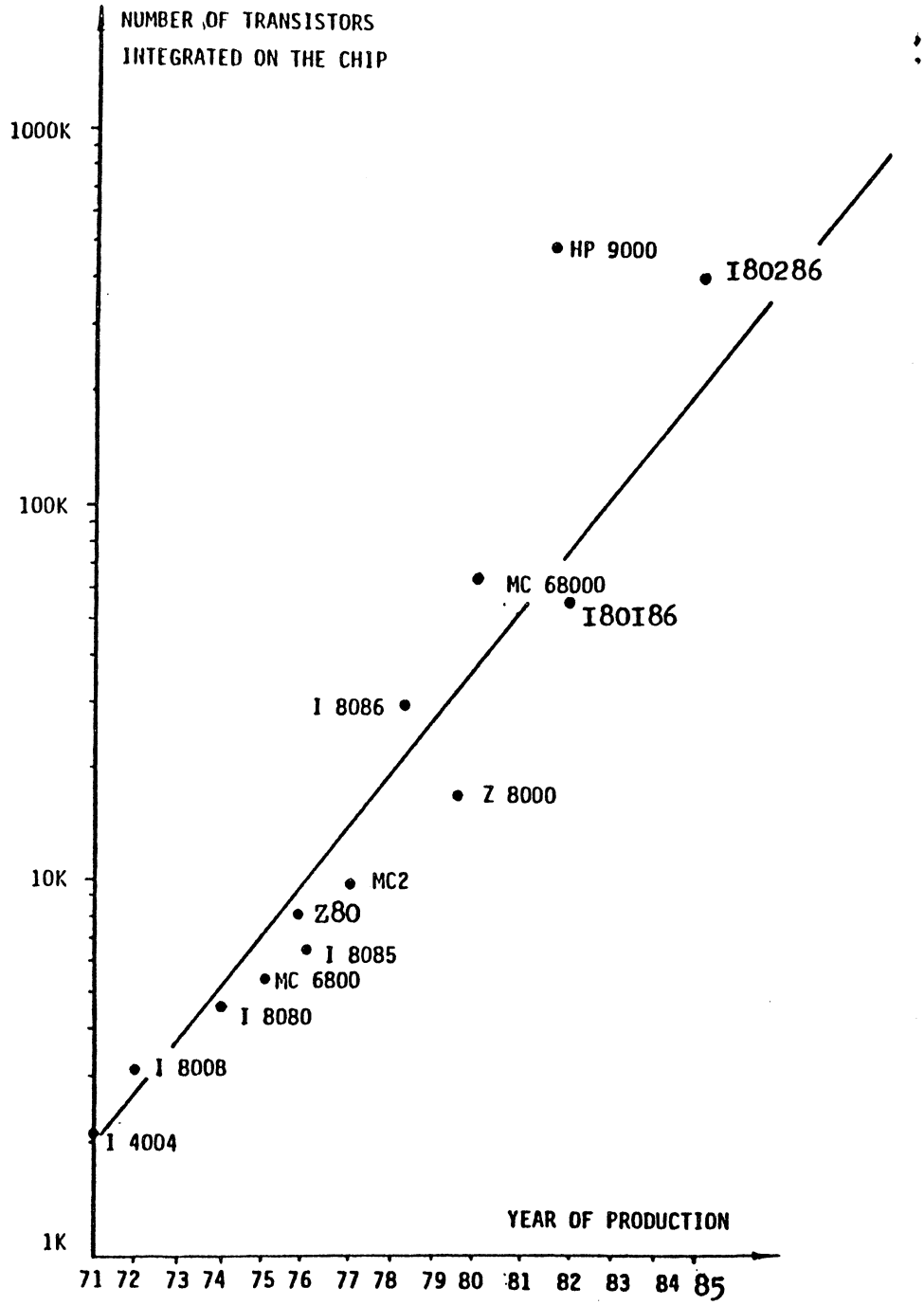


fig:I.4 Evaluation du nombre total de transistors (LAT79)

Dans ces conditions, la tâche du concepteur devient de plus en plus difficile du fait de la quantité considérable d'informations qu'il doit appréhender et manipuler simultanément.

Voulant diminuer le coût de la conception et donc aussi bien la durée d'un projet que le nombre de personnes y travaillant dessus, il faut augmenter l'efficacité des méthodes de conceptions. Ceci se réalise en mettant à la disposition des concepteurs:

a) des outils de plus en plus puissants permettant de manipuler des entités d'un niveau plus élevé que les transistors.

Trois axes de recherche vont dans ce sens:

- utilisation de bibliothèque de blocs pré-dessinés,
- utilisation d'outils de génération automatique de blocs (paramétrés)
- utilisation des outils de dessins symboliques.

b) des méthodes de conception; elles permettent au concepteur d'utiliser une structure de blocs où de sous-blocs fonctionnels intégrant les fonctions puissantes définies grâce à ces outils.

La nécessité d'une démarche de conception sûre et bien structurée des circuits à très haute intégration est devenue une évidence. La définition d'une telle démarche fait l'objet de recherches actives.

Les circuits doivent être:

- d'une part, facilement vérifiables en fin de conception; le coût et le temps de validation de prototype, c'est à dire le temps de détection des erreurs de conception ("debugging time") est en effet un paramètre critique.

- d'autre part, facilement testables en fin de fabrication; la structure du circuit doit permettre le diagnostic sûr et rapide des défaillances physiques.

Les démarches de conception étudiées sont en général des démarches de conception descendante c'est à dire partant des spécifications initiales du circuit (cahier des charges) et allant jusqu'à la description finale du produit. La complexité des circuits et à fortiori des circuits futurs exige une étude minutieuse des choix architecturaux. En effet, si pour les circuits M.S.I. ("Middle Scale Integration"), un concepteur efficace concevait "intuitivement" son schéma logique, l'élaboration d'un microordinateur n'est plus envisageable sans un support informatique.

C'est tout particulièrement à cette étape que nous nous intéressons ici. Une deuxième étape consiste à transcrire les schémas logiques en réseaux de transistors par la génération du dessin des masques.

Il est clair que l'étude architecturale d'un circuit montre l'importance d'un savoir faire antérieur (conception de systèmes

complexes, architectures d'ordinateurs...) et l'on assiste à un "transfert ou une adaptation sur silicium" d'un savoir faire de conception de systèmes.

I 2 Les spécifications initiales ou cahier des charges

Le cahier des charges relatif à un circuit intégré est constitué par des spécifications fonctionnelles (fonctions que le circuit doit réaliser) et par un ensemble de contraintes à respecter.

2-1 Spécifications fonctionnelles

Il s'agit de spécifications de type comportementale du circuit donnant les fonctions "usager" que le circuit offre; cette description est indépendante de la structure interne qui est encore non définie. Ce type de spécification, outre les informations relatives aux fonctions doit préciser les formats des données ainsi que les chronogrammes des entrées/sorties.

Un microprocesseur est spécifié par son jeu d'instructions avec la largeur du chemin de données; on peut remarquer que ces instructions sont exprimées en termes de registres internes et d'opérations élémentaires, laissant supposer que la structure interne est partiellement figée. En fait, il n'en est rien. Car il s'agit là de la définition du circuit vue par l'usager qui ne correspond pas forcément à la réalité matérielle (principe des machines virtuelles en informatique).

2-2 Les contraintes

Les contraintes auxquelles le circuit intégré devra répondre sont variées et dépendent du type de circuit ainsi que de l'application envisagée.

On dénombre:

- les performances exprimées en temps d'exécution des fonctions du circuit,
- la surface et l'encombrement, étroitement liés au coût de fabrication du circuit,
- le coût comprenant le coût de la conception,

- la fiabilité de la conception, liée au nombre de phases de conception et à la régularité de la structure,

- la testabilité; ce mot traduit en fait, d'une part la possibilité d'identifier rapidement une erreur de conception (non conformité aux spécifications initiales), d'autre part, la possibilité de tester et localiser rapidement une défaillance physique dans un circuit. Le premier point concerne la validation du prototype, le deuxième le test de fin de fabrication,

- la consommation et la dissipation d'énergie.

On cherchera a montrer comment une méthode de conception telle que celle proposée ici permet de maîtriser le choix des concepteurs en étudiant les solutions possibles par rapport aux critères ci-dessus.

I 3 Principe de la conception descendante

L'architecture d'une machine doit satisfaire des critères de coût, de service, de performance et de fiabilité.

"Si vous ne savez pas où vous voulez aller, vous avez de fortes chances de vous retrouver ailleurs"

(L.J. PETER, R HULL "Principe de Peter")

On suppose que le style et les paramètres de la réalisation ont été choisis. Ces choix sont établis sur l'expérience acquise lors des évaluations effectuées sur des réalisations existantes.

Deux démarches s'opposent, le but étant toutefois le même: répondre aux spécifications initiales.

3-1 La démarche ascendante:

Elle consiste à regrouper des opérations élémentaires existantes (moyens) pour constituer des opérations de plus en plus complexes permettant d'aboutir aux spécifications initiales désirées par

l'utilisateur.

MOYENS ---> SPECIFICATIONS

Cette technique classique conduit à des calculateurs pouvant être universels, c'est à dire capables de résoudre de nombreux problèmes, mais ceci de façon non optimale.

"Pour aller quelque part, en général, le plus simple était encore de partir de là où l'on voulait aller, et avec du temps et un peu de chance, on y arrivait effectivement"

(J. ROUXEL "Les shadock")

3-2 La démarche descendante:

Elle consiste à partir de spécifications initiales (besoin de l'utilisateur) pour aboutir progressivement à une architecture adaptée (moyens) qui permette la réalisation physique.

SPECIFICATIONS ---> MOYENS

L'évolution technologique des circuits V.L.S.I. (Very Large Scale Integration) en particulier autorise une telle approche pour définir des architectures nouvelles spécialisées et donc mieux adaptées à leur utilisation.

C'est une méthode dite de conception progressive qui consiste à concevoir un système en s'appuyant sur une succession de spécifications obtenues par affinements successifs à partir d'une spécification initiale.

On passe d'une spécification d'un niveau donné à une spécification de niveau inférieur en explicitant (ou interprétant) une ou plusieurs fonctions en termes de primitives plus simples jusqu'à arriver aux primitives matérielles considérées comme entités de base. Une telle méthode est dite de conception sûre si le passage d'un niveau à un niveau plus fin est soit prouvé (méthode complètement sûre) soit

partiellement validé. Il s'agit de démontrer au cours de ce passage qu'aucune erreur ou non conformité aux spécifications initiales n'a été introduite.

"Il est beaucoup plus intéressant de regarder où l'on ne va pas,pour la bonne raison que là où on va,il sera toujours temps d'y regarder quand on y sera"

(J. ROUXEL "Les shadock")

Un passage entre deux niveaux implique, en général, un choix de conception, c'est à dire la décision du concepteur de choisir parmi un ensemble de solutions qu'il est capable d'énumérer. Dans ce cas, on cherche à contrôler ce choix et à l'optimiser par rapport aux contraintes fixées lors des spécifications initiales.

Pour ce faire, un certain nombre d'outils viennent en aide au concepteur pour la détermination de son choix; utilisant une description structurelle ou comportementale, certains outils permettent de décrire le circuit déjà conçu à un niveau de finesse donné (architectural, logique, électrique...) tandis que d'autres autorisent le passage à un niveau plus "fin".

Dans la démarche que nous proposons, trois niveaux de spécification sont considérés, traduisant la prise en compte des choix, à savoir:

- le choix de l'algorithme de calcul (enchaînement de microopérations décrites par un algorithme),
- le choix du chemin de données,
- le choix de la structure de contrôle.

A chaque niveau, la spécification du circuit permet de contrôler et d'optimiser le choix du concepteur.

Il est clair que dans la pratique, cette démarche n'est pas linéaire.

CHAPITRE II:

GENERALITES ARCHITECTURALES

CHAPITRE II: Généralités architecturales

Une machine séquentielle est assimilée à la réalisation d'un algorithme. Partant de sa description, on descend jusqu'à sa réalisation matérielle en appliquant des opérations d'interprétation selon la méthode descendante; l'algorithme se décompose en deux fonctions: le séquencement et les actions.

A ces deux parties, on fait correspondre deux éléments disjoints: la partie opérative et la partie contrôle, dissemblables tant au point de vue topologique que fonctionnel.

Un microprocesseur, comme toute machine informatique, est défini par son langage, c'est à dire le jeu d'instructions qu'il exécute. Le choix d'une réalisation matérielle est déterminé par le concepteur durant la phase de définition de l'architecture interne; cette réalisation concrétise un choix fait parmi de nombreuses implantations possibles succeptibles d'exécuter l'algorithme d'interprétation.(SUZ81)

II 1 Descriptions algorithmiques

Le comportement d'une machine séquentielle peut être décrit sous la forme d'un algorithme "A". L'expression de "A" dans un langage "L", représenté par "A/L" est un programme "P" du langage "L".

Le même algorithme peut être exécuté sur du matériel. La réalisation de "A" sur un matériel "M", représenté par "A//M" est une machine "M".

Un algorithme donné peut avoir différentes formes, aussi bien dans sa réalisation matérielle que dans son expression graphique (programmes dans différents langages, organigrammes,etc...). Ces différentes formes sont représentées dans la figure II.1 où "P1", "P2", "P3"... sont les différents programmes (où expressions graphiques), "M1", "M2", "M3"... les différentes machines. "L1", "L2" sont des langages et "M1", "M2", "M3" les différentes organisations matérielles.

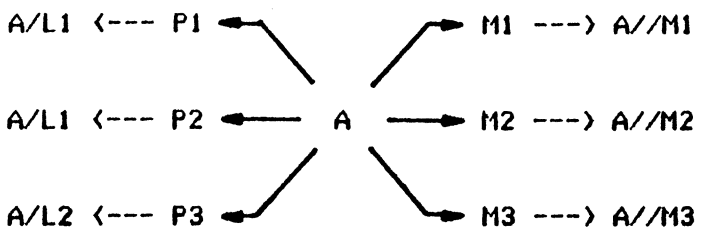


fig:II.1 Réalisations différentes de l'algorithme "A" (SUZ81)

Les langages par lesquels est exprimé l'algorithme "A" constituent des ensembles possibles de ressources. Les ressources d'un langage sont dites des primitives alors que les ressources matérielles sont appelées composants.

Notre but est de construire des machines informatiques, c'est à dire, de réaliser matériellement des algorithmes.

Un circuit dont l'algorithme est exprimé par un programme peut être réalisé directement à partir de ce programme. Si les primitives du langage utilisé sont réalisables sur du matériel, alors une transposition directe est possible.

Lorsque les primitives du langage utilisé sont trop complexes et que leur réalisation directe sur du matériel n'est pas possible (ou pas intéressante), on procède alors à des descriptions de plus en plus fines, réduisant ainsi à chaque niveau la complexité des primitives utilisées. Le dernier niveau atteint ne doit donc avoir que des primitives réalisables sur du matériel. (transferts de registre à registre, opérations appliquées à des variables...)

On définit comme interpréteur "Ii" du langage "Li" un algorithme capable d'exécuter tout programme écrit dans ce langage. Le processus d'interprétation est noté:

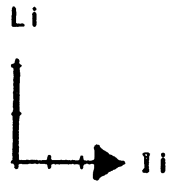


fig:II.2 Interpréteur "Ii" d'un langage "Li"

L'interpréteur lui même, peut être programmé dans un langage "Li-1". Le processus d'interprétation sera répété, s'il le faut, jusqu'à atteindre le niveau de langage désiré.

On obtient ainsi une chaine dont l'interpréteur de niveau le plus bas "Io" peut être réalisé de manière physique. Sa réalisation en est la machine "M" capable d'exécuter, indirectement, les instructions de "Ln" au travers des interprétations imbriquées.

Ainsi l'algorithme d'interprétation des instructions d'un microprocesseur explique la sémantique du jeu d'instructions exécuté. Il est écrit dans un langage dit de définition. Le passage par un ou plusieurs interpréteurs successifs autorise une description (et une réalisation) de la machine
physique.

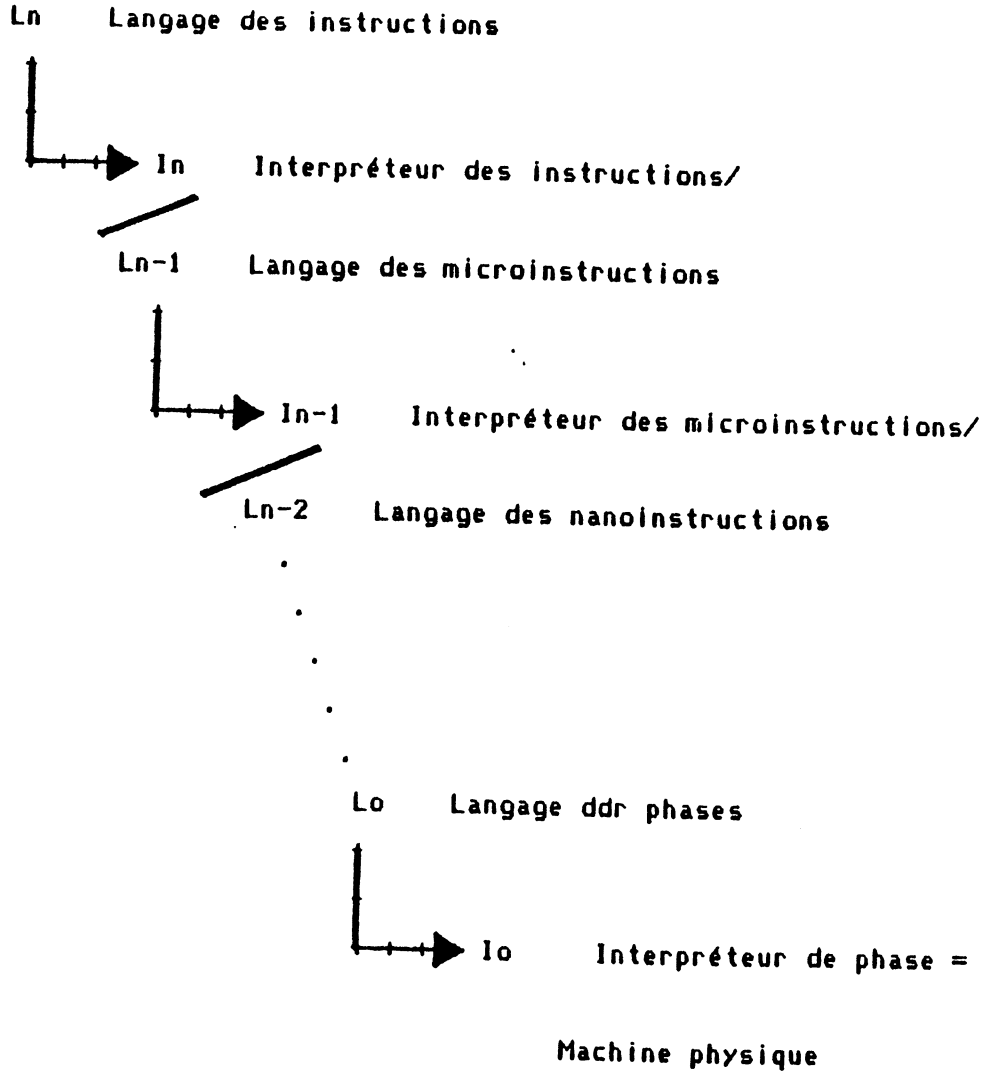


fig:II.3 Exemple de plusieurs niveaux d'interprétation: l'interpréteur des instructions du niveau "i" est écrit dans le langage des instructions du niveau inférieur "i-1". (ANC74, SCH77)

Les primitives d'un langage sont constituées par un ensemble d'opérateurs et de variables.

Pour un microprocesseur, l'algorithme d'interprétation ("interrupt", "halt".etc...exclus) utilise des primitives définies par le jeu d'instructions et les variables (registres, mémoire).

Cet algorithme transforme un ensemble de données (entrées) dans un autre ensemble (sorties). On peut représenter cette structure par le schéma de la figure II.4a ou de façon plus synthétique par celui de la fig II.4b. Dans ce cas, les variables sont internes et représentent la logique associée à la réalisation des opérations.

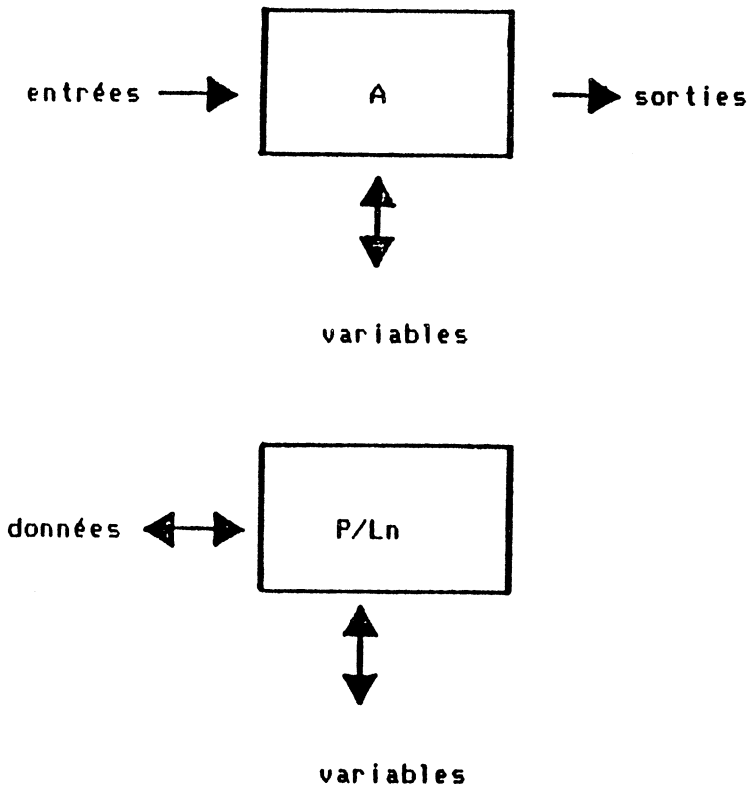


fig:II.4 Un algorithme et son environnement: a) et b). (SUZ81)

L'algorithme, exprimé sous la forme d'un programme écrit dans un langage "Ln", est noté "A/Ln". Un interpréteur de "Ln" doit accéder au programme écrit en "Ln" ainsi qu'aux variables et données qu'il manipule. Le programme, les variables et les données constituent des données pour l'interpréteur. L'interpréteur utilisera en plus de ces données des variables de travail internes. Ce phénomène se répète tout au long des

niveaux d'interprétation, ce que l'on schématise par la figure:

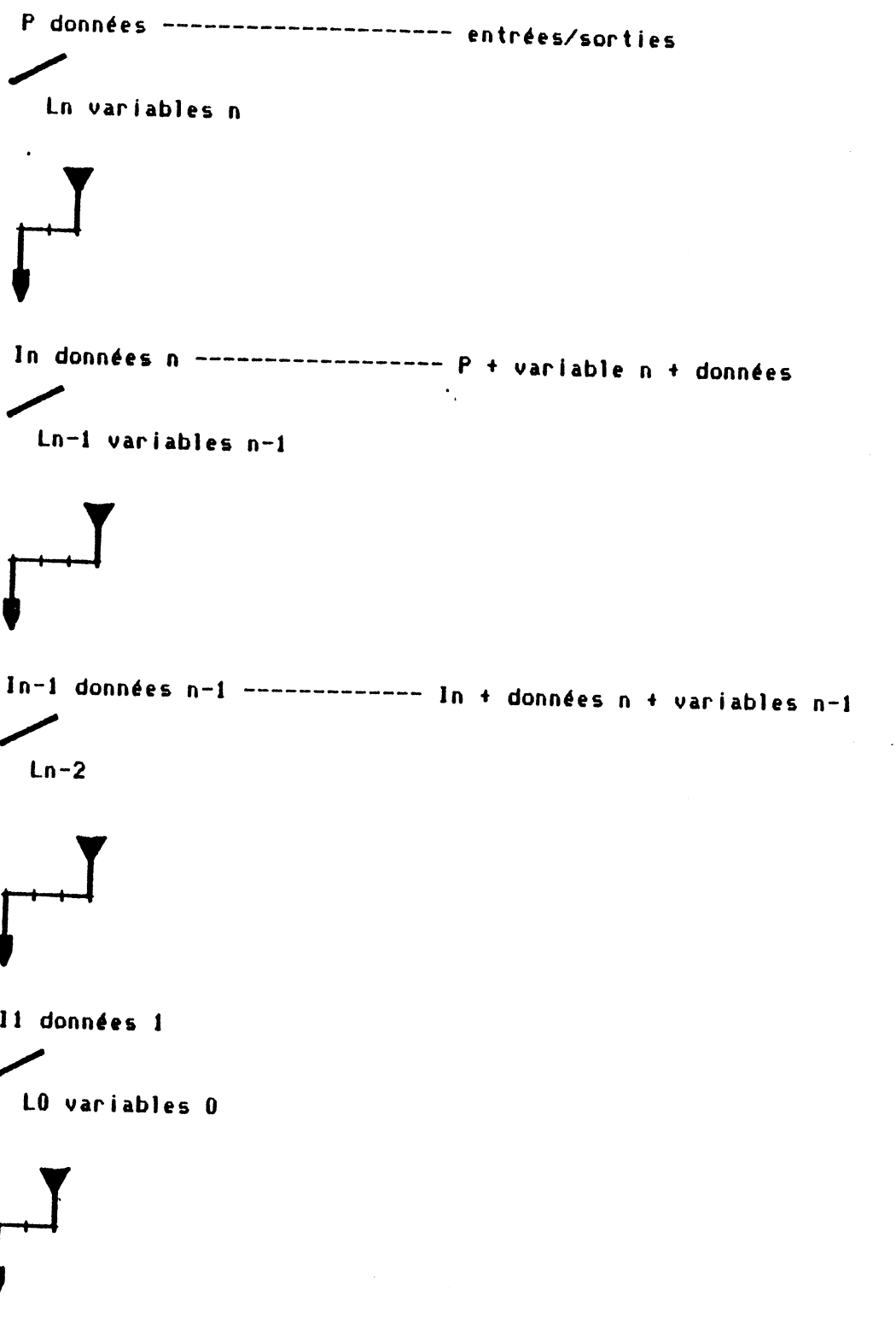


fig:II.5 Enchaînement des niveaux d'interprétation. (SUZ81)

"L0" est un langage de description utilisant des primitives matérielles, "I0" l'interpréteur physique associé. "I0" est figé; il n'interprète donc pas tous les programmes écrits en "L0" mais seulement un seul.

A chaque niveau d'interprétation correspond un langage dont les primitives sont plus élémentaires, permettant ainsi une description plus fine des actions.

Les limitations relatives au matériel sont supposées connues du concepteur; la fonctionnalité et la faisabilité de chaque bloc, tant au point de vue logique qu'électrique nécessite une remise en question permanente des options requises.

II 2 Notion de partie controle et de partie opérative

Les machines séquentielles en général et les processeurs tout particulièrement peuvent être décomposés en deux parties: une partie contrôle (P.C.) et une partie opérative (P.O.). Cette décomposition est faite, non seulement du point de vue fonctionnel mais aussi topologique.(ANC82, OBR82)

2-1 Partie opérative

La partie opérative contient les ressources de base pour le traitement des données et des adresses. Elle comprend:

- le stockage des données (mémoire, registre, bascules),
- l'acheminement des données (bus),
- la transformation des données par des opérateurs (UALs, incrémenteurs/ décrémenteurs, circuits d'ajustement décimal...).

Les opérations admises par l'interpréteur permettent de déterminer les opérations nécessaires tandis que les affectations définissent les chemins de données reliant les différents blocs fonctionnels.

A chaque niveau d'interprétation correspond une partie opérative mais seule celle du niveau le plus bas existe en tant que structure physique;

cette structure dépend des choix pris à des niveaux supérieurs face à la nécessité du compromis entre le coût en matériel et la vitesse requise. L'exemple classique concerne le choix de la taille d'une UAL (Unité Arithmétique et Logique) par rapport à la taille des données traitées.

Une partie opérative ainsi déterminée peut être plus où moins bien adaptée à l'exécution d'un jeu d'instruction considéré. Sa structure permet potentiellement l'exécution d'un certain nombre d'opérations, de transferts et de mémorisations à une vitesse déterminée; cette vitesse se définit en terme de:

- temps de cycle nécessaire à l'exécution d'une opération élémentaire (ex: addition..),
- temps de transfert (lié à des critères de structure et de technologie).

2-2 Partie contrôle

La partie contrôle a pour rôle de piloter la partie opérative en lui envoyant les commandes relatives à l'exécution des instructions.

Elle est définie à partir de l'organigramme de l'algorithme d'interprétation; il s'agit en fait d'un automate de contrôle qui, d'une part, envoie les commandes vers la partie opérative pour lui indiquer les opérations à effectuer à chaque étape, et d'autre part, assure la progression de l'exécution d'une étape à l'étape suivante.

De plus, pour exécuter le séquencement, la partie contrôle à besoin de connaître l'état de la partie opérative; des mots d'états et des prédicats calculés (test) sont générés à cet effet.

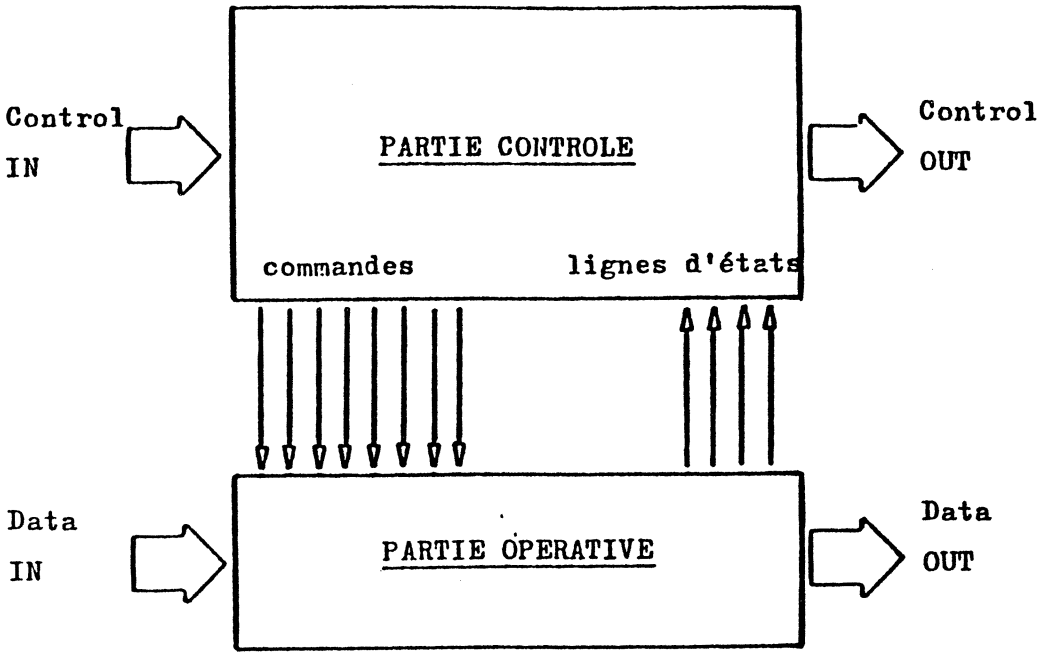


fig:II.6 Architecture interne d'un microprocesseur

Contrairement à la définition de la partie opérative, celle de la partie contrôle peut, mais ne doit pas forcément être définie à partir de l'interpréteur du niveau le plus bas. En général, la spécification d'une partie contrôle résulte de l'empilement des spécifications de tous les niveaux de la partie contrôle. De plus, a un niveau donné, le même algorithme peut être décrit de différentes façons, impliquant des implémentations matérielles différentes. (OBR 82)

CHAPITRE III:

APPLICATION DE LA DEMARCHE DESCENDANTE AU 80C48

CHAPITRE III: Application de la démarche descendante au 80C48

Partant des spécifications initiales, la réalisation de circuits intégrés VLSI est effectuée par application de la démarche descendante "CAPRI".(CAP82)

Son principe est basé sur une architecture modulaire utilisant des blocs paramétrés prédéfinis dont les caractéristiques logiques, électriques et topologiques ont été préalablement testées et validées, assurant ainsi une fiabilité accrue des étapes de conception.(ANC82, ANC83)

Ces blocs correspondent à de véritables fonctions intégrées permettant de diminuer la durée et donc le coût de réalisation de circuits intégrés complexes. L'utilisation de structures prédéfinies, donc moins souples augmente le coût en matériel; méanmoins, cette augmentation est minimisée par l'optimisation maximale du contenu de chaque bloc.

Le processus de réalisation consiste à parcourir les deux étapes principales suivantes:

- détermination de l'architecture et du chemin de données associé,
- détermination de l'unité de contrôle et de séquencement.

Cette démarche est appliquée pour réaliser un microprocesseur huit bits compatible 80C48; il s'agit de définir un CPU constituant le coeur d'un circuit complexe de synthèse de la parole (contrat industriel). Pour ce faire, une extension du jeu d'instructions de base et des entrées/sorties est nécessaire.

III 1 Les spécifications du 80C48

Issue de la famille MCS-48 (Intel), le 80C48 est un microordinateur monoboitier de quarante broches dont les principales caractéristiques sont les suivantes:

- chemin de données de huit bits,

- mémoire morte de 1K octets,
- mémoire vive de 64 octets,
- 27 lignes d'entrées/sorties (dont: trois ports et trois lignes de contrôle),
- compteur et timer de huit bits,
- huit niveaux de sauvegarde,
- deux banques de registres de travail,
- jeu de 96 instructions (de un ou deux cycles machine),
- interruptions à un niveau,
- mode pas à pas,
- temps de cycle de 2,5 microsecondes,
- technologie CMOS,
- alimentation sous cinq volts,
- mode basse consommation,
- horloges externes,
- possibilités d'extension (mémoire externe, entrées/sorties).(INT48)

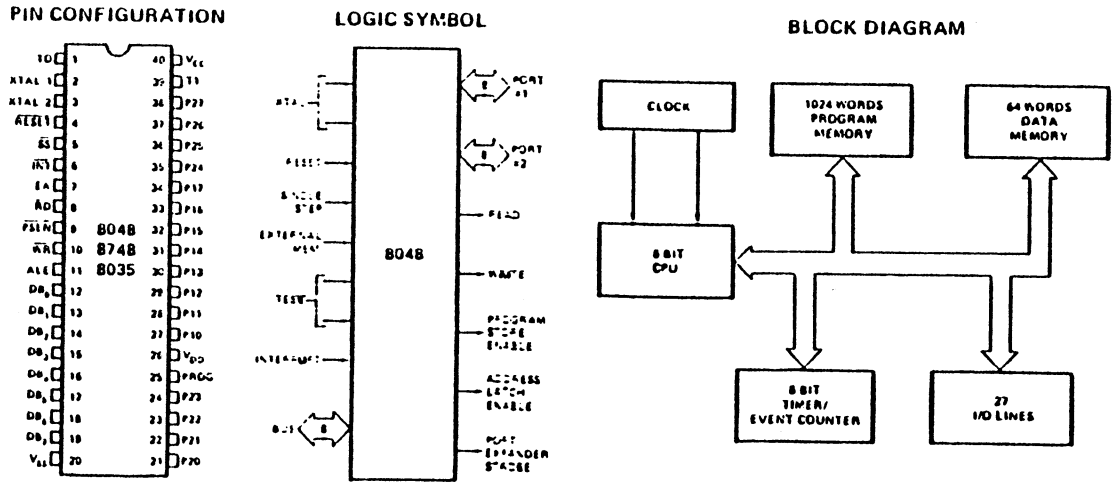


fig:III.1 Symboles logiques/ Blocs fonctionnels. (INT48)

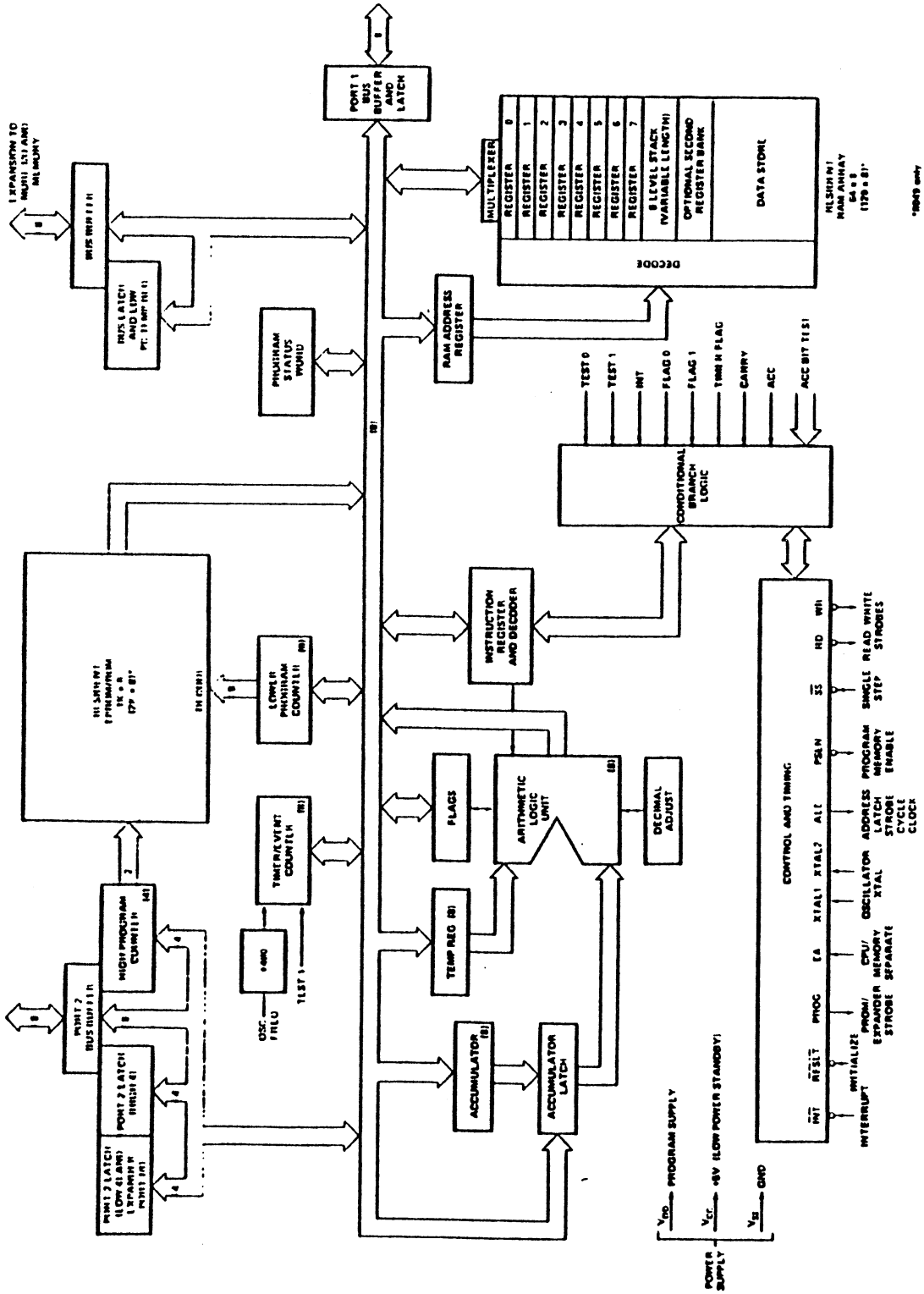
1-1 Eléments architecturaux

Le 80C48 est un microprocesseur huit bits(INT48). Conçu pour effectuer des opérations arithmétiques en binaire et en BCD ("Binary Coded Decimal"), son architecture utilise une structure à un bus. Elle se

compose des blocs fonctionnels suivants:

- une unité arithmétique et logique (UAL),
- des registres de travail (A, AL, TR),
- une mémoire morte (ROM) avec son registre d'adressage(PC),
- une mémoire vive (RAM) avec son registre d'adressage (RAR),
- une unité de contrôle et de séquencement (CAT),
- des registres tampons d'entrées/sorties (I/O),
- une unité de branchements conditionnels,
- un registre d'instructions (RI),
- un mot d'état programme (PSW),
- un compteur/temporisateur (T),
- un registre d'états (FLAGS).

fig:III.2 Architecture du 80C48 (User manuel MCS-48)



a) l'unité arithmétique et logique:

L'unité arithmétique et logique réalise les opérations suivantes:
- additions, soustractions (avec ou sans retenue),
- opérations logiques (ET, OU, OUex, fonction complément),
- incrémentations, décrémentations,
- rotations droite/gauche (avec ou sans report),
- permutations poids-faibles/poids-forts,
- ajustement décimal.

Les opérandes sont de huit bits; un éventuel débordement de la valeur du résultat génère une retenue mémorisée dans le mot d'état programme.

b) les registres de travail:

L'accumulateur "AC", à la fois source et destination de l'UAL, constitue un des registres de travail du microprocesseur. Il est directement accessible à l'utilisateur que ce soit après un accès mémoire ou suite à une opération d'entrée/sortie.

D'autres éléments de mémorisation permettent la sauvegarde de résultats intermédiaires. Il s'agit du "latch" d'accumulateur "AL" et du registre temporaire "TR"; n'étant pas pris en compte au niveau du jeu d'instructions, ils sont transparents vis à vis de l'utilisateur.

c) la mémoire morte:

La mémoire morte résidente est constituée de 1024 mots de huit bits adressés par le compteur de programme "PC"

PC:= PCH // PCL

A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0

fig:III.3 Le compteur de programme (PC)

L'adresse poids-forts de quatre bits "PCH" sélectionne la page courante

tandis que les huits bits de poids-faibles désignent l'un des 256 mots de la page considérée. Le bit le plus significatif "A11" valide l'accès à la mémoire interne ou externe (cf III 1-4a).

Trois adresses importantes sont à signaler:

- adresse "0": elle contient le code opération de la première instruction à exécuter suite à l'activation de la ligne de contrôle "Reset".

- adresse "3": elle contient la première instruction de sous-routine suivant une interruption externe.

- adresse "7": elle contient la première instruction de sous-routine due à une interruption interne résultant du débordement du compteur/temporisateur "T".

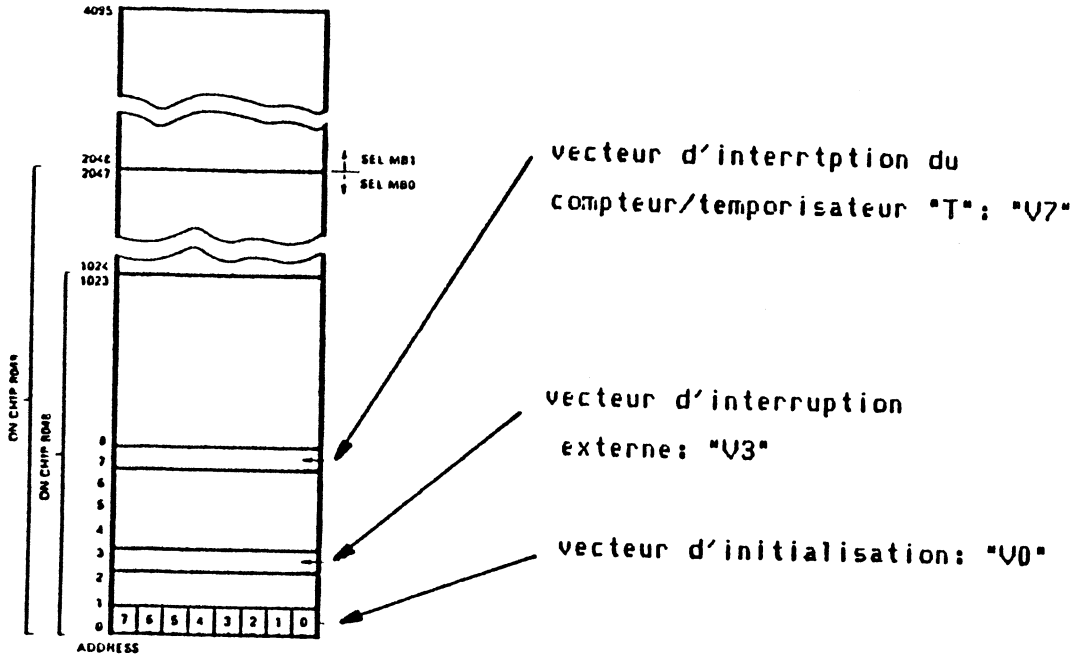


fig:III.4 Configuration interne de la mémoire morte. (INT48)

d) la mémoire vive:

La mémoire vivd résidente est constituée de 64 registres de huit bits distribués en une banque "0", une pile, une banque "1" et des registres utilisateur:

- la banque "0" se compose de huit registres de travail (position

"0" à "7"), directement adressables,

- la pile est constituée de huit paires de registres (position "8" à "23"). Ces mots mémoire sont adressés par l'intermédiaire d'un pointeur de pile contenue dans le mot d'état programme (cf III 1-1i); durant une sous-routine (interruption, "CALL"), les contenus du compteur de programme et du mot d'état programme sont sauvegardés dans une paire de registres désignée par ce pointeur.

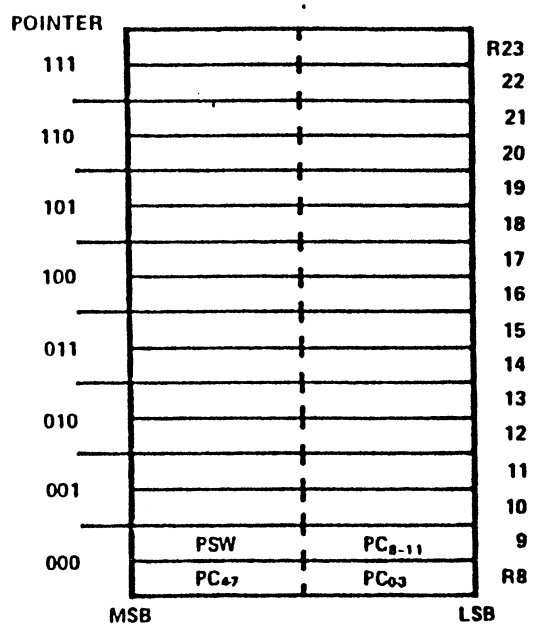


fig:III.5 Représentation de la pile de sauvegarde. (INT48)

- la banque "1" est constituée de huit registres de travail (position "24" à "31"), directement adressables.

La sélection de l'une de ces banques se fait sur une requête du logiciel ("SEL RB0", "SEL RB1").

- les registres utilisateur (position "32" à "64") sont adressables de façon indirecte par l'un des deux registres "R0" ou "R1" situés à l'adresse "0" ou "1" de l'une des banques sélectionnée.

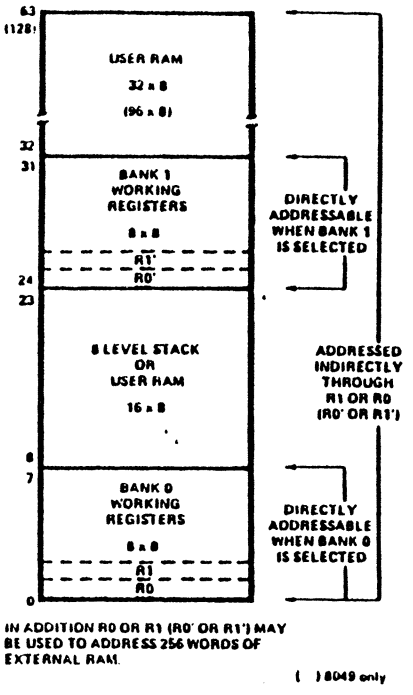


fig:III.6 Configuration de la mémoire vive

e) l'unité de contrôle et de séquencement:

Chargée d'assurer le bon déroulement de l'exécution d'un programme, l'unité de contrôle et de séquencement doit à cet effet:

- générer les signaux d'horloge interne nécessaires au séquencement des instructions (cf III 2-3b),
- valider et analyser les lignes de contrôle externe ("INT*", "RESET*", "SS*"; cf fig: II.6, III.1),
- générer les signaux de synchronisation qui autorisent les transferts d'informations entre le microprocesseur et son environnement. ("PROG", "ALE", "PSEN*", "RD*", "WR*"; cf fig: II.6, III.1).

(cf III 1-4)

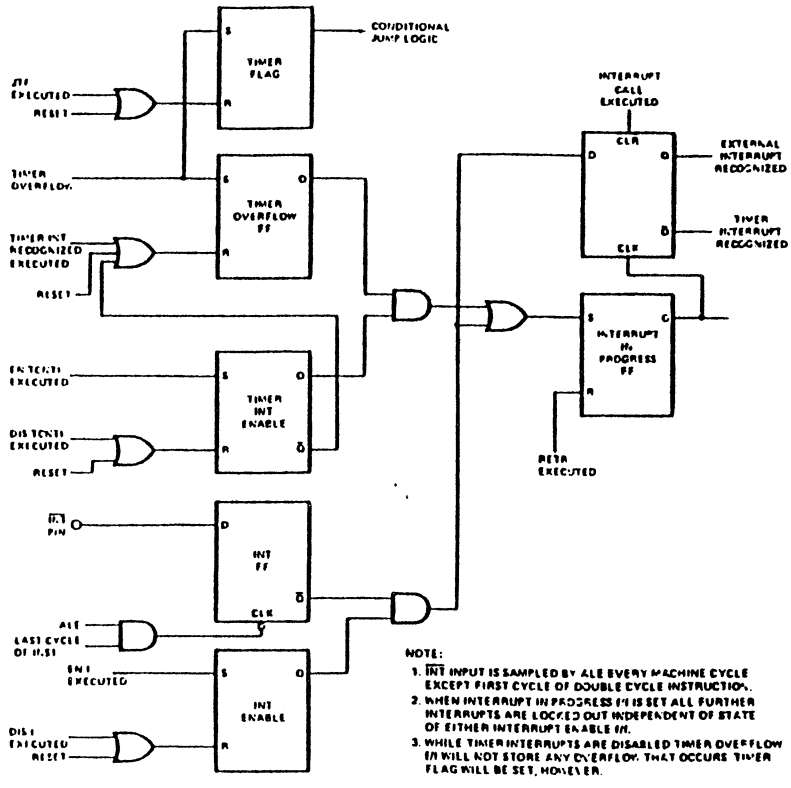


fig:III.7 Logique d'interruption. (INT48)

Ainsi, une séquence d'interruption est initialisée de l'extérieur en appliquant un niveau bas sur la broche "INT*" du 80C48. La ligne d'interruption est échantillonnée à chaque cycle machine durant la phase active de "ALE". Une fois détectée, un branchement s'éffectue au vecteur d'interruption "V3" de la mémoire morte. Pendant l'exécution de l'interruption courante, toute nouvelle demande d'interruption sera ignorée (interruption à un seul niveau). Le retour au programme principal se fait par une instruction "RETR".

Une telle séquence est aussi valable pour une interruption interne au microprocesseur; toutefois, les interruptions générées par l'activation des lignes d'entrées/sortie sont prioritaires.

Une seconde ligne d'interruption externe "T1" peut être validée: le compteur/temporisateur "T" est chargé à sa valeur maximale "FFH" et l'on fait travailler celui-ci en compteur d'évènements externes; un débordement de sa valeur génère un branchement au vecteur d'interruption "V7" de la mémoire morte. (cf II 1-1c)

f) l'unité de branchements conditionnels:

Elle permet la prise en compte des variables d'états ("T0", "T1", "INT", "FLAG0", "FLAG1", "TF", "C", "AC", "bit(AC)") lors des instructions relatives au déroutement d'un programme. (cf III 3-1b2,III 3-1c3)

g) les entrées/sorties:

Le 80C48 possède 27 lignes d'entrées/sorties regroupées en trois ports de huit bits ("B", "P1", "P2") et trois lignes de tests ("T0", "T1", "INT*"). Les ports peuvent être lus, écrits ou multiplexés.

- les ports "1" et "2" sont identiques. Une valeur écrite est mémorisée dans un registre statique ("P1L", "P2L") et ne sera écrasée que par une nouvelle écriture ou une réinitialisation ("RESET") du processeur; la lecture se fait de façon dynamique, la variable devant être présente durant l'instruction de lecture.

Les lignes constituant les ports "1" et "2" sont dites quasi-bidirectionnelles car leur structure est telle que chacune d'entre elles peut être lue ou écrite de façon indépendante sans détruire la valeur initialement enregistrée.

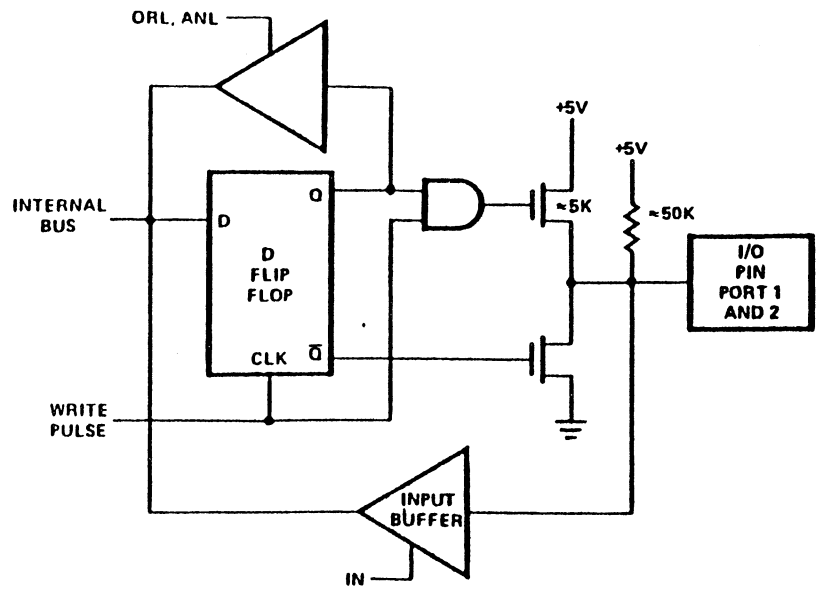


fig:III.8 Structure du port quasi-bidirectionnel. (INT48)

- le port "B" est réellement bidirectionnel; il peut être lu, écrit en association avec des signaux de synchronisation lors d'une configuration "multi-chip", ceci sans toutefois autoriser le multiplexage. Lorsque le mode bidirectionnel n'est pas requis, une valeur écrite sur le port peut être mémorisée de façon statique (registre "BL") tandis que la lecture se fait par échantillonnage.

h) le registre d'instruction:

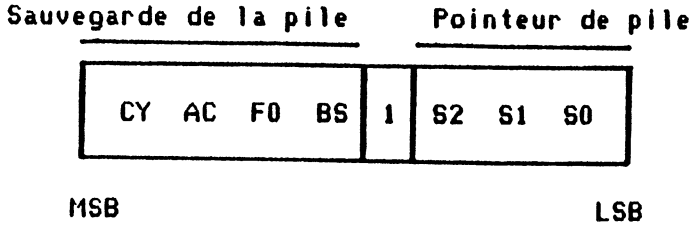
Le registre d'instruction reçoit de la mémoire les codes opérations ("CODOPs") relatifs à l'exécution d'un programme. Leur décodage par l'unité de contrôle et de séquencement permet de déterminer les transferts et les opérations nécessaires au traitement de l'instruction courante.

i) le mot d'état programme:

Le "PSW" ("Program Status Word") est constitué d'un ensemble de

bascules distribuées à travers le microprocesseur; assimilé à un registre de huit bits, il contient les informations suivantes:

- PSW(0,1,2):= pointeur de la pile "S0", "S1", "S2",
- PSW(3):= constante,
- PSW(4):= sélection de la banque des registres de travail
  BS=0; banque "0"
  BS=1; banque "1"
- PSW(5):= flag "0" (F0),
- PSW(6):= retenue auxillaire "AC" résultant d'un calcul sur les quatre bits de poids-faibles de deux opérandes,
- PSW(7):= retenue "C" générée par une opération arithmétique sur deux opérandes de huit bits.



"CY": carry
"AC": auxillary carry
"F0": flag 0
"BS": register bank select

fig:III.9 Configuration du mot d'états programme "PSW"

j) le compteur/temporisateur (T):

Ce registre travaille en compteur d'évènements externes ou en temporisateur. La sélection de l'un de ces modes se fait par des instructions spécifiques "STRT CNT" et "STRT T".

Un débordement du compteur (passage de sa valeur maximale à zéro) rend actif la bascule de requête d'interruption; celle-ci sera alors validée s'il n'y a pas de conflit avec une demande d'interruption externe.

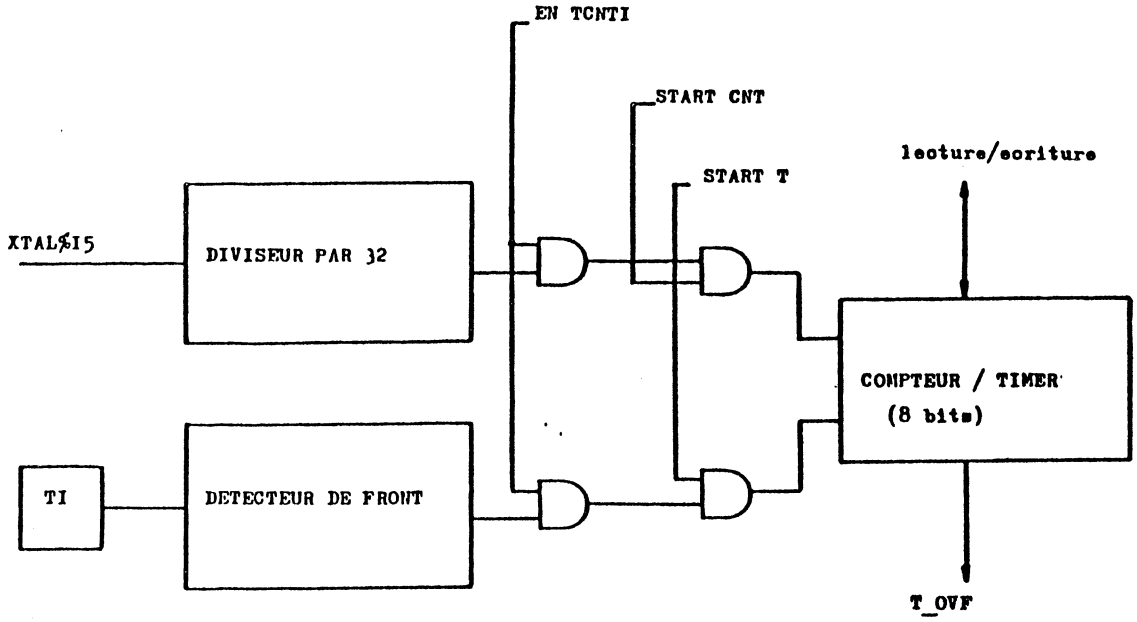


fig:III.10 Configuration du compteur/temporisateur "T"

- mode compteur:

l'entrée du compteur est connectée à la broche "T1"; le passage de "T1" du niveau haut au niveau bas incrémente le registre à l'occurrence de une fois tous les trois cycles machine.

Sa sélection s'éffectue par l'instruction "STRT CNT".

- mode temporisateur:

l'entrée du compteur est connectée à une horloge interne obtenue en divisant par trente deux la fréquence machine (fréquence du cristal divisée par quinze); le compteur est ainsi incrémenté toutes les 80 microsecondes.

Sa sélection s'effectue par l'instruction "STRT T".

1-2 Le jeu d'instructions

Les instructions sont au nombre de 96; plus de la moitié ne nécessitent qu'un seul cycle machine. Les instructions à deux cycles concernent des opérations d'entrées/sorties, de saut ainsi que l'accès à des valeurs

immédiates; elles traitent un ou deux octets.

Le tableaux ci-dessous énumèrent les instructions de base du 80C48; elles sont regroupées par fonctions qui constituent des classes.

fig:III.11 Le jeu d'instructions du 80C48. (INT48)

	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
	JZ 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
	XCHA, @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

fig:III.12 Les codes opérations du 80C48.

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, #d	MOV @R0, B	MOV A, B	XRL A, @R0	IDH	MOV A, @R0
1															
DBUG	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, #d	MOV @R1, P1	MOV B, A	XRL A, @R1	IDNH	MOV A, @R1
2															
OUT B,A	JB0 ADD	IN A,B	JB1 ADD	MOV A,T	JB2 ADD	MOV T,A	JB3 ADD	MOV A,STS	JB4 ADD	MOV A,STS	JB5 ADD	MOV B,P2	JB6 ADD	MOV P2,B	JB7 ADD
3															
ADD A,#d	ADDC A,#d	MOV A,#d	****	ORL A,#d	ANL A,#d	****	****	RET	RETR	MOVP A,@A	JMPP @A	MOV B,P3	XRL A,#d	MOVP 3A,@A	***
4															
JMP P0	CALL P0	JMP P1	CALL P1	JMP P2	CALL P2	JMP P3	CALL P3	JMP P4	CALL P4	JMP P5	CALL P5	JMP P6	CALL P6	JMP P7	CALL P7
5															
EN INT	DIS INT	EN TINT	DIS TINT	STRT CNT	STRT T	STOP TCNT	EN TO	CLR F0	CPL F0	CLR F1	CPL F1	SEL RB0	SEL RB1	SEL MB0	SEL MB1
6															
JNT F	JIF	JNTO	JTO	JNT1	JNF1	JF1	JNI	JNZ	JNZ	JNB	JF0	JZ	JNI BF	JNC	JC
7															
DEC A	INC A	CLR A	CPL A	SWAP	DAA	RRCA	RRA	CLR B	CLR CY	CPL CY	CPL B	MOV A, PSW	MOV A, PSW	RLA	RLC
8															
INS A, P0	INC @R0	XCH A, @R0	OUTL B, A	ORL A, @R0	ANL A, @R0	ADD A, @R0	ADDC A, @R0	ORL P0, #d	ANL P0, #d	MOV @R0, A	MOV @R0, #d	DEC @R0	XRL A, @R0	DJNZ R0	MOV A, @R0
9															
INS A, P1	INC @R1	XCH A, @R1	OUTL P1, A	ORL A, @R1	ANL A, @R1	ADD A, @R1	ADDC A, @R1	ORL P1, #d	ANL P1, #d	MOV @R1, A	MOV @R1, #d	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, #d	ANL P2, #d	MOV @R2, A	MOV @R2, #d	DEC @R2	XRL A, @R2	DJNZ R2	MOV A, @R2
B															
INS A, P3	INC @R3	XCH A, @R3	OUTL P3, A	ORL A, @R3	ANL A, @R3	ADD A, @R3	ADDC A, @R3	ORL P3, #d	ANL P3, #d	MOV @R3, A	MOV @R3, #d	DEC @R3	XRL A, @R3	DJNZ R3,	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, #d	ANLD P4, #d	MOV @R4, A	MOV @R4, #d	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, #d	ANLD P5, #d	MOV @R5, A	MOV @R5, #d	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, #d	ANLD P6, #d	MOV @R6, A	MOV @R6, #d	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, #d	ANLD P7, #d	MOV @R7, A	MOV @R7, #d	DEC @R7	XRL A, @R7	DJNZ R7	MOV A, @R7

8048

1-3 Séquencement des instructions

a) les horloges

Les signaux d'horloges sont générés par l'unité de contrôle et de séquencement à partir d'une référence de fréquence externe "XTAL".

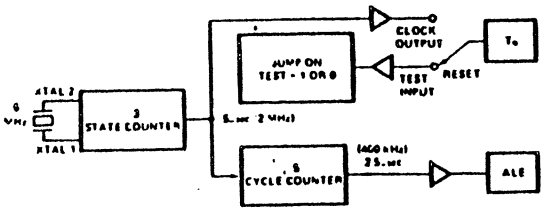


fig:III.13 Génération des signaux d'horloges. (INT48)

On distingue les trois blocs fonctionnels suivants:

- l'oscillateur:

c'est un circuit série résonnant à gain élevé dont la gamme de fréquence est comprise entre 3 et 6 MHz.

- le compteur d'état:

la fréquence de l'oscillateur est divisée par trois dans le compteur d'état pour créer une horloge "CLK" correspondant aux états machine. "CLK" peut être disponible sur la broche "T0", suite à l'exécution d'une instruction "ENTO CLK".

- le compteur de cycles:

la division par cinq du compteur de cycles permet de générer un signal définissant le cycle machine; sa valeur est présente sur la broche "ALE" du 80C48.

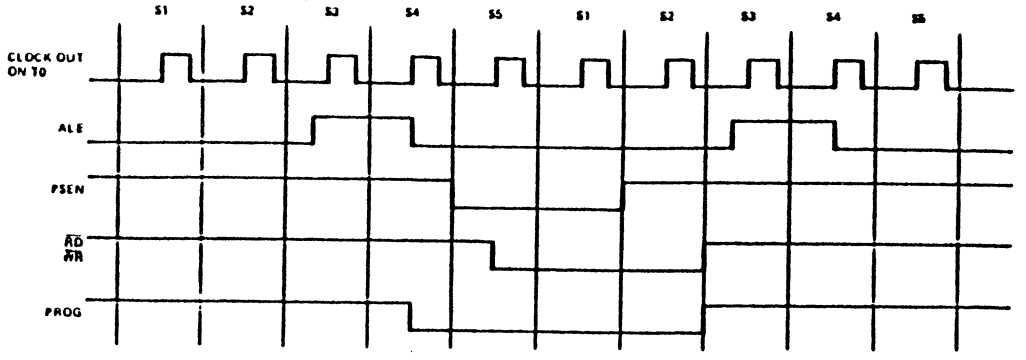


fig:III.14 Chronogramme des horloges de référence. (INT48)

b) le séquencement des instructions:

Les instructions sont de un ou deux cycles machine "ALE"; chaque cycle se décompose en cinq états "S1", "S2", "S3", "S4", "S5" caractérisant un ou des transferts et/ou une ou des opérations.

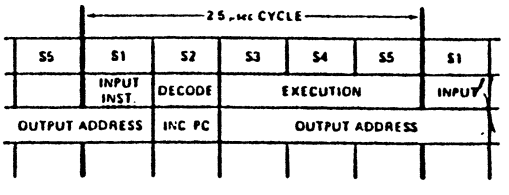


fig:III.15 Cycles d'instruction. (INT48)

Le 80C48 travaille en "pipe-line", le calcul d'adresse étant effectué en parallèle avec l'exécution de l'instruction courante.

Pour les instructions citées en exemples (fig III.16), on remarque certaines opérations systématiques au niveau des états, à savoir:

- la recherche de l'instruction en "S1",
- l'incrémentation du compteur de programme en "S2",
- l'incrémentation du timer en "S4",
- la recherche de valeurs immédiates (adresses, opérandes) en "S6" pour les instructions traitant deux octets,

- l'incrémentation du compteur de programme en "S8" (pour les instructions de deux cycles machine).

fig:III.16 Séquencement des instructions. (INT48)

INSTRUCTION	CYCLE 1					CYCLE 2				
	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
NOP	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	—	—	READ PORT	• —	—	—
OUTL P,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	OUTPUT TO PORT	—	—	• —	—	—
ANL P,#DATA	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	• INCREMENT PROGRAM COUNTER	OUTPUT TO PORT	—
ORL P,#DATA	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	• INCREMENT PROGRAM COUNTER	OUTPUT TO PORT	—
MOVX A,@R	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	—	—	READ PORT	—	—	—
OUTL BUS,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	OUTPUT TO PORT	—	—	—	—	—
ANL BUS,#DATA	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	• INCREMENT PROGRAM COUNTER	OUTPUT TO PORT	—
ORL BUS,#DATA	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	• INCREMENT PROGRAM COUNTER	OUTPUT TO PORT	—
MOVX @R,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	OUTPUT DATA TO RAM	—	—	—	—	—
MOVP A,@A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	—	—	READ DATA	—	—	—
MOVD A,P,	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT SPECIFY ADDRESS	INCREMENT TIMER	—	—	READ P2 LOWER	—	—	—
MOVD P,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT SPECIFY ADDRESS	INCREMENT TIMER	OUTPUT DATA TO P2 LOWER	—	—	—	—	—
ANLD P,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	—	—
ORLD P,A	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	—	—
JCONDITIONAL	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	SAMPLE CONDITION	INCREMENT TIMER	—	FETCH IMMEDIATE DATA	—	• UPDATE PROGRAM COUNTER	—	—
STRT CNT	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	—	START COUNTER					
STOP TCNT	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	—	STOP COUNTER					
EN I	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	ENABLE INTERRUPT	—					
DIS I	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	DISABLE INTERRUPT	—					
ENTO CLK	FETCH INSTRUCTION	• INCREMENT PROGRAM COUNTER	—	ENABLE CLOCK	—					

*VALID INSTRUCTION ADDRESSES ARE OUTPUT AT THIS TIME IF EXTERNAL PROGRAM MEMORY IS BEING ACCESSED.

1-4 Configuration "multi-chip"

Afin d'augmenter sa puissance de traitement, le 80C48 peut se voir allouer:

- de la mémoire morte, jusqu'à 4K mots,
- de la mémoire vive, jusqu'à 320 octets,
- un nombre "illimité" d'entrées/sorties.

L'échange d'informations entre le microprocesseur et son environnement est réalisé à l'aide de signaux de synchronisation représentés par le chronogramme suivant:

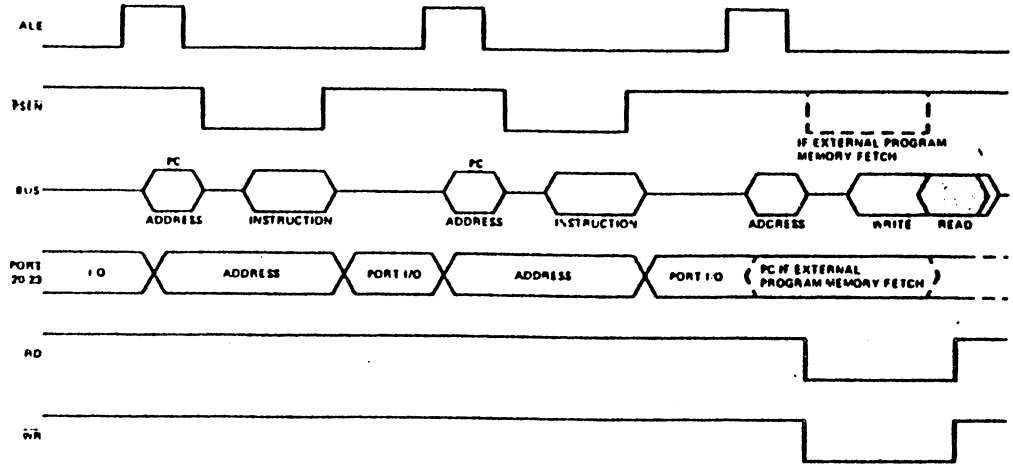


fig:III.17 Chronogramme d'échanges de données dans une configuration "multi-chip". (INT48)

a) extension de la mémoire morte:

Une extension à la mémoire résidente de 1K mots peut être établie par l'adjonction d'une mémoire externe (ex: "8308 Static ROM").

Son accès nécessite la séquence suivante:

1-le contenu du compteur de programme "PC" est émis sur le bus "B" ainsi que sur les bits de poids-faibles du port 2 ("P2L"),

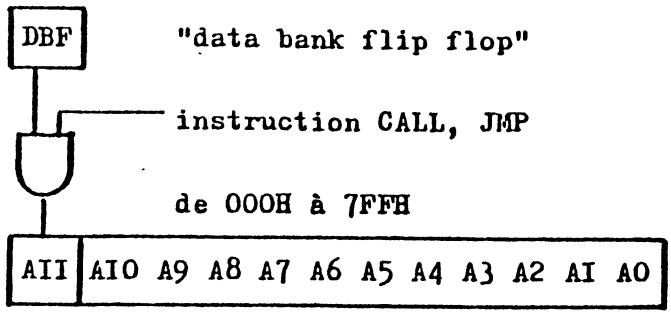


fig:III.18 Le compteur de programme "PC"

2-le signal "ALE" (Address Latch Enable) signale aux périphériques que l'adresse est disponible; l'adresse poids-faibles est alors mémorisée dans un registre externe sur la transition du niveau haut au niveau bas de ALE,

3-le signal "PSEN*" ("Program Store Enable") valide la recherche de l'instruction externe,

4-le bus "B" commute en lecture et autorise le transfert de la valeur au "CPU".

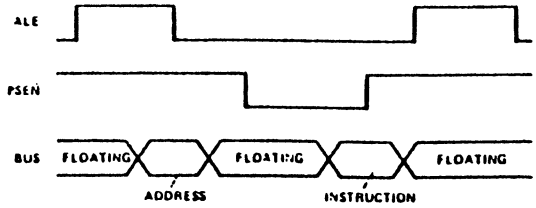


fig:III.19 Recherche d'une instruction en mémoire externe

La sélection de la mémoire morte interne (A11=0B) ou externe (A11=1B) s'effectue selon la valeur du bit le plus significatif du compteur de programme; son contenu n'est pas altéré par une incrémentation de "PC", le chargement est conditionné à l'exécution d'une instruction "CALL" (ou "JMP") par la lecture de la bascule "DBF" ("Data Bank Flip-flop").

Une interruption génére un déroutement du programme aux vecteurs "V3" ou "V7" de la mémoire interne; le onzième bit de "PC" sera forcé à la

valeur zéro durant le traitement de la routine d'interruption.

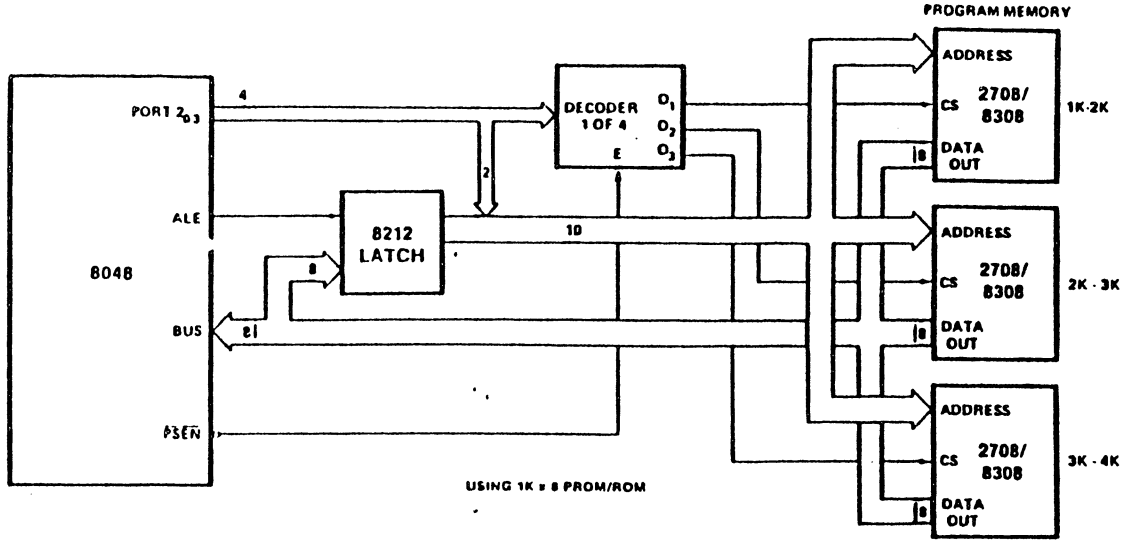


fig:III.20 Configuration étendue de la mémoire morte. (INT48)

b) extension de la mémoire vive:

La mémoire vive de 64 mots peut être étendue à 320 octets dans une configuration "multi-chip".

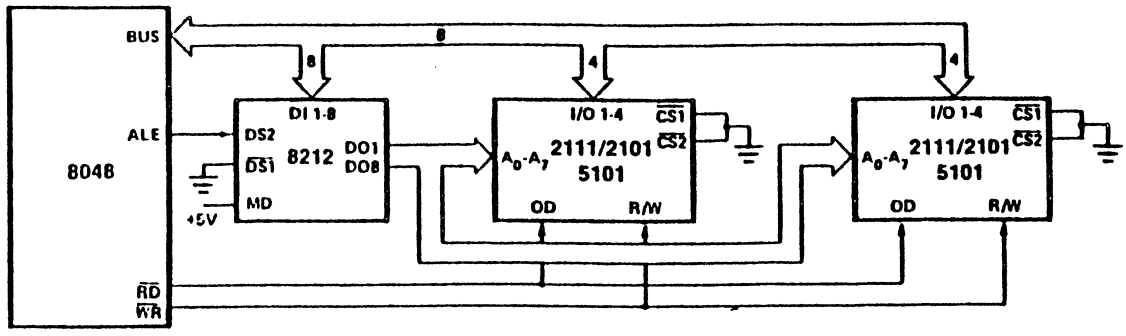


fig:III.21 Configuration étendue de la mémoire vive. (INT48)

Les huits lignes du bus "B" véhiculent les adresses et les données selon la séquence suivante:

1-la valeur contenue dans un des registres "R0" ou "R1" est

écrite sur le bus "B",

2-le signal "ALE" ("Address Latch Enable") indique aux périphériques que l'adresse est disponible; celle-ci sera alors mémorisée sur le front montant de "ALE",

3-un signal de lecture "RD*" ou d'écriture "WR*" sur les broches de sorties du 80C48 indique à la RAM le type d'accès en cours; la donnée est alors disponible pendant sa transition du niveau bas au niveau haut,

4-la donnée est tranférée - lue ou écrite - à travers le bus.

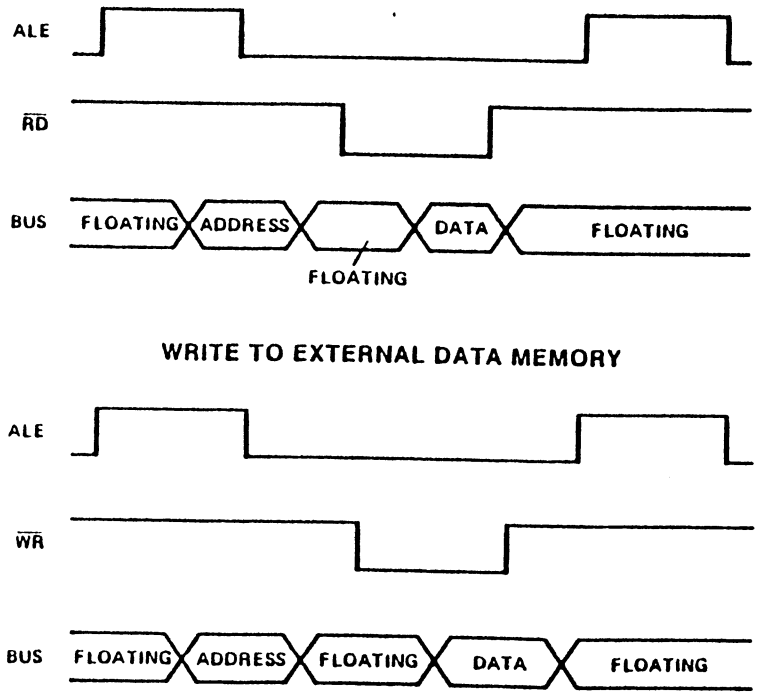


fig:III.22 Lecture/écriture d'une donnée en mémoire externe. (INT48)

c) extension des entrées/sorties:

Elle est facilement réalisable par l'utilisation de circuits spécifiques tels que le "8243"; tout échange avec le 80C48 est réalisé sur les bits de poids-faibles du port 2 ("P2L") grâce au séquencement du signal "PROG".

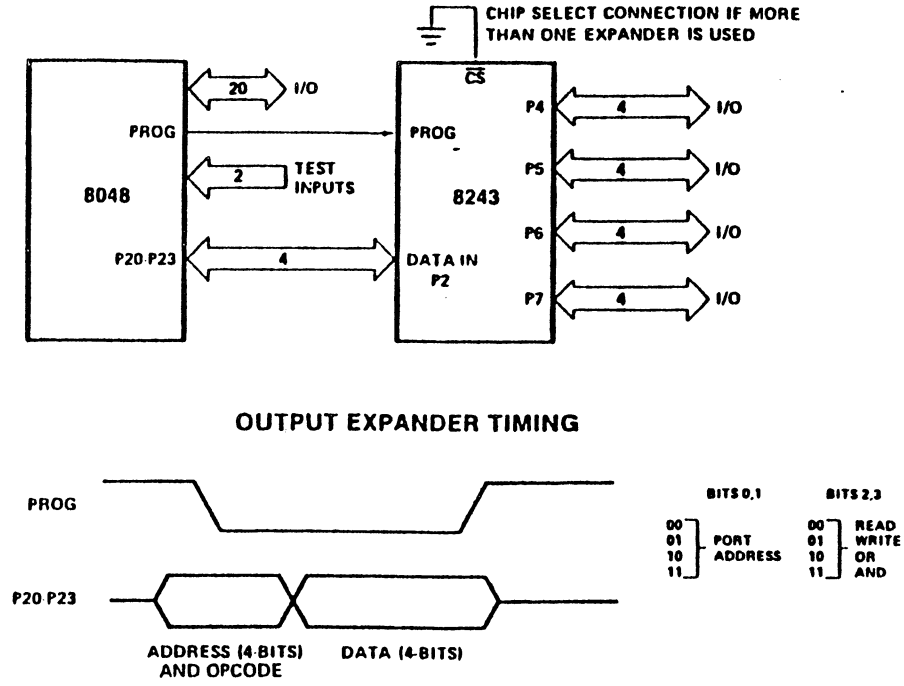


fig:III.23 Configuration étendue des entrées/sorties. (INT48)

Chaque échange consiste en deux mots de quatre bits: le premier contient l'adresse et le code opération ("CODOP") relatifs au port, le second transfère la donnée.

Port P2L:	P23	P22	P21	P20
	I	I	A	A

avec

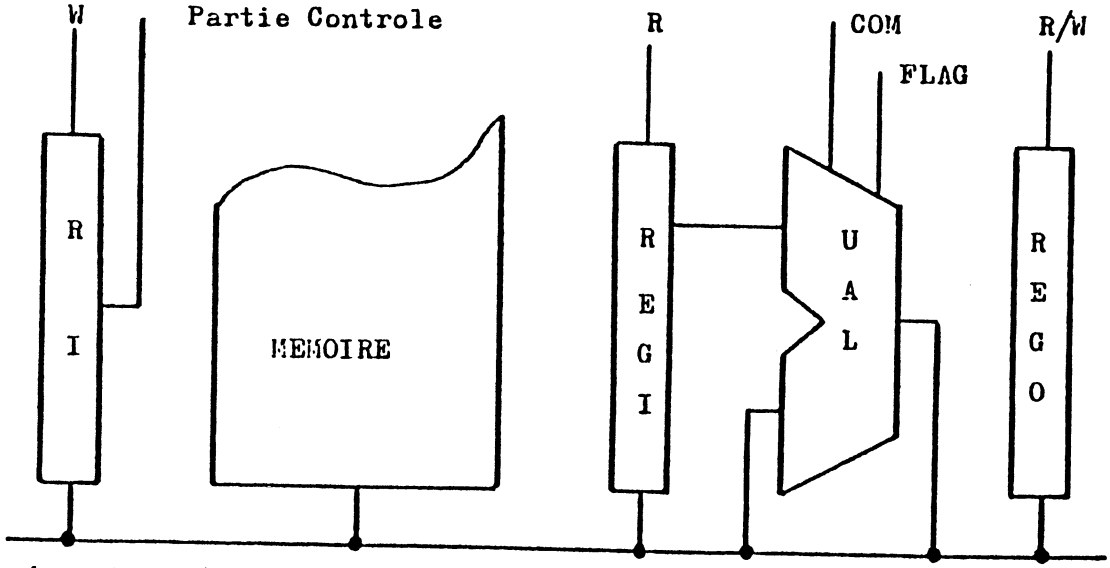
I I:			A A:		
	0 0	Read		0 0	port1
	0 1	Write		0 1	port2
	1 0	OR		1 0	port3
	1 1	AND		1 1	port4

La validation de l'adresse et du code relatifs à l'opération se fait sur le front descendant du signal "PROG", le transfert de la donnée sur le front montant.

III 2 Détermination du chemin des données

2-1 Considérations générales

On appelle chemin de données l'ensemble des éléments permettant de transférer, mémoriser ou traiter les informations (instructions, adresses, opérandes) issues de la mémoire ou de l'environnement du microprocesseur. Les composants - ou primitives matérielles - du chemin des données répondent à l'une des ces trois fonctions; ce sont les bus d'interconnexion pour les transferts d'informations, les registres et mémoires pour leur stockage, les unités fonctionnelles pour leur traitement.



R(read); W(write); RI(registre instruction); REG(registre)
COM(commandes); FLAG(drapeau d'états); UAL(unité arith. & logique)

fig:III.24 Réalisation d'une partie opérative

Tout au long du chemin de données sont distribués des signaux de commande permettant l'exécution de différentes opérations de transfert, de stockage et de traitement d'informations; il s'agit des microcommandes, issues de l'unité de contrôle et de séquencement dont l'étude fera l'objet du paragraphe suivant. (cf III 3)

Les points de contact entre le chemin de données et le générateur de microcommandes se situe à deux niveaux:

- le registre d'instruction, puisque les microcommandes à

distribuer dépendent du code opération,

- l'impact des microcommandes sur les divers composants du chemin de données (lignes de contrôle, lignes d'états).

Notre étude du chemin des données porte sur les trois points suivants:

- les éléments de mémorisation et leurs interconnexions,
- le cheminement des opérandes et des adresses,
- la détermination des unités fonctionnelles.

a) les éléments de mémorisation et leurs interconnexions:

Les éléments mémoire constituent un espace d'adressage. Comme l'on est au niveau de la machine physique, cet espace d'adressage représente l'implantation des données de "LO"; il n'est pas homogène, c'est à dire qu'il est composé d'objets dont le comportement n'est pas identique. On les appelle tous mémoire parce que la fonction physique qu'ils remplissent est une mémorisation. Cette notion recouvre le stockage des constantes (lecture seule), des variables (lecture/écriture) et des mots de commandes et d'états (écriture seule).

Sur le plan fonctionnel, on distingue deux types d'éléments de mémorisation:

- les éléments non adressables, caractérisés par une fonction bien définie qu'ils sont seuls à remplir (par exemple, le registre d'instruction, le mot d'états programme..). Certains de ces registres peuvent rester inconnus du programmeur (registres dit "transparents") alors que d'autres sont implicitement adressés.
- les éléments adressables, utilisés pour conserver certaines informations au cours de calcul; leurs valeurs initiales n'est pas définie.

Sur le plan structurel, on distingue deux organisations possibles: les registres indépendants, directement accessibles après validation d'un signal de lecture ou d'écriture; les registres implantés en mémoire dont un accès nécessite un décodage préalable.

Les interconnexions entre registres ou entre unités fonctionnelles et registres peuvent être directes ou utiliser des lignes de bus.

L'interconnexion par bus présente , par rapport aux liaisons directes, l'inconvénient de ne pas autoriser certains transferts simultanés; en contre partie, l'implantation en est plus facile et plus régulière.

On s'efforcera donc d'utiliser ce type de liaison pour implanter le 80C48.

La réalisation physique de la machine doit supporter l'algorithme d'interprétation qui exécute le jeu d'instructions du 80C48; la détermination du nombre d'éléments et de leur fonctionnalité doit être adaptée à l'algorithme, soit dans le sens de le simplifier, soit de minimiser la surface requise.

b) le cheminement des opérandes et des adresses:

Pour que les variables puissent communiquer entre elles, il faut qu'il existe un chemin de données les reliant: ces chemins sont appelés bus; ils représentent des varaibles de "IO".

La planéarité d'un circuit intégré pose le problème des interconnexions et par là-même de la topologie globale des différents blocs fonctionnels.

Le schéma ci-dessous (fig:III.25) représente les différents éléments "A", "B", "C", "D" d'une partie opérative avec leurs possibilités de communication "alpha", "béta". Ces accès peuvent être uni ou bi-directionnels.

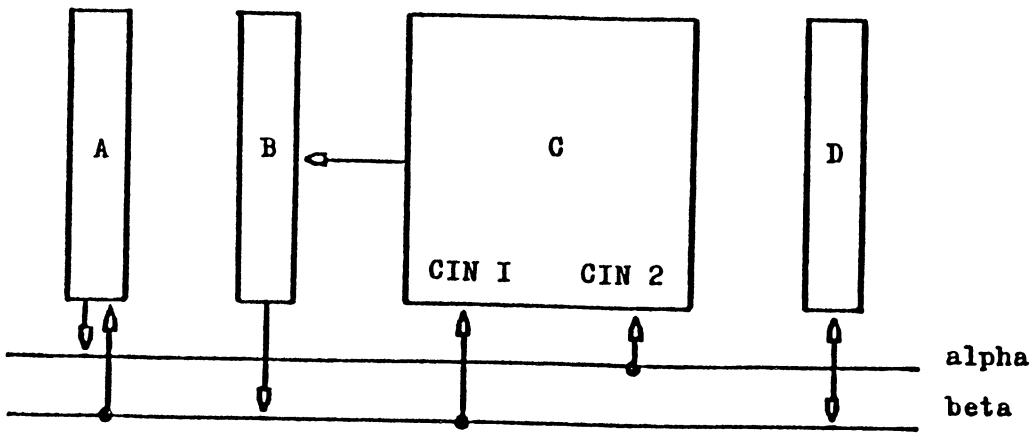


fig:III.25 Exemple de schéma graphique d'une partie opérative

On symbolise par "<--" le chargement d'une variable mémorisée. Il

s'agit là d'une opération d'affectation.

ex: A <-- B

Une connexion correspond au transfert d'une variable dans une variable intermédiaire (et réciproquement); elle est représentée par l'identificateur ":="

ex: alpha:=A;

Chargements et connexions constituent des actions. Le séquencement d'une instruction s'effectue dans un langage de phase "L0"; il progresse pas à pas. A chaque pas peut correspondre un certain nombre d'actions parallèles:

```
ex: phase 0: CIN1 <-- A
             CIN2 <-- D
    phase 1:    B <-- C
    .
    .etc
```

Les chemins de données sont définis à partir des transferts explicités dans l'algorithme d'interprétation. Il est facile de constater que c'est le nombre des opérateurs et la structure des bus qui détermine les possibilités de parallélisme dans le fonctionnement de la machine.

Les spécifications de la partie opérative déterminent les performances, le rôle de la partie contrôle n'étant que de mettre en oeuvre les capacités disponibles.

c) les unités fonctionnelles:

Les unités fonctionnelles constituent des modules qui transforment des données par l'intermédiaire d'un réseau cablé ou grâce à des fonctions combinatoires (ex: décalages binaires, opérations arithmétiques et/ou logiques ...etc).

Ces fonctions peuvent être de une ou plusieurs variables, chaque fonction étant réalisée par un opérateur spécifique.

Un opérateur de deux variables génère une sortie: il a donc trois connexions. L'action s'écrit:

ex: B <-- alpha op bêta

ou "alpha" et "bêta" sont les variables intermédiaires et "B" une variable (fig:III.25). L'opérateur calcule en permanence la fonction "op" lorsque les données sont présentes. Cependant, le chargement de "B" est conditionné au signal d'écriture du registre.

Les fonctions plus complexes du jeu d'instructions du 80C48 devront être réalisées par étapes successives pour se ramener aux fonctions simples utilisant les opérateurs prédéfinis; la suite de ces étapes réalise l'interprétation de l'instruction.

2-2 Ressources matérielles

L'interpréteur "IO" du niveau le plus bas emploie des variables telles que registres, variables d'état de la partie opérative...etc. Les opérations admises dans ce langage permettent de déterminer les opérateurs spécifiques tels que l'UAL, les circuits d'ajustement décimal, de test, de décalage... etc, constituant les ressources matérielles nécessaires à la réalisation de la machine physique.

Les différentes unités fonctionnelles font l'objet de simulations logiques et électriques (cf III 2-5) pour garantir leur fiabilité.

a) Les portes logiques de base

La technologie CMOS permet l'emploi de deux types de transistors, les transistors "N" et "P" symbolisés de la façon suivante:

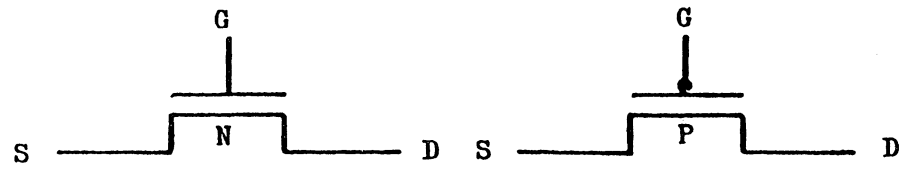


fig:III.26 Représentation symbolique des transistors
a/transistor N, b/transistor P

Ils sont caractérisés par le fait que l'un conduit sur le niveau haut de sa tension de grille (transistor "N") et l'autre sur le niveau bas; on les dit complémentaires.

L'agencement de ces composants permet de réaliser les portes logiques de base nécessaires à la réalisation du 80C48, à savoir:

- l'inverseur:

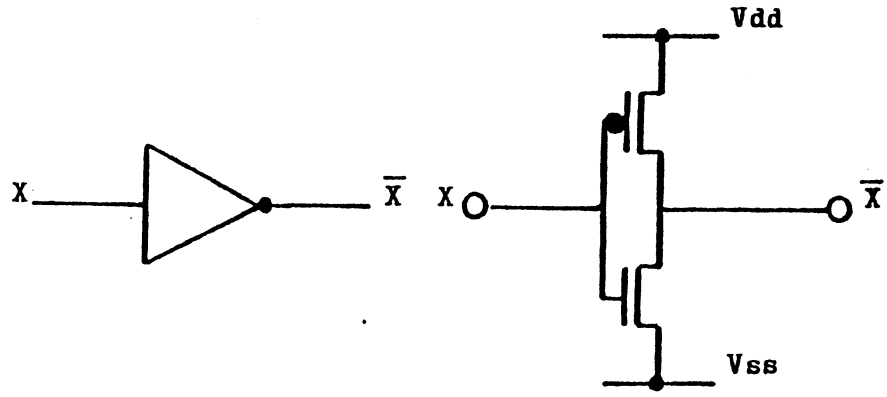


fig:III.27 Représentation a)symbolique et b) électrique d'un inverseur en technologie CMOS

La variable d'entrée "E" commande en parallèle les entrées des transistors "N" et "P".

- la porte de passage:

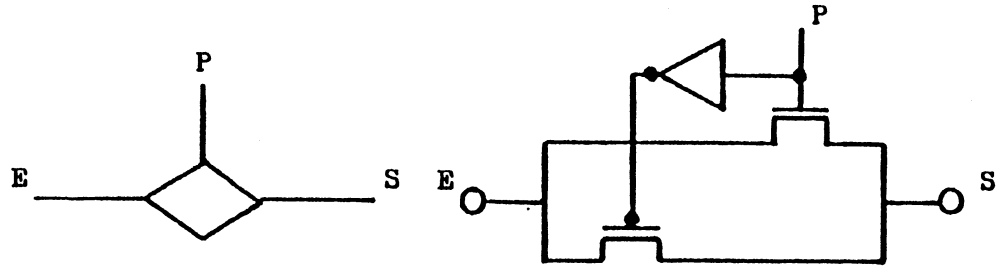


fig:III.28 Représentation a) symbolique et b) électrique d'une porte de passage en technologie CMOS

Un niveau haut de "P" autorise le transfert de l'information de "E" vers "S". Dans le cas contraire (P=0), l'interrupteur est ouvert; la sortie est isolée électriquement de l'entrée.

- l'égalité nonOUX:

Elle se réalise grâce à quatre transistors selon le schéma suivant.

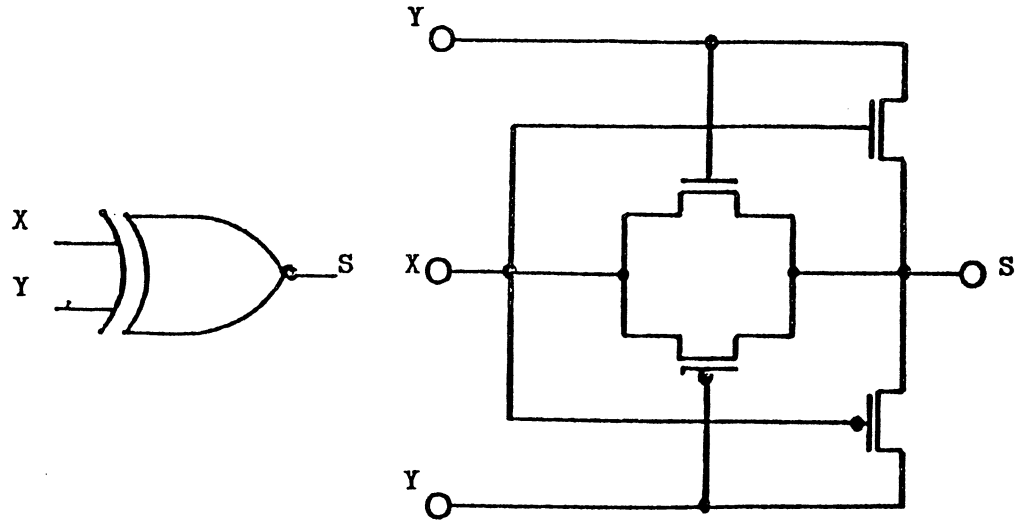


fig: III.29 Représentation a) symbolique et b) électrique de la fonction nonOUX en technologie CMOS

- la fonction NONET et NONOU:

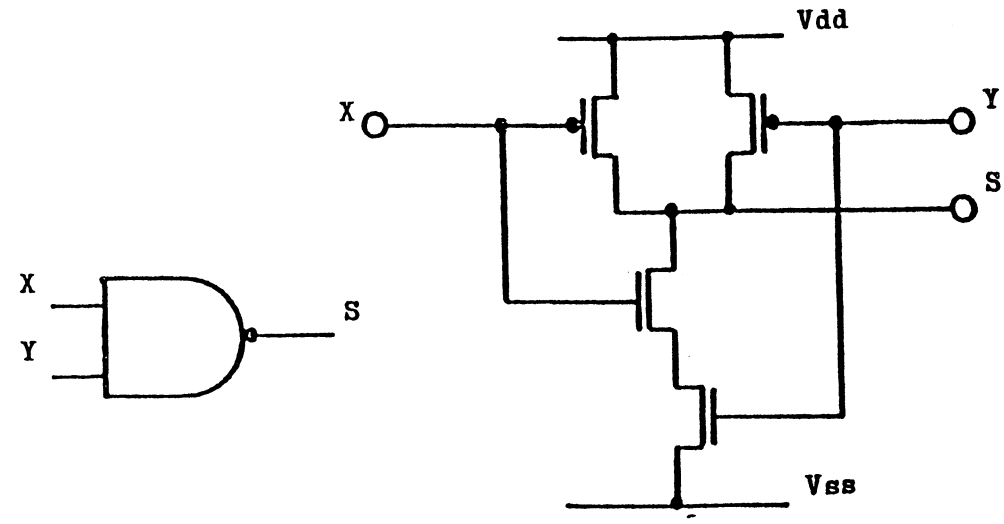


fig:III.30 Représentation a) symbolique et b) électrique de la porte NONET en technologie CMOS

On rappelle que pour une porte CMOS, l'élément de charge est constitué par la fonction duale du réseau signal; le circuit signal est composé de

transistors "N", la charge de transistors "P".

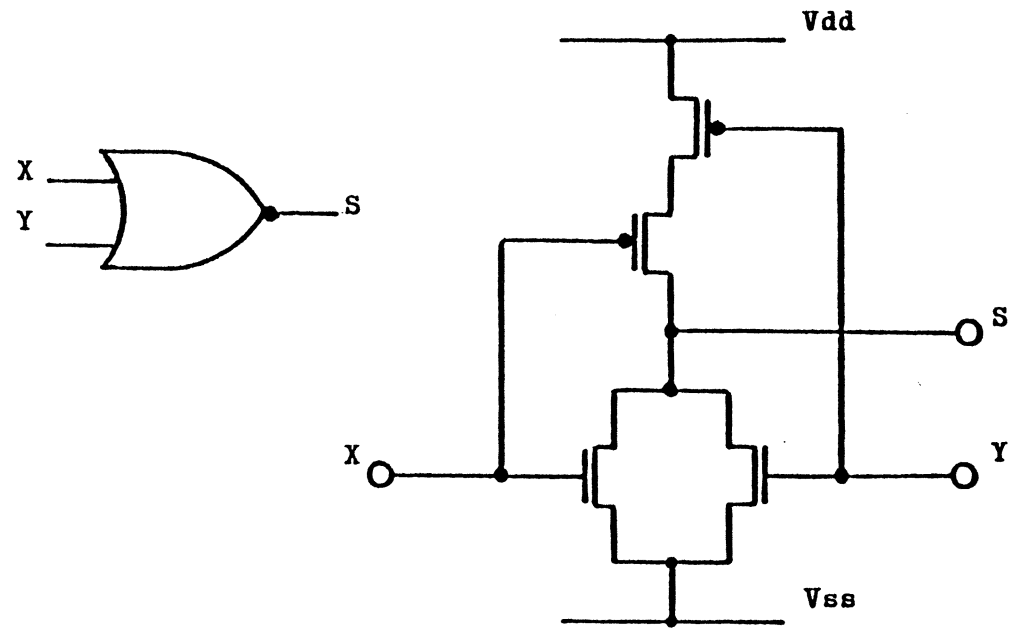


fig:III.31 Représentation a) symbolique et b) électrique de la porte NONOU en technologie CMOS

b) Les points mémoire

Les points mémoire sont constitués de deux inverseurs rebouclés reliés par l'intermédiaire d'interrupteurs à un système de bus unifilaire. Un signal d'écriture "WMEM" autorise le transfert de la valeur portée par le bus dans le point mémoire, "RMEM" le sélectionne en lecture.

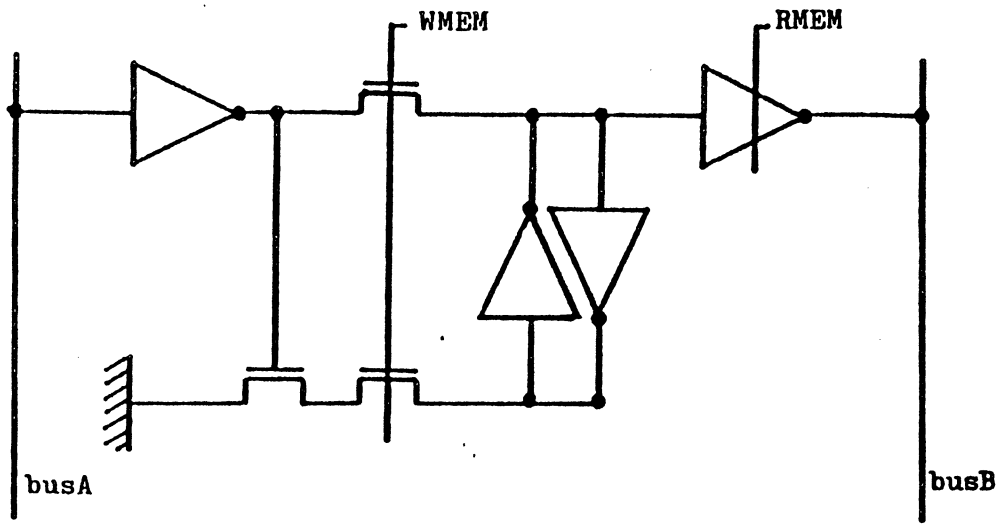


fig:III.32 Représentation d'un point mémoire relié à son système de bus

L'intérêt de l'emploi de bus unifilaires réside dans le fait de disposer de lignes de service disponibles au niveau du "bit slice". (cf III 2-6)

Un autre type de registre permet de forcer la valeur mémorisée par l'utilisation de deux portes logiques "NAND" en remplacement des inverseurs; l'activation de "RTMEM" réinitialise la bascule, "STMEM" force sa valeur au niveau haut.

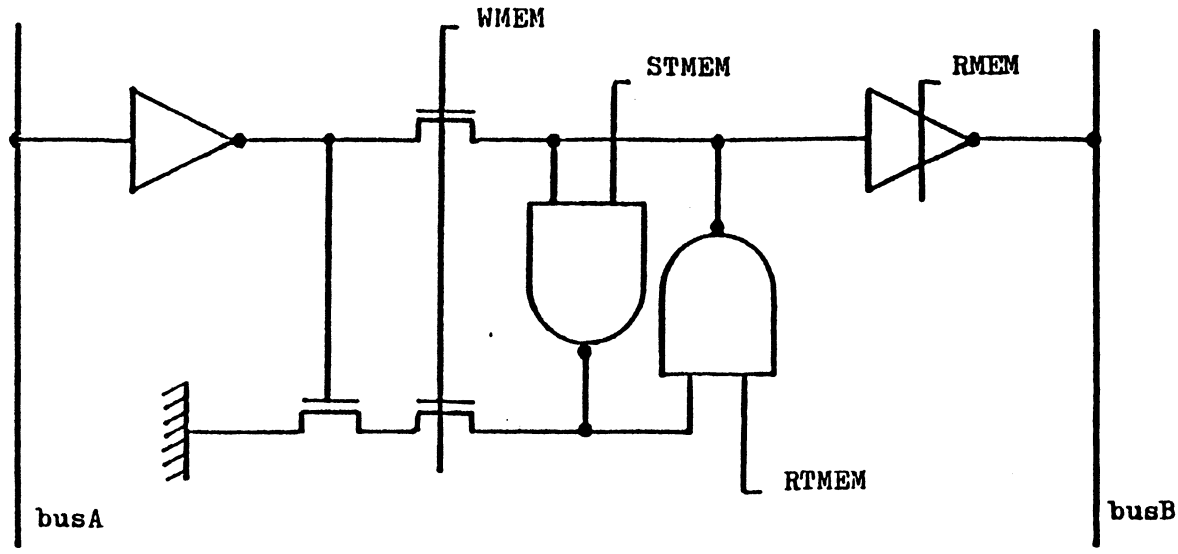


fig:III.33 Configuration d'une bascule avec signal d'initialisation et de remise à zéro

c) Mécanisme de précharge de bus

Il s'agit d'un mécanisme d'anticipation permettant de minimiser la puissance, et donc la surface occupée par les inverseurs de sortie des points mémoire.

Un bus, du fait de sa longueur - il occupe toute la largeur de la partie opérative - représente une charge importante pour un amplificateur d'écriture. On utilise la phase inactive du bus pour anticiper la charge de sa capacité, réduisant ainsi au minimum l'apport de courant fourni par la porte de sortie; on peut considérer la phase de précharge comme une phase d'initialisation du bus à l'état logique "1".

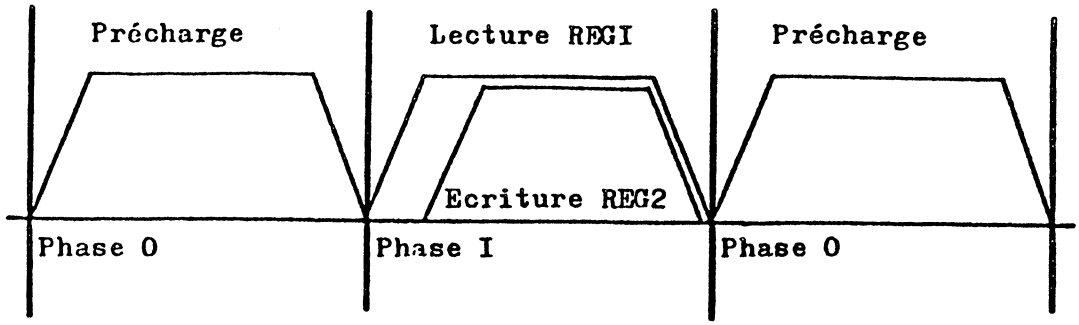


fig:III.34 Diagramme de temps d'une opération de transfert avec précharge de bus

Le transfert d'informations d'un registre "REG1" à un registre "REG2" s'effectue selon la séquence suivante:
(fig III.35)

- phase 0:

précharge du bus; le transistor de précharge "P" charge la capacité "C" que représente le bus,

- phase 1:

le registre "REG1" est sélectionné en lecture (RMEM=1); la valeur du point mémoire lue est écrite sur le bus (décharge de "C" si le niveau logique est zéro); le registre "REG2" est alors sélectionné en écriture (WMEM=1).

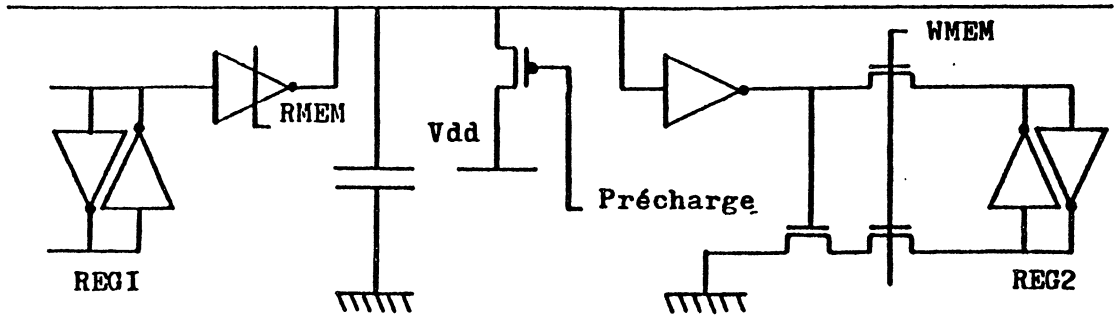


fig:III.35 Opération de transfert d'un registre "REG1" à un registre REG2 avec précharge de bus

On peut remarquer qte le bus est utilisé comme une mémoire dynamique (mémorisation temporaire de l'information avant le transfert dans le point mémoire écrit); cette propriété garantit ainsi la validité de l'échange en cours.

d) L'unité arithmétique et logique

L'unité arithmétique et logique du 80C48 doit permettre d'effectuer l'addition, la disjonction, le ET, le OU, le OUX, l'incrémentation, la décrémentation et la fonction complément.

Cependant, on remarque que l'on réalise facilement la décrémentation d'un opérande par une addition avec "FF" en hexadécimal;

ex: U1 - 1 = U1 + FFH

quant à la fonction complément, elle peut être obtenue par une disjonction.

ex: U1* = U1 oux FFH

On se ramène ainsi à la nécessité de ne disposer que des quatre opérations élémentaires suivantes:

- disjonction: (U1 oux U2),
- OU logique: (U1 ou U2),
- ET logique (U1 . U2),
- addition: (U1 + U2).

Les constantes nécessaires ("00H" et "FFH") sont générées par cablage et sélectionnées par un opérateur spécifique de multiplexage.

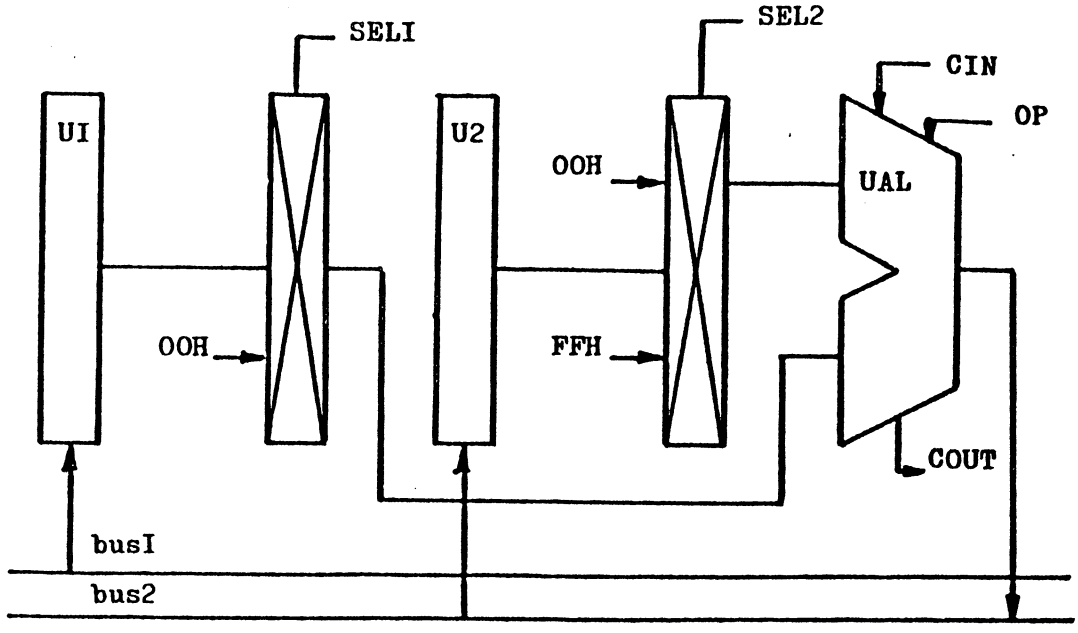


fig:III.36 Configuration de l'unité arithmétique et logique

"U1" et "U2" sont les registres tampon matérialisant une barrière temporelle; l'unité arithmétique et logique calcule en permanence la fonction "op" lorsque les données sont présentes à l'entrée.

L'UAL est réalisée par un réseau de portes logiques réalisant les fonctions combinatoires suivantes:

- disjonction

$(U1 \text{ oux } U2)^* = U1^*.U2^* + U1.U2$

$(U1 + U2)^* + U1.U2$ d'où

$U1 \text{ oux } U2 = (U1.U2 + (U1 + U2)^*)^*$

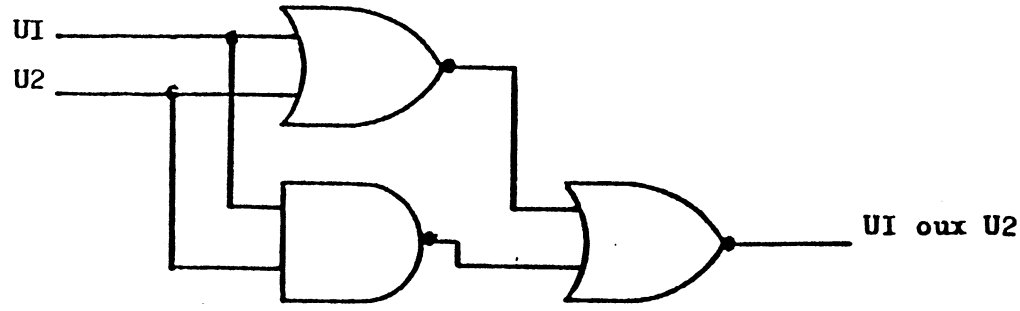


fig:III.37 Réseau de portes matérialisant la fonction OUX

- OU/ET* logique

Partant du schéma précédent, on réalise le "OU" en invalidant le "ET" par la commande "C1";

C1 = 0; R = U1 ou U2

C1 = 1; R = U1 oux U2

de même, l'action d'une commande "C2" sur le "NONOU" permet de réaliser la fonction "NONET".

C1	C2	R
1	1	(U1.U2)*
1	0	U1 oux U2
0	0	U1 ou U2

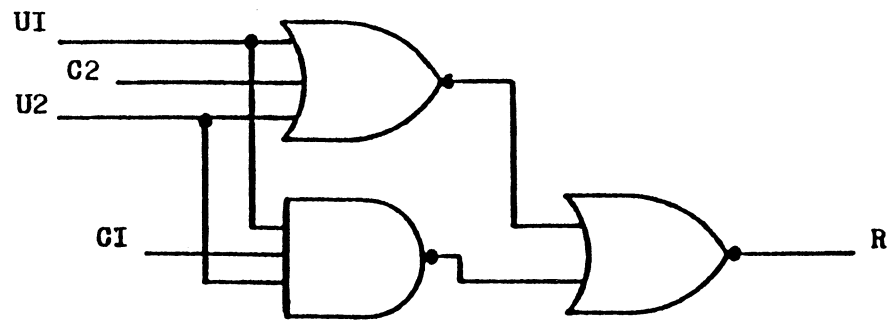


fig:III.38 Réseau de portes matérialisant les fonctions OUX, ET, OU

- addition

L'addition binaire est obtenue par:

Si = (U1 oux U2)I oux Ci (1)

$$Ci+1 = (U1.U2)i + Ci.(U1 \text{ oux } U2) \quad (2)$$

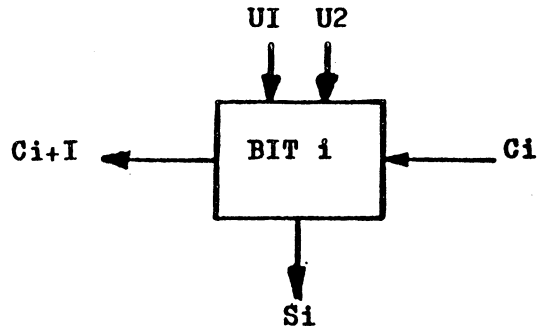


fig:III.39 Cellule d'addition

ou "Ci" et "Ci+1" sont respectivement la retenue entrante et sortante de la cellule d'addition pour le bit "i". Elles seront notées "CIN" et "COUT".

L'équation (1) permet d'écrire:

$$S = R \text{ oux } CIN \quad (3)$$

où R = (U1 oux U2) est obtenu par le réseau de la fig III.38 avec la configuration C1 = 1 et C2 = 0.

L'équation (2) donne:

$$COUT = G + P.CIN \quad (4)$$

avec G = (U1.U2) et P = (U1 oux U2) où "G" désigne le terme de génération de la retenue et "P" sa propagation.

U1	U2	G	P	COUT	COUT*
0	0	0	0	0	1
0	1	0	1	CIN	CIN*
1	0	0	1	CIN	CIN*
1	1	1	0	1	0

fig:III.40 Table de vérité de la propagation "P" et de la génération "G" de la retenue

On remarque que:

- COUT* = CIN lorsque P = 1; en d'autres termes, la retenue est propagée pour des valeurs différentes de "U1" et "U2",
- COUT* = G* lorsque P = 0; la retenue est générée pour des configurations identiques des variables d'entrées.

Pour ce faire, on propage la retenue entrante "CIN" à l'aide du transistor de passage "P", d'où le schéma suivant pour la ligne de retenue.

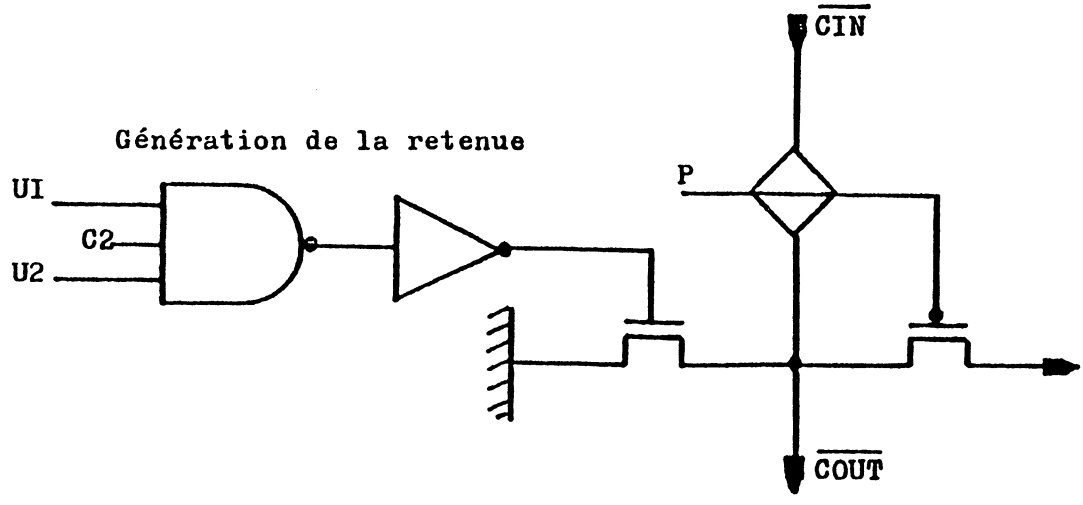


fig:III.41 Ligne de retenue

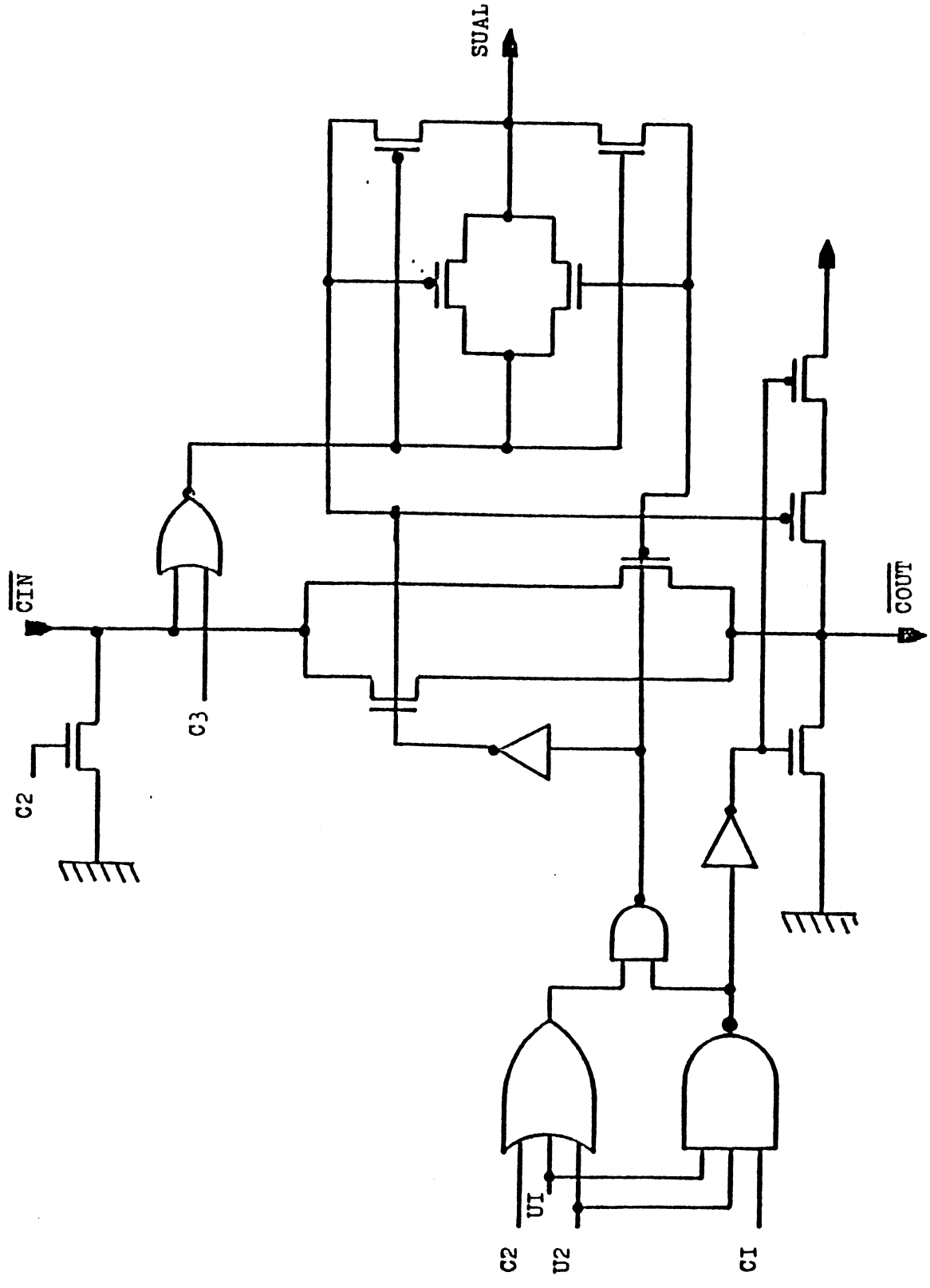
Regroupant les schémas III.38, III.39 et III.41, on peut constituer l'UAL réalisant les fonctions requises.

La ligne de retenue est mise à la masse par la commande "C2" de manière à obtenir le "ET" non complémenté. "C3" inhibe la retenue pendant la disjonction et le "OU".

C1	C2	C3	Opérations
1	0	1	disjonction
1	0	0	addition
1	1	0	et
0	0	1	ou

fig:III.42 Paramétrage des fonctions réalisées par l'unité arithmétique et logique

fig:III.43 Unité arithmétique et logique du 80C48
réalisant les fonctions ET, OU, OUX, Addition



e) l'opérateur de décalage

L'implémentation d'un opérateur de décalage est nécessaire pour réaliser les opérations de rotation (avec ou sans retenue) définies dans le jeu d'instructions du 80C48, soit: "RLA", "RLCA", "RRA, "RRCA".

Implanté derrière l'UAL, le "shifter" autorise les décalages à droite ou à gauche à l'aide des transistors de passage contrôlés par la commande "SL". (SL; "shift left")

SL = 1; décalage à gauche

SL = 0; décalage à droite

La ligne de commande "NS" (NS; "no shift") permet d'invalider cette opération.

NS = 1; pas de décalage

NS = 0; décalage

(fig III.45)

Ces opérations de décalage affectent de la même façon tous les bits de données manipulés; des exceptions concernent les bits externes, c'est à dire les bits de poids le plus fort A(7) et de poids le plus faible A(0). Ecrivons tous d'abord les actions engendrées par les instructions de rotation à droite et à gauche:

- RLA (rotation à gauche sans retenue)

  A(n+1) <-- A(n) avec n:=0..6

  A(0) <-- A(7)

- RLC A (rotation à gauche avec retenue)

  A(n+1) <-- A(n) avec n:=0..6

  A(0) <-- PSW(7)

  PSW(7) <-- A(7)

- RRA (rotation à droite sans retenue)

  A(n) <-- A(n+1) avec n:=0..6

  A(7) <-- A(0)

- RRC A (rotation à droite avec retenue)

A(n) <-- A(n+1) avec n:=0..6

A(7) <-- PSW(7)

PSW(7) <-- A(0)

On remarque, d'après les instructions "RLA" et "RLC A" que le bit "A(0)" peut recevoir la variable "A(7)" et "PSW(7)". De même, au regard des instructions "RRA" et "RRC A", "A(7)" est chargé par le bit "A(0)" ou la retenue "PSW(7)".

La sélection des entrées aux bits zéro et sept se fait par la commande "RC" (RC; "rotate with carry"); cette dernière permet de choisir entre les rotations avec ou sans retenue.

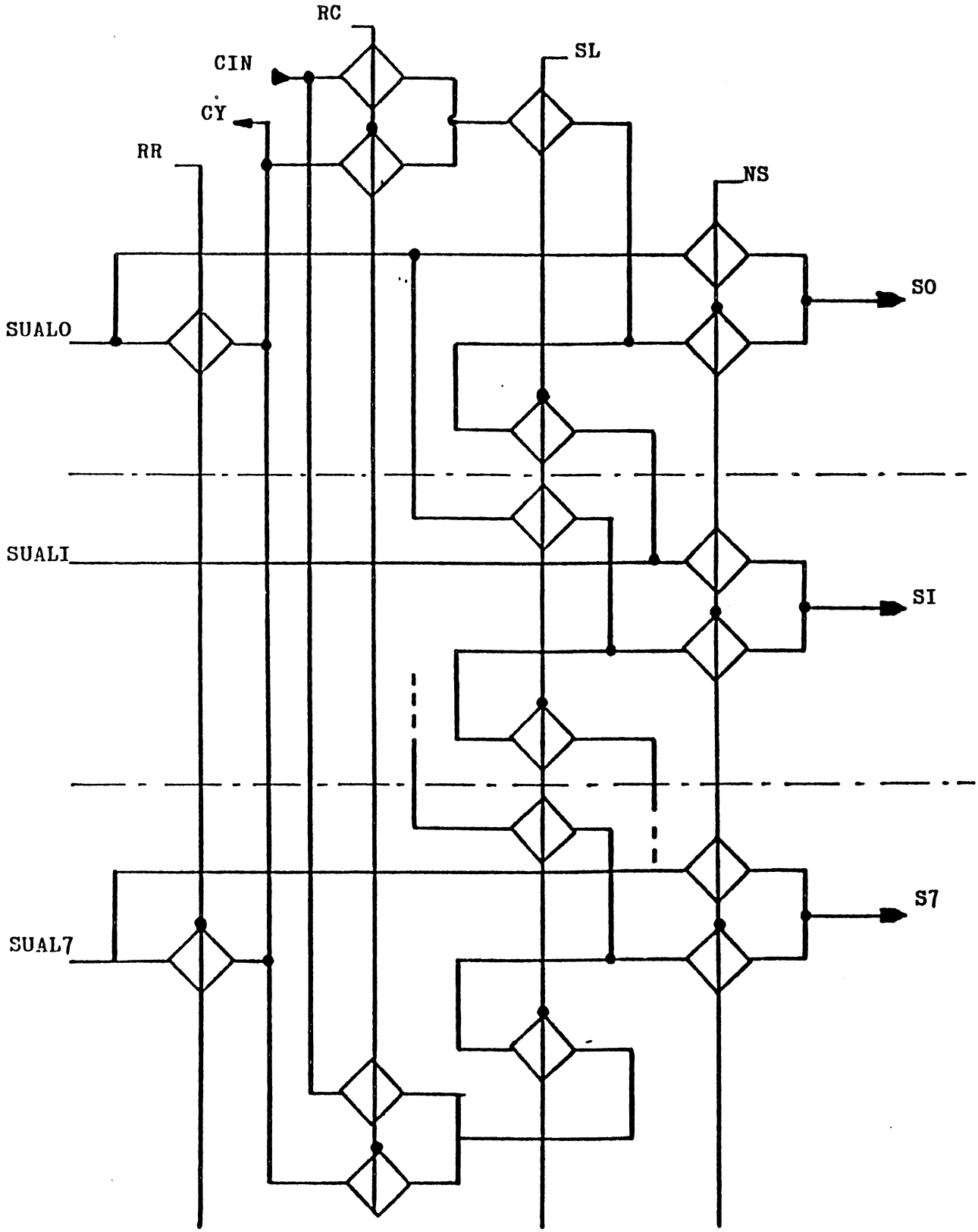
D'autre part, les rotations avec retenue affectent sa valeur. Ainsi le "PSW(7)" reçoit le bit "A(0)" ou le bit "A(7)" de l'accumulateur par l'intermédiaire de la ligne "CY"; la commande "RR" permet d'effectuer la sélection.

commandes: / instructions	RR	RC	SL	NS
RLA	0	0	1	0
RLC A	0	1	1	0
RRA	1	0	0	0
RRC A	1	1	0	0

fig:III.44 Récapitulatif des lignes de commandes avec leurs actions

remarque: l'activation de la ligne "NS" invalide l'opérateur de décalage.

fig:III.45 Représentation logique de l'opérateur de décalage du 80C48



f) opérateur de permutation

Un opérateur spécifique à la permutation poids-faibles/poids-forts est élaboré à la sortie du "shifter" pour permettre l'exécution de l'instruction "SWAP A":

A(7:4) (-- A(3:0)

A(3:0) (-- A(7:4)

Cette opération est effectuée en une seule phase machine. La permutation est réalisée, suite à l'activation de la ligne de commande "SW", par le réseau de portes suivants:

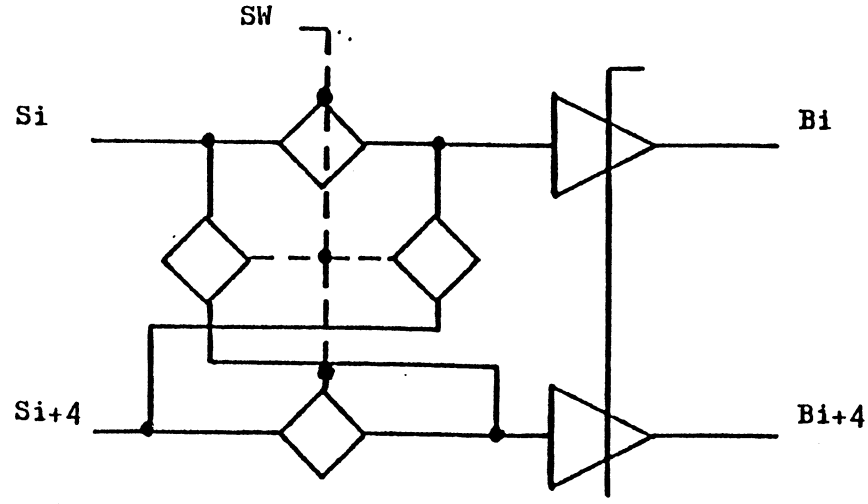


fig:III.46 Représentation de l'opérateur de permutation
SW = 1/ permutation; SW = 0/ invalidation

remarque: une permutation poids-faibles/poids forts peut être réalisée de façon séquentielle par une suite de décalage; cette solution algorithmique n'a pas été retenue car elle n'est pas compatible avec le séquencement des instructions.
(cf III 2-3b)

g) opérateurs de test

Divers opérateurs de test doivent permettre de rendre compte à l'unité de contrôle et de séquencement de l'état de certaines variables du chemin de données.
On dénombre:

- le test d'une valeur nulle ("TZ"),
- le test de la valeur d'un bit ("TB"),

- la détection des codes invalides ("TDAH", "TDAL").

1) Le test d'une valeur nulle est nécessaire lors de l'exécution de certaines instructions telle que les branchements "JZ", "JNZ", "DJNZ"; il est réalisé à l'aide d'un NOR distribué, selon le schéma suivant:

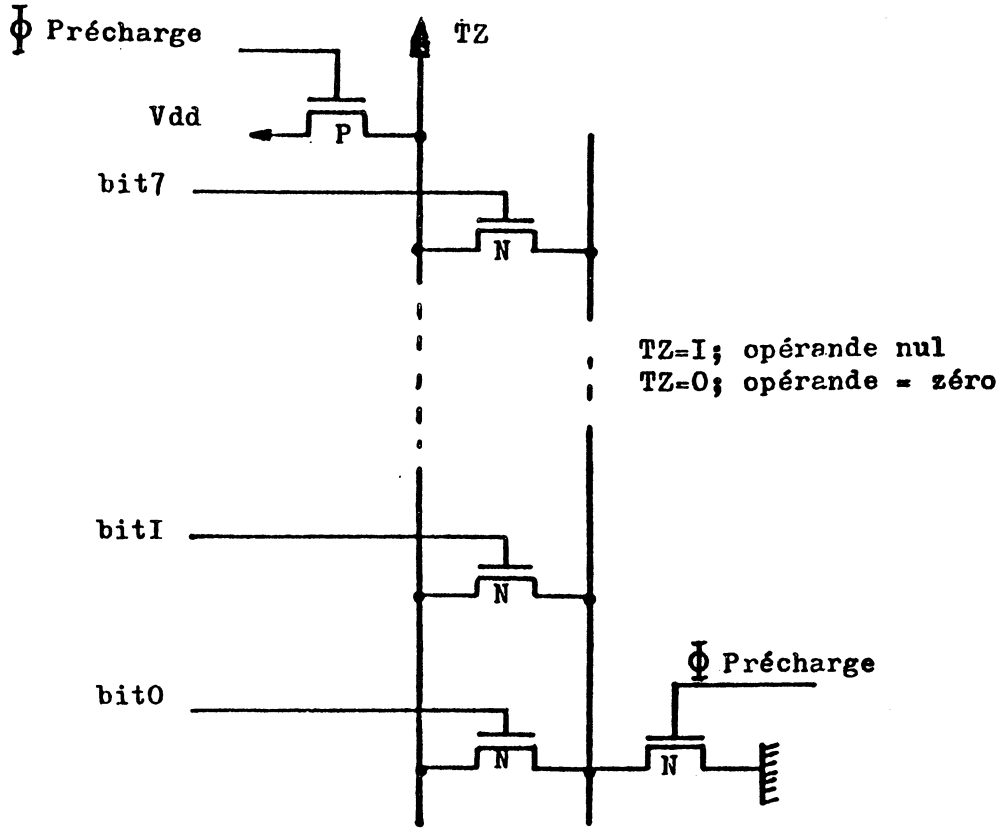


fig:III.47 Réalisation d'un opérateur de test d'opérande nul

L'état de la ligne de contrôle passe à l'état haut lors de la détection d'une variable nulle.

2) Le test de l'état d'un bit conditionne le déroutement du programme courant durant l'exécution d'une instruction de branchement.

La sélection du drapeau testé se fait, grâce à un réseau de transistors de passage, suite au décodage des trois bits de poids-forts du code opération de l'instruction courante RI(7:5).

(n)	RI(7:5)	Instructions mises en oeuvre
7	111	JC; JNC
1	001	JT0; JNT0
2	010	JT1; JNT1
5	101	JF0
3	011	JF1 (*)
0	000	JTF
4	100	JNI

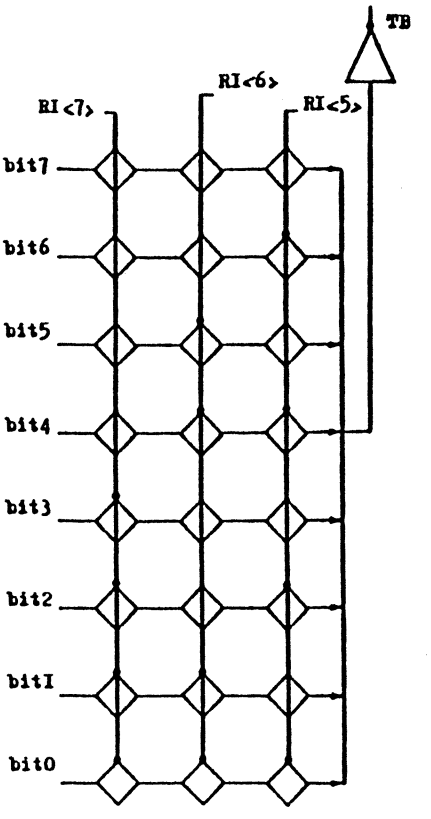
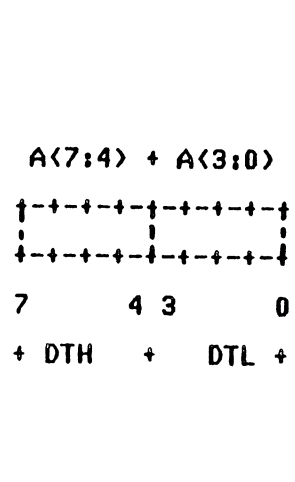


fig:III.48 Réalisation d'un opérateur de test de l'état d'un bit
a) table de vérité; b) représentation logique

(*) on doit effectuer la permutation poids-faibles/poids-forts du registre considéré avant de tester la valeur du drapeau.

3) La détection des codes invalides

L'instruction "DAA" effectue une opération de conversion binaire/décimal sur des opérandes de huits bits:



Nb binaire	DTi
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9

fig:III.49 Nombres décimaux
a/formats; b/ codes représentatifs

La valeur représentée par chaque digit "DTi" étant comprise entre zéro et neuf, un opérateur spécifique à la détection des codes invalides permet d'effectuer la correction selon l'algorithme ci-contre.

```
i;   si TDAL = 1 allera i+1;
                 sinon TDL:= TDL + 6D;
                 i:=i+1;
i+1; si TDAH = 1 fin;
                 sinon TDH:= TDH + 6D;
                 fin;
```

TDAH = 1; 0 <= DTH <= 9

TDAL = 1; 0 <= DTL <= 9

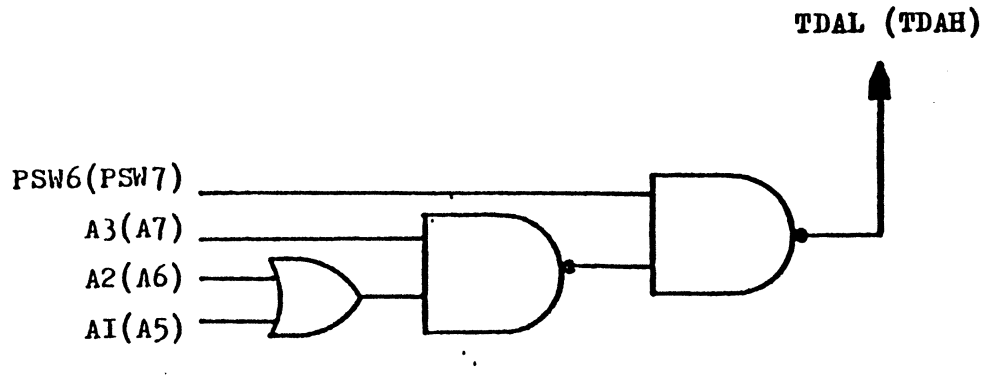


fig:III.50 Réalisation d'un opérateur de détection des codes invalides

h) Configuration du registre d'instruction

Elément constituant de la partie opérative, le registre instruction "RI" contient le code opération de l'instruction courante.

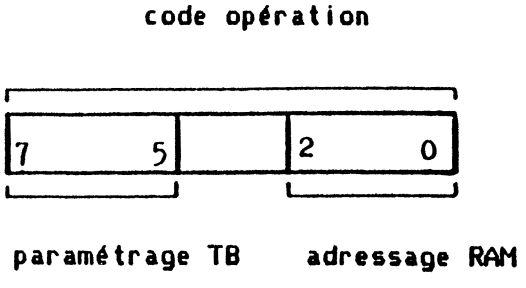


fig:III.51 Configuration du registre instruction

Lors d'un accès à la mémoire vive, l'adresse d'un mot désiré est contenue dans les bits RI<2:0>:

- RI<2:0> permet d'adresser un des huit registres dans la banque mémoire courante,
- RI<0> adresse le registre "R0" ou "R1" de la banque mémoire courante. (cf III 1-1d)

Le paramétrage de l'opérateur de test binaire "TB" s'effectue sur les bits RI(7:5) du code de l'instruction courante.

i) Configuration de la mémoire morte

Elle est subdivisée en quatre pages de deux cent cinquante six mots, adressés au moyen du compteur de programme "PC".
(cf III 1-1c)

- PCH(3) sélectionne l'accès à la mémoire interne (PCH(3) = 0) ou externe (PCH(3) = 1); sa valeur n'est pas modifiée par une incrémentation du compteur de programme mais lors de l'exécution des instructions "CALL" ou "JMP" par la lecture de la bascule "DBF" (DBF; "data bank flip-flop"),

- PCH(2:0) sélectionne la page mémoire considérée

tandis que,

- PCL(7:0) adresse l'octet désiré dans la page courante.

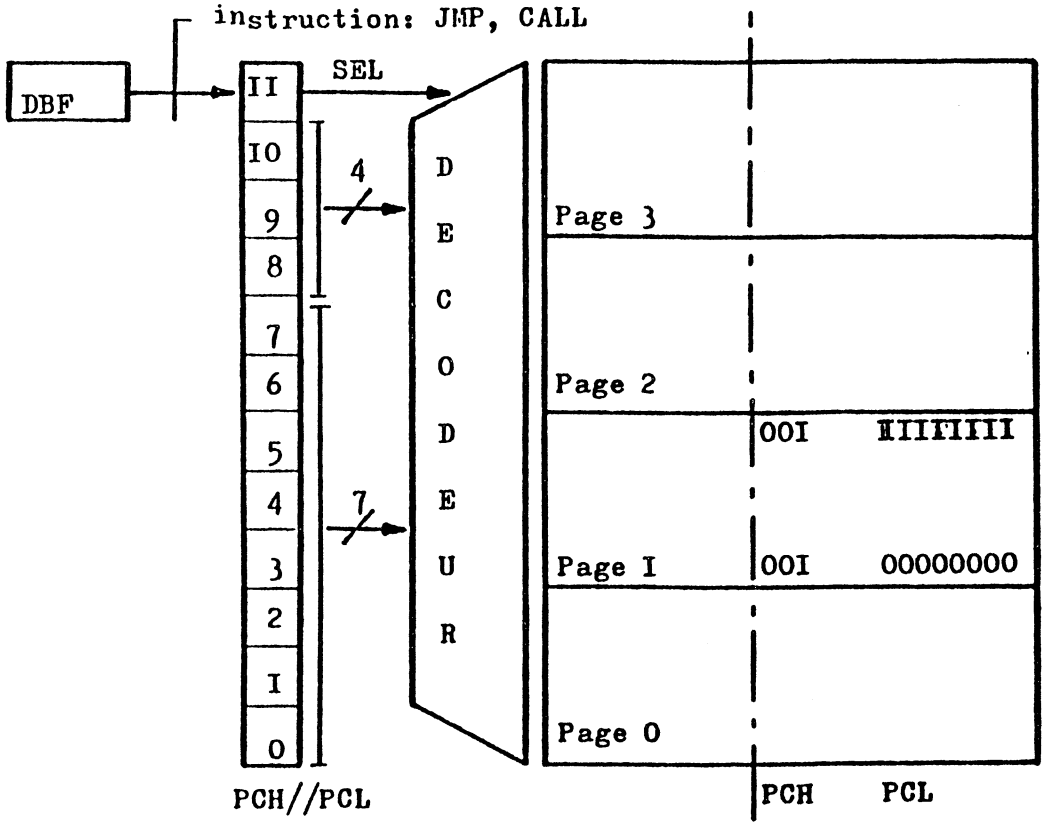


fig:III.52 Adressage de la mémoire morte résidente du 80C48

j) Génération des constantes

L'algorithme d'interprétation élaboré fait apparaître le besoin de disposer d'un certain nombre de constantes nécessaires à l'implémentation et au transcodage de certaines instructions; il s'agit d'opérandes de huit bits.

Au nombre de sept, elles permettent:

- d'exécuter l'instruction "DAA";

constantes mises en oeuvre: 06H, 60H

- d'adresser la pile,

contenue dans la RAM (cf III 1-1d, III 2-2k)
constantes mises en oeuvre: 04H

- d'adresser la RAM (cf III 2-2k)

constantes mises en oeuvre: 00H, 18H

- d'effectuer les opérations d'entrées/sorties

en constituant le mot de quatre bits qui définit le code opération et l'adresse du port désiré. (cf III 1-4c)
constantes mises en oeuvre: 04H, 08H, 00H, 0CH.

exemple: soit à réaliser l'instruction "MOVD A,Pi" dont le code opération est 0000 11ii (avec Pi:= 4..7; n° du port).

L'opération "CODOP + CTE" permet de transcoder les bits de poids-faibles et générer ainsi le mot relatif à l'adresse.

```
MOVD A,Pi       CODOP:= 0000 11iiB
                CTE  := 0000 0100B  (04H)

          CODOP + CTE:= 0001 00iiB
                P2L  :=      00iiB
```

avec P2L(1:0):= ii; adresse du port
P2L(3:2):= 00; code opération de l'instruction

	CODOP	P2L	CTE
MOVD A,Pi	0000 11ii	00ii	04H
MOVD Pi,A	0011 11ii	01ii	08H
ANLD Pi,A	1001 11ii	11ii	00H
ORLD Pi,A	1000 11ii	10ii	0CH

fig:III.53 Constantes mises en oeuvre lors des instructions d'entrées/sorties

La génération des constantes est réalisée à l'aide d'un réseau de portes logiques dont le codage s'effectue sur trois bits "A", "B", "C"

CTE<n>	A	B	C	CTE
0	0	0	0	00H
1	0	0	1	60H
2	0	1	0	04H
3	0	1	1	06H
4	1	0	0	08H
5	1	0	1	18H
6	1	1	0	00H

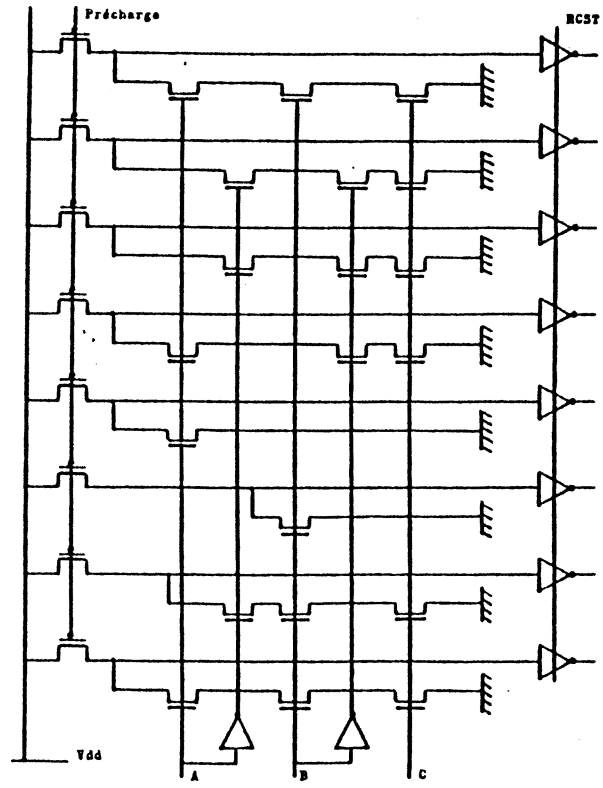


fig:III.54 Génération des constantes
a/adresse; b/réalisation

K) Configuration de la mémoire vive

De soixante quatre octets, la mémoire vive résidente est adressée par l'intermédiaire du registre "W" de huit bits; "W<5:3>" sélectionne la page tandis que les bits "W<2:0>" adressent le mot dans la page considérée.

Une écriture sélective des poids faibles "W<2:0>" et des poids forts "W<7:3>" est autorisée grâce à la duplication des commandes d'écriture, "WWL" et "WWH" respectivement; la lecture se fait quant à elle de façon globale.

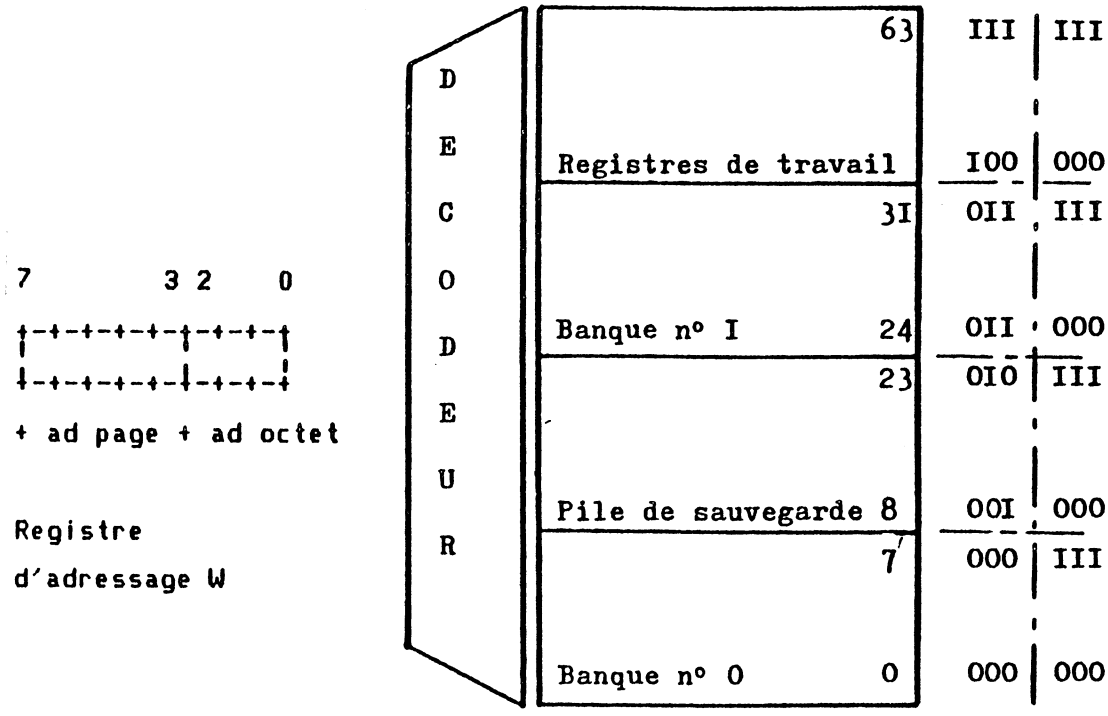


fig:III.55 Adressage de la mémoire vive résidente du 80C48

- l'accès aux banques "0" et "1";

la sélection de l'une des banques est réalisée par logiciel à l'aide des instructions spécifiques "SEL MB0" et "SEL MB1"; on génère alors l'adresse de la banque désirée à partir des constantes "00H" ou "18H", la sélection du mot se faisant après décodage des bits "RI(2:0)" du registre instruction.

```
si MB = 1; * banque 1 *
        W(7:3):= CTE(5)(7:3); * 011B; adresse banque 1 *
        W(2:0):= RI(2:0); * rrrB; adresse de l'octet *

si MB = 0; * banque 0 *
        W(7:3):= CTE(0)(7:3); * 000B; adresse banque 0 *
        W(2:0):= RI(2:0); * rrrB; adresse de l'octet *
```

- l'accès à la pile:

la gestion de la pile est assurée par le compteur de trois bits contenu dans le mot d'états programme "PSW(2:0)"; huit couples d'octets représentant huit imbrications de sous-routines sont ainsi disponibles à l'utilisateur.

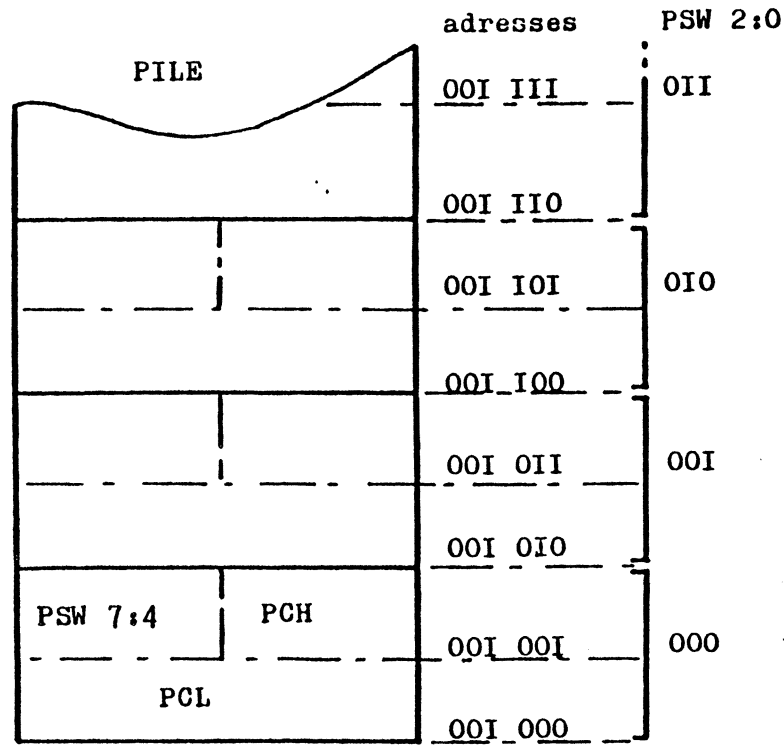


fig:III.56 Adressage de la pile du 80C48

Un transcodage de la valeur de pointeur permet de lui faire correspondre une adresse physique.

exemple:
soit à adresser le troisième doublé référencé PSW(2:0):=010B.

```
            PSW(2:0):=       010B;
              CTE(2):= 0000 0100B; * 04H *
X= PSW(2:0) + CTE(2):= 0000 0110B;
               SL(X):= 0000 1100B; * décalage à gauche *
              W(7:0):= SL(X);
```

W(5:0)= 001 100B constitue l'adresse du premier octet; une

incrémentation unitaire adresse le mot suivant.

- l'accès à la mémoire usuelle:

son adressage est indirect. L'adresse de l'octet désiré est entièrement contenue dans l'un des registres "R0" ou "R1" de la banque courante et son accès s'effectue selon la séquence suivante:

```
i; si MB = 1; * banque 1 *
        W(7:3):= CTE(5)(7:3); * 011B; adresse banque 1 *
        W(2:0):= RI(2:0); * 00rB; adresse octet R0/R1 *
   si MB = 0; * banque 0 *
        W(7:3):= CTE(0)(7:3); * 000B; adresse banque 0 *
        W(2:0):= RI(2:0); * 00rB; adresse octet R0/R1 *

i+1; W(7:3):= RAM(W); * adresse indirecte poids forts *
     W(2:0):= RAM(W); * adresse indirecte poids faibles *

i+2;       X:= RAM(W); * lecture opérande *
```

1) les ports d'entrées/sortie

- les ports "P1" et "P2" ont la même fonctionnalité (cf III 1-1g). Leur structure est telle qu'ils autorisent le multiplexage des entrées/sortie.

Un point mémoire réalisé par un inverseur et un "nand" rebouclés autorise la mémorisation des sorties (fig III.56a), suite à l'activation de la commande "WDPp"; cette valeur demeure inchangée jusqu'à l'exécution d'une instruction "OUTL Pp,A".

On sait que les ports 1 et 2 sont utilisés comme registre de travail pendant l'exécution des instructions "ANL Pp,A" (ET logique) et "ORL Pp,A" (OU logique); leur lecture est réalisée par la commande "DPPp". De même, le "RESET" permet leur initialisation à la valeur logique "1" par la conduction du transistor de charge "P".

On ne peut réaliser une lecture des broches d'entrées qu'après avoir initialisé le port au niveau haut par l'exécution de l'instruction "OUTL Pp,A"; le choix des paramètres permet à l'utilisateur de multiplexer

entrées et sorties sur le même port. La valeur lue n'est pas mémorisée et ne sera validée qu'après activation de la commande "INPp" lors de l'exécution de l'instruction "IN A,Pp".

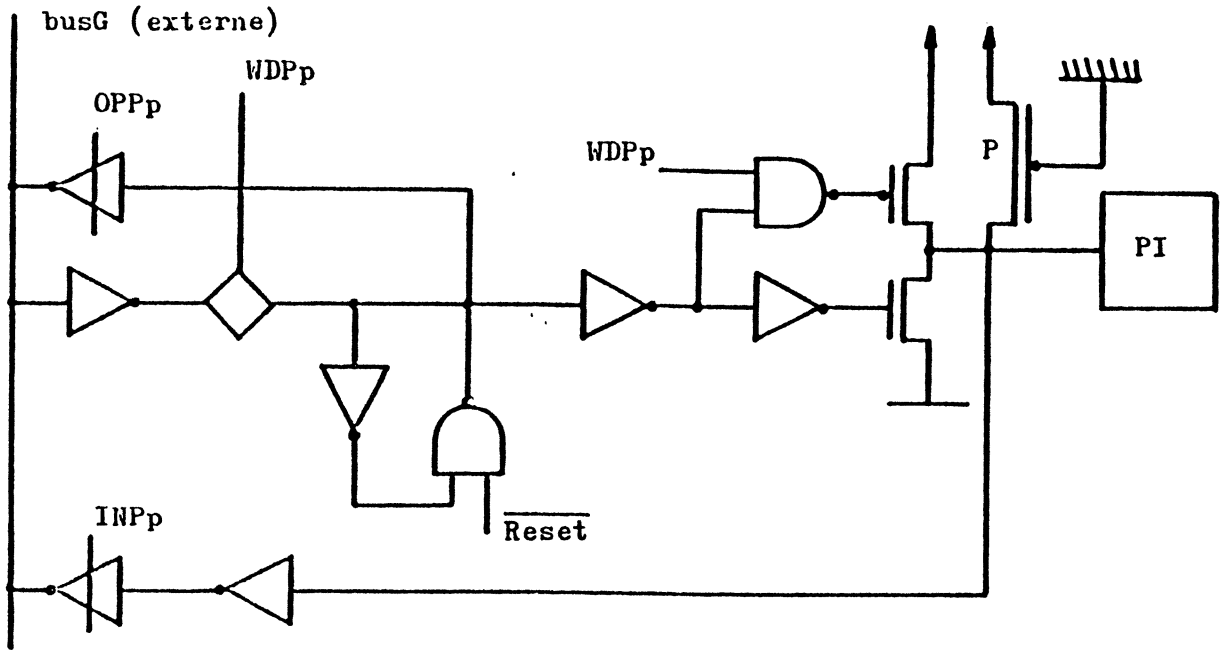


fig:III.56a Logique associé aux ports d'entrées/sortie "P1" et "P2"

On note une particularité au niveau des plots "P20" à "P23". Le fait de devoir exécuter les instructions "MOVD A,Pi, MOVD Pi,A, ANLD Pi,A et ORLD Pi,A" dans une configuration étendue des entrées/sortie (cf III 1-4c) nécessite de doubler les points mémoire associés à chaque bits; on évite ainsi l'écrasement par les données de l'adresse d'allocation contenue dans le port. La sélection de l'un des deux points mémoire est réalisée par une commande "SAD2".

- les huits lignes du port B véhiculent adresses et données sans autoriser le multiplexage des entrées et des sorties. (cf III 1-1g)

Un point mémoire (fig III.56b) constitué de deux inverseurs rebouclés permet de mémoriser l'adresse courante suite à l'activation du signal de chargement "WB". L'exécution de l'instruction "OUTL" permet d'écrire sur les plots par activation du signal d'écriture "WR".

On réalise une lecture par activation de la commande "RD" générée

durant l'exécution de l'instruction "INST".

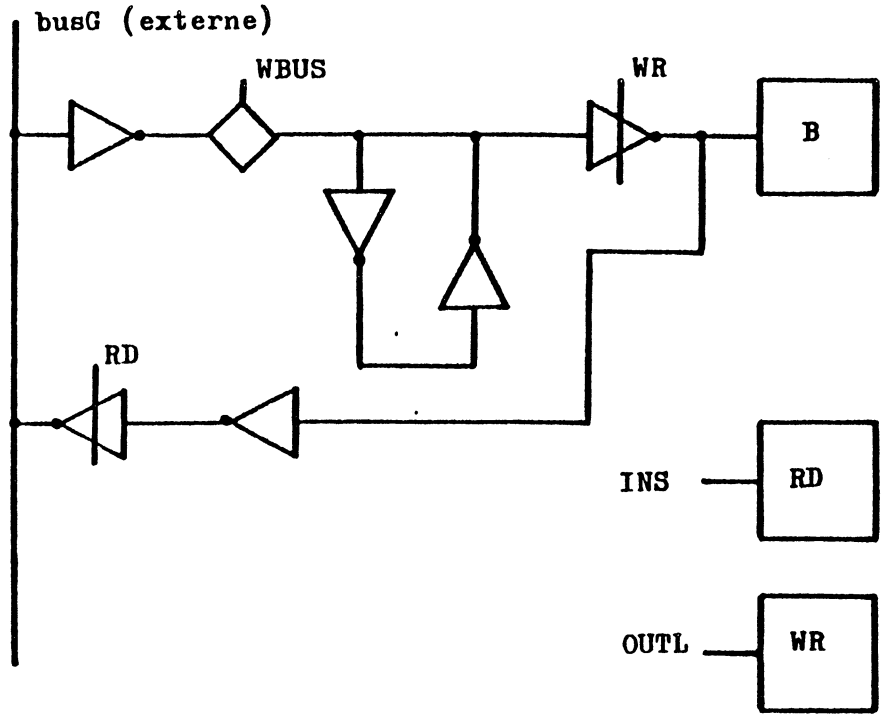


fig:III.56b Logique associée au port d'entrée/sortie "B"

2-3 Algorithme d'interprétation

Le jeu d'instructions (cf annexe 2) définit le langage de plus haut niveau du 80C48. Il détermine un ensemble de primitives, opérateurs et variables (registres, mémoires..), manipulés par l'interpréteur. Cet algorithme transforme un ensemble de données en un autre ensemble, ou traite des variables internes.

Les affectations permettent de définir un chemin de données reliant les différents blocs fonctionnels; on peut donc réaliser une partie opérative mais celle-ci ne peut exister en tant que structure physique.

ADD A,@R_r Add Data Memory Contents to Accumulator

0 1 1 0	0 0 0 r

The contents of the resident data memory location addressed by register 'r' bits 0-5 are added to the accumulator. Carry is affected.

(A)←(A) + ((Rr)) r=0-1

fig:III.57 Exemple: interpréteur de niveau II de l'instruction ADDC A,aRr. (INT48)

A cette description sont associées des contraintes temporelles inhérentes aux signaux de contrôle relatifs aux transferts d'informations (cf III 1-4).

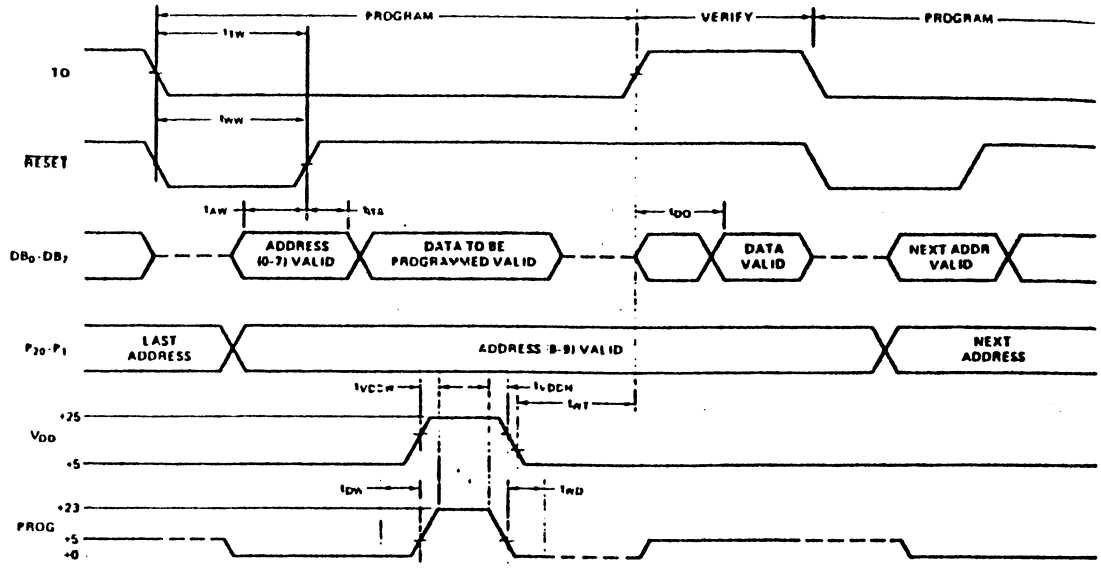


fig:III.58 Chronogrammes des signaux de synchronisation. (INT48)

L'expression de cet interpréteur en terme d'éléments matériel détermine l'algorithme d'interprétation de niveau "I0" (cf II 1); il ne met en oeuvre que des primitives matérielles constituant la machine physique. Cette description, et par là même la structure de la partie opérative, dépend néanmoins des choix pris au niveau supérieur face à la nécessité du compromis entre le coût (nombre d'éléments avec leurs interconnexions, surface de silicium) et la vitesse.

La partie opérative ainsi déterminée peut être plus ou moins bien adaptée à l'exécution du jeu d'instruction. Sa structure permet potentiellement l'exécution d'un certain nombre d'opérations, de transferts et de mémorisations à une vitesse déterminée.

Le 80C48 ne comportant que des instructions à durée déterminée de un ou deux cycles, les différentes opérations doivent être séquencées de façon à satisfaire aux spécifications ainsi définies. Au delà d'une certaine limite, une augmentation des capacités de traitement de la partie opérative n'est pas traduite par une rapidité accrue de l'exécution des instructions; le surcoût en matériel n'est donc pas justifié.

La partie contrôle est définie une fois la partie opérative spécifiée; elle découle directement des besoins en commande de la partie opérative. Toutefois, on peut être amené à modifier l'architecture du chemin de données, et donc l'algorithme d'interprétation s'y référant, pour simplifier la génération de certaines commandes. Ainsi une grande régularité dans l'exécution des instructions se traduit par des commandes aux équations simplifiées grâce à une paramétrisation importante.

a) Séquencement d'une instruction (au niveau des états)

On a vu (cf III 1-3b) que les instructions sont décomposées en un ou deux cycles machine, chaque cycle étant lui même subdivisé en cinq états: "S1", "S2", "S3", "S4", "S5". (S; "state")

D'autre part, l'exécution d'une instruction s'opère selon la séquence suivante:

- recherche du code opération et décodage,
- calcul de l'adresse du code opération suivant,
- exécution de l'instruction en cours et recherche de l'instruction suivante.

```
etc..
S5;
+- - - - - - - - cycle machine i - - - - - - - -+
S1; recherche du code    génération de l'adresse;
    opération;
S2; décodage du code     incrémentation compteur
    opération;           programme;
S3; exécution de         génération de l'adresse;
    l'instruction;
S4; exécution de         génération de l'adresse;
    l'instruction;
S5; exécution de         génération de l'adresse;
    l'instruction;
+- - - - - - - -  cycle machine i+1 - - - - - - -+
S1;
etc..
```

fig:III 59 Séquencement en cinq états d'une instruction d'un cycle

On définit un état comme étant une sous-phase du cycle machine; il lui correspond différentes opérations ou affectations du chemin de données.

La recherche du code opération de l'instruction suivante se fait à partir de l'état "S3" de l'instruction courante (travail en "pipe-line").

De la même façon, après avoir défini "Rg" comme étant un registre de travail, une instruction de deux cycles machine se formulera de la façon suivante:

```
etc..
S5;
+- - - - - - - - - - - - cycle machine i - - - - - - - - - - - -+
S1; RI <-- ROM(PC); * recherche du code opération *
S2; PC <-- PC + 1;  * incrémentation du compteur de programme *
S3;                 * exécution *
S4;  T <-- T + 1;   * incrémentation du temporisateur *
S5;                 * exécution *
S1; Rg <-- ROM(PC); * recherche valeur immédiate *
S2; PC <-- PC + 1;  * incrémentation du compteur de programme *
S3;                 * exécution *
S4;                 * exécution *
S5;                 * exécution *
+- - - - - - - - - - - - cycle machine i+1 - - - - - - - - - - - -+
S1;
etc..
```

fig:III.60 Séquencement d'une instruction de deux cycles

On se propose à titre d'exemple d'étudier de façon détaillée deux séquencements d'instructions typiques; leur détermination permet de définir une structure supportant le chemin de données.

a1) Exemple: instruction "ADDC A,Rr"

Il s'agit d'une instruction à un cycle machine permettant d'additionner l'accumulateur "A" et la retenue "C" au registre de travail "Rr" contenu dans la mémoire vive; le résultat de l'opération est transféré à l'accumulateur "A" et la retenue "PSW(7)" est réinitialisée:

```
A <-- A + PSW(7) + Rr;
PSW(7) <-- 0B;
```

Cette séquence peut se formuler avec les primitives suivantes:

```
+- - - - - - - - - - - - - - - cycle i - - - - - - - - - - - - - - - -+
S1; RI <-- ROM(PC);     * recherche du code opération*
S2; PC <-- PC + 1;      * incrémentation  du compteur programme *
S3;  A <-- A + Rr + PSW(7);
                        * addition *
S4;  T <-- T + 1;       * incrémentation du temporisateur *
S5;
PSW(7) <-- 0B;          * réinitialisation de la retenue *
+- - - - - - - - - - - - - - - cycle i+1 - - - - - - - - - - - - - - - -+
```

Séquence d'exécution de l'instruction "ADDC A,Rr"

On note que:

- le chargement du registre instruction "RI" se fait par le bus interne durant l'état "S1"; le code opération est ensuite décodé pour permettre l'exécution de l'instruction courante.

- on a vu que le compteur de programme "PC" (cf III 1-1c) se subdivise en deux parties, le champ relatif aux poids faibles "PCL" désignant l'octet de la page mémoire courante tandis que "PCH" sélectionne la dite page. L'incrémentation de "PCL" est réalisée en "S2", un éventuel débordement de ce compteur génère un saut à la page suivante par une incrémentation de "PCH".

On peut alors écrire:

```
PCL <-- PCL + 1;
(PCH <-- PCH + 1) si OVF= 1B; * "overflow" *
```

- d'autre part, l'accès à la mémoire vive est réalisé par l'intermédiaire du registre "W" (cf III 2-2K); la lecture de la valeur du mot mémoire "RAM(W)" n'est effective qu'après l'attente d'un état machine, suite à son adressage, ceci afin de respecter le temps de décodage nécessaire.

Ainsi, le fait de devoir disposer de l'opérande en "S3" nous oblige à adresser la mémoire en "S2".

- l'addition est réalisée à l'état "S3" par l'unité arithmétique et logique à deux entrées, la valeur de la retenue "PSW(7)" étant

introduite par la retenue entrante "CIN" (cf III 2-2d).

Suite à ces considérations, l'algorithme d'interprétation de l'instruction "ADDC A,Rr" peut se formuler de la façon suivante:

```
S1;      RI <-- ROM(PC);
        (PCL <-- PCL + 1);
S2;       W <-- CTE;
        (PCH <-- PCH + 1);
S3;      U2 <-- RAM(W);
S4;       A <-- A + U2 + PSW(7);
         (T <-- T + 1);
S5; PSW(7) <-- 0B;
```

Une affectation occupant le bus interne du microprocesseur, chaque échange consommera un état; se pose alors le problème du conflit de bus et de ressources. Pour ce faire, on incrémente le compteur de programme et le temporisateur à l'aide d'un additionneur spécifique "INC".

L'exécution de l'algorithme défini ci-dessous est supporté par la structure suivante:

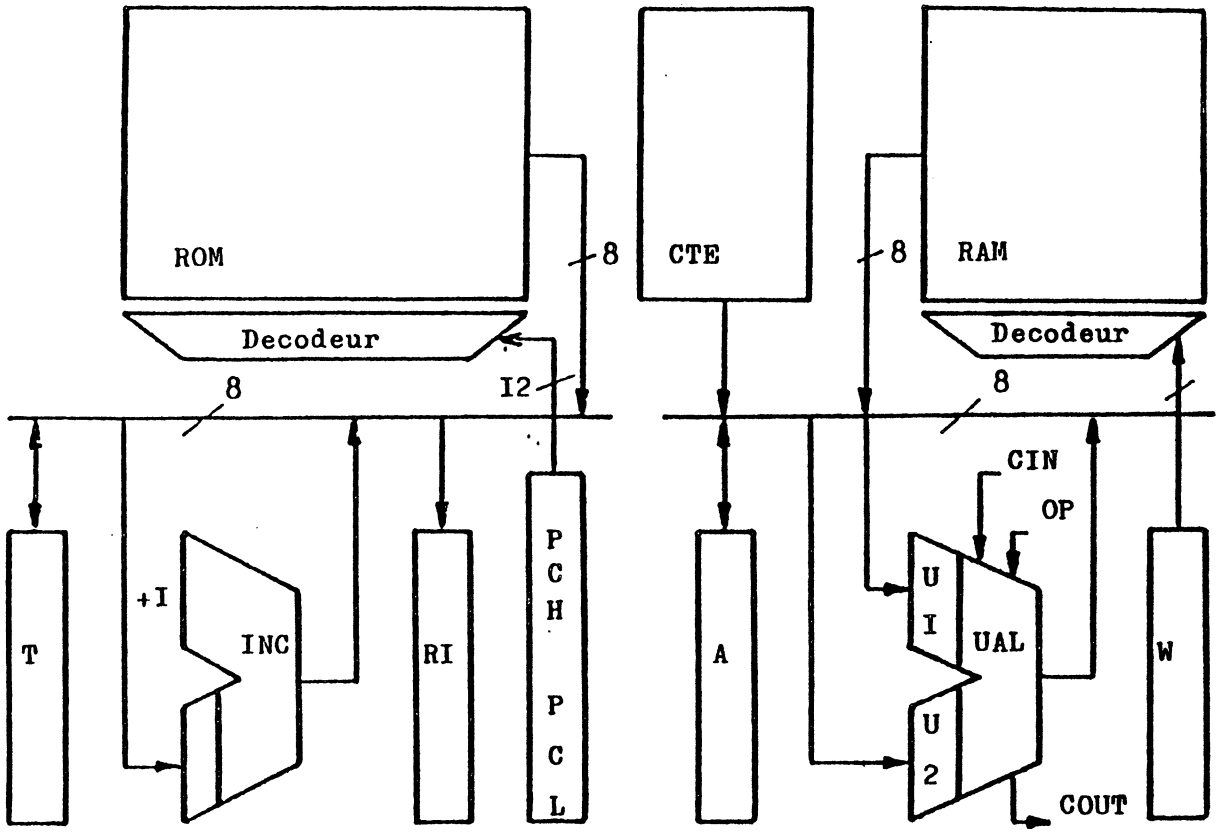


fig:III.61 Définition d'un chemin de données permettant l'exécution de l'instruction "ADDC A,Rr" (structure à un bus interne)

Ce schéma est bien sûr incomplet puisque l'algorithme d'interprétation n'a pas été pris en compte dans sa totalité, mais nous pouvons imaginer qu'il représente une partie de la machine.

a2) Exemple: instruction ADDC A,aRr

Cette instruction est similaire à la précédente mis à part le fait que la mémoire vive est adressée indirectement par le registre "R0" ou "R1" (cf III 2-2k).

La valeur du registre "W" doit être maintenue pendant le temps d'accès à la RAM; pour ce faire, il est nécessaire de disposer d'un

registre de travail "Rr" qui permet de sauvegarder l'adresse indirecte.
La lecture de l'opérande se fait alors selon la séquence suivante:

```
W  <-- CTE;     * adresse du registre R0/R1 *
Rr <-- RAM(W); * adresse indirecte *
W  <-- Rr;
U2 <-- RAM(W); * lecture de l'opérande *
```

fig:III.62 Séquence d'adressage indirect à la mémoire vive résidente

L'indirection fait apparaitre deux étapes supplémentaires pour l'exécution de l'instruction "ADDC A,aRr"; on dépasse ainsi la durée du cycle machine.

Les deux instructions choisies sont critiques par le nombre d'affectations qu'elles imposent durant un cycle machine.
Il apparait que ce modèle de séquencement ne nous permet pas d'apporter une solution aux différents problèmes, d'où la nécessité de pouvoir effectuer plusieurs transferts durant un état machine; pour ce faire, on propose un système biphasé associé à une architecture à deux bus internes.

b) Séquencement d'une instruction (au niveau des phases)

b1) Définition des phases

Le séquencement des instructions repose sur un système de deux phases à non recouvrement "P1" et "P2"; le cycle machine étant de 500 nanosecondes, la phase est active pendant 166 nanosecondes, le non recouvrement représentant 83 ns.

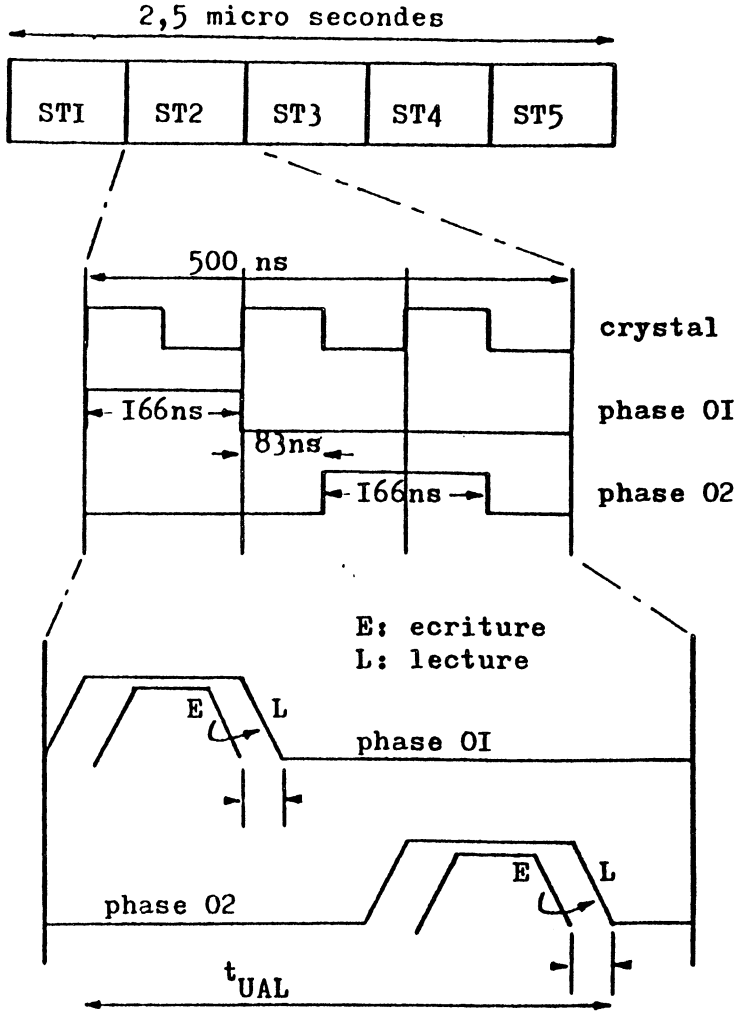


fig:III.63 Chronogramme des phases "P1" et "P2"

Une phase machine autorise le transfert de registre à registre (ou d'opérateur à registre) par un signal de lecture "R" et d'écriture "W". Le signal de lecture est généré à partir du signal d'écriture, le retard "t" ainsi produit garantit un bon tranfert de l'information par la stabilisation des bus.

b2) Choix d'une structure

L'utilisation d'une architecture à deux bus internes ne coûte pas trop cher en surface de silicium si l'on utilise la propriété de transparence des briques (cf 2-6, SUZ81).

En contre partie, une telle structure autorise le travail en parallèle permettant ainsi souplesse et régularité de l'algorithme d'interprétation.

Nous désignerons par "alpha" et "bêta" les bus internes.

Ainsi, les registres tampons "U1" et "U2" de l'UAL sont chargés systématiquement sur la première phase d'un état, le résultat est transféré durant la phase suivante; ceci permet à l'unité arithmétique et logique de disposer de deux phases machines pour effectuer les calculs. Seules les adresses des sources "X" et "Y" sont à spécifier.

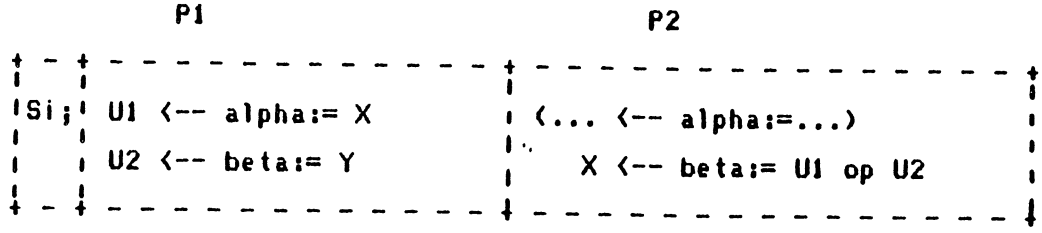


fig:III.64 Opération de l'unité arithmétique et logique
P1/ chargement des opérandes
P2/ lecture du résultat

Les conditions ne portent que sur le paramétrage de l'opération "op" et l'exécution du chargement en "P2".

Les échanges avec l'environnement du microprocesseur sont effectués par l'intermédiaire d'un bus périphérique "gamma", relié de façon bidirectionnelle au bus "bêta"; il permet de distribuer les données et leurs signaux de validation aux broches d'entrées/sorties du microprocesseur.

On obtient ainsi un niveau supplémentaire de parallélisme entre la saisie d'information et son traitement par le CPU.

b3) Choix d'un séquencement

La mémoire morte résidente devant présenter une superficie minimale, son temps d'accès sera important; un minimum de trois états machine représentant 1,5 microsecondes est à considérer comme un impératif. Afin de conserver un coté systématique à l'algorithme d'interprétation, la lecture du mot mémoire se fait sur la phase "P1" de l'état considéré.

Son adressage s'éffectue en deux temps - poids faibles/poids forts -

sur les états "S1" et "S2" du cycle; un accès en mémoire externe est réalisé par sauvegarde de la valeur de "PCL" et "PCH" sur les registres tampons (respectivement) "BL" et "AP2L" d'entrées/sorties; l'affectation sur les plots se fait sur le front descendant de "ALE". "PSEN*", actif en (S5, S6) valide ensuite la recherche du code opération suivant (cf III 1-4a).

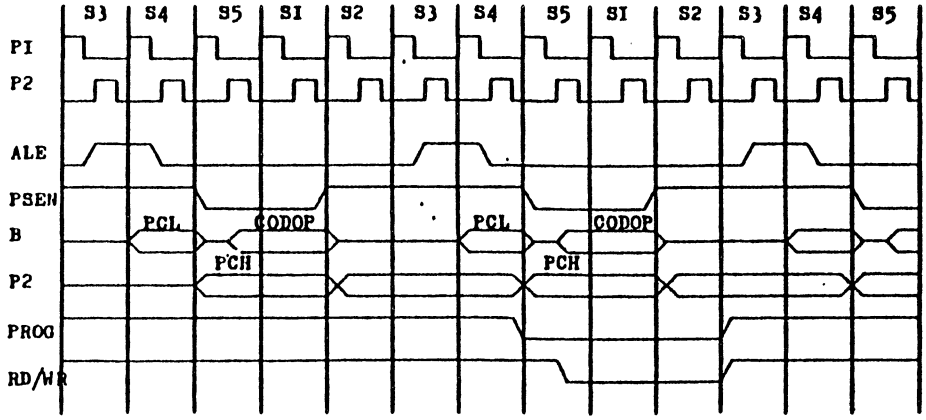
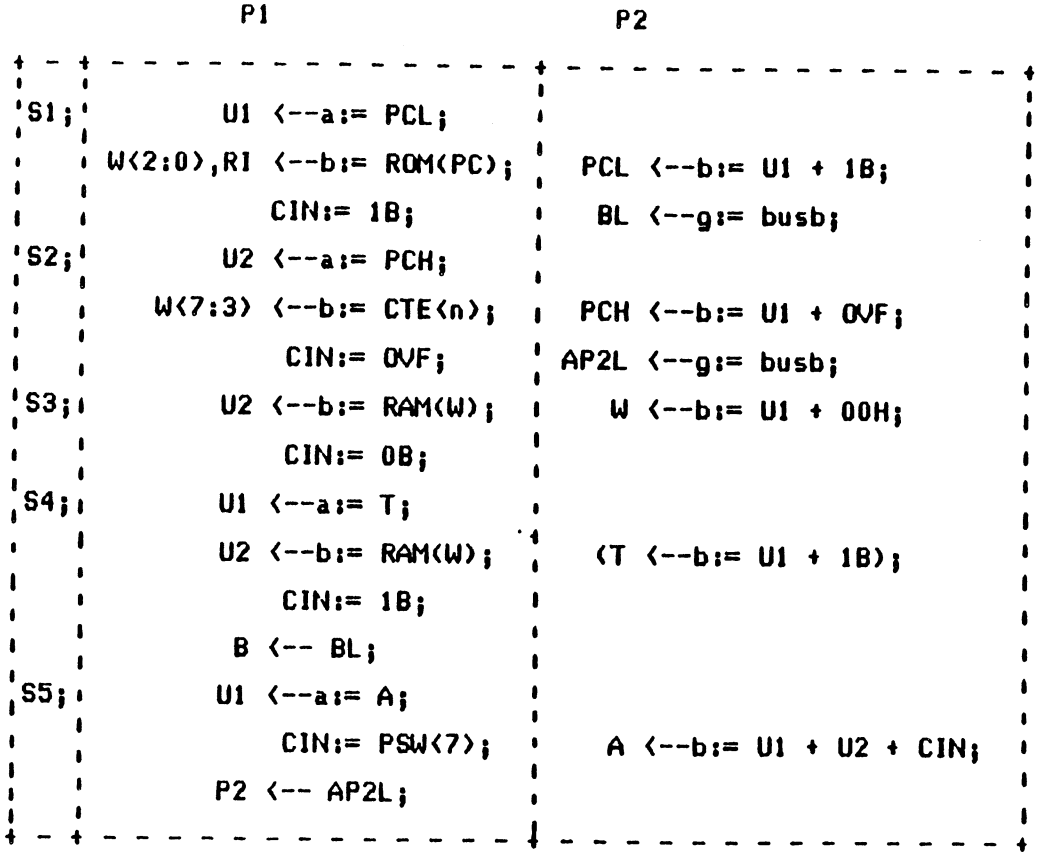


fig:III.65 Adressage de la mémoire externe

On a vu précédement que le temps d'accès à la mémoire vive est de un état machine. Son adressage est réalisé de façon systématique sur la phase "P1", la lecture du mot mémoire s'opère sur la même phase de l'état suivant.

Au vu de ces considérations générales, l'instruction "ADDC A,aRr" se formule de la façon suivante:



si MB = 1B, CTE<5> = 18H;
si MB = 0B, CTE<0> = 00H;

Note: une architecture à deux bus permet une plus grande disponibilité de l'unité arithmétique et logique; on réalise l'incrémentation du compteur de programme et du temporisateur en forçant la retenue entrante "CIN" à l'état haut.

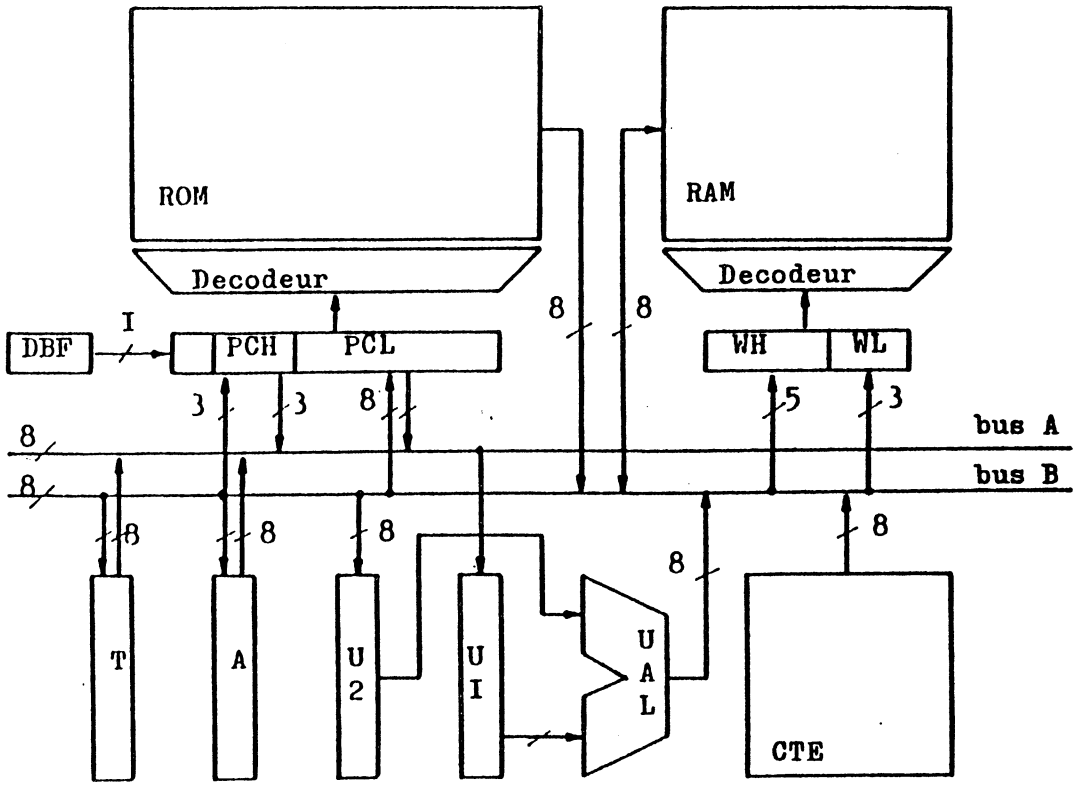


fig:III.66 Définition du chemin de données permettant l'exécution de l'instruction "ADDC A,aRr" (structure à deux bus internes)

La généralisation de l'étude au jeu d'instruction du 80C48 permet de proposer un séquencement "type" d'exécution de l'algorithme d'interprétation; cette nouvelle approche présente des actions systématiques:

- les étapes "S1" et "S2" sont identiques à toutes les instructions; le décodage du registre instruction peut être lent, la paramétrisation n'intervenant qu'à l'état "S3".

- l'incrémentation (conditionnelle) du timer est réalisée par l'UAL à l'état "S4".

- les étapes "S6" et "S7" sont fixes pour toutes les instructions à deux cycles.

- la mémorisation des bascules d'états a lieu en "S5" et "S10" pour les instructions de un et deux cycles respectivement.

- on simplifie tous les transferts en passant systématiquement par l'UAL; le chargement des opérandes est réalisé sur la phase "P1", la lecture du résultat en "P2". Seules les adresses des sources sont à spécifier, le paramétrage porte sur la validation du résultat et la fonction réalisée par l'opérateur. (DEM82, APL82, APL83)

De plus, on s'efforce à ne mettre en oeuvre que des registres à simple accès - lecture/écriture sur un seul bus - pour réduire le nombre de lignes de commande, et donc la longueur de la partie opérative.

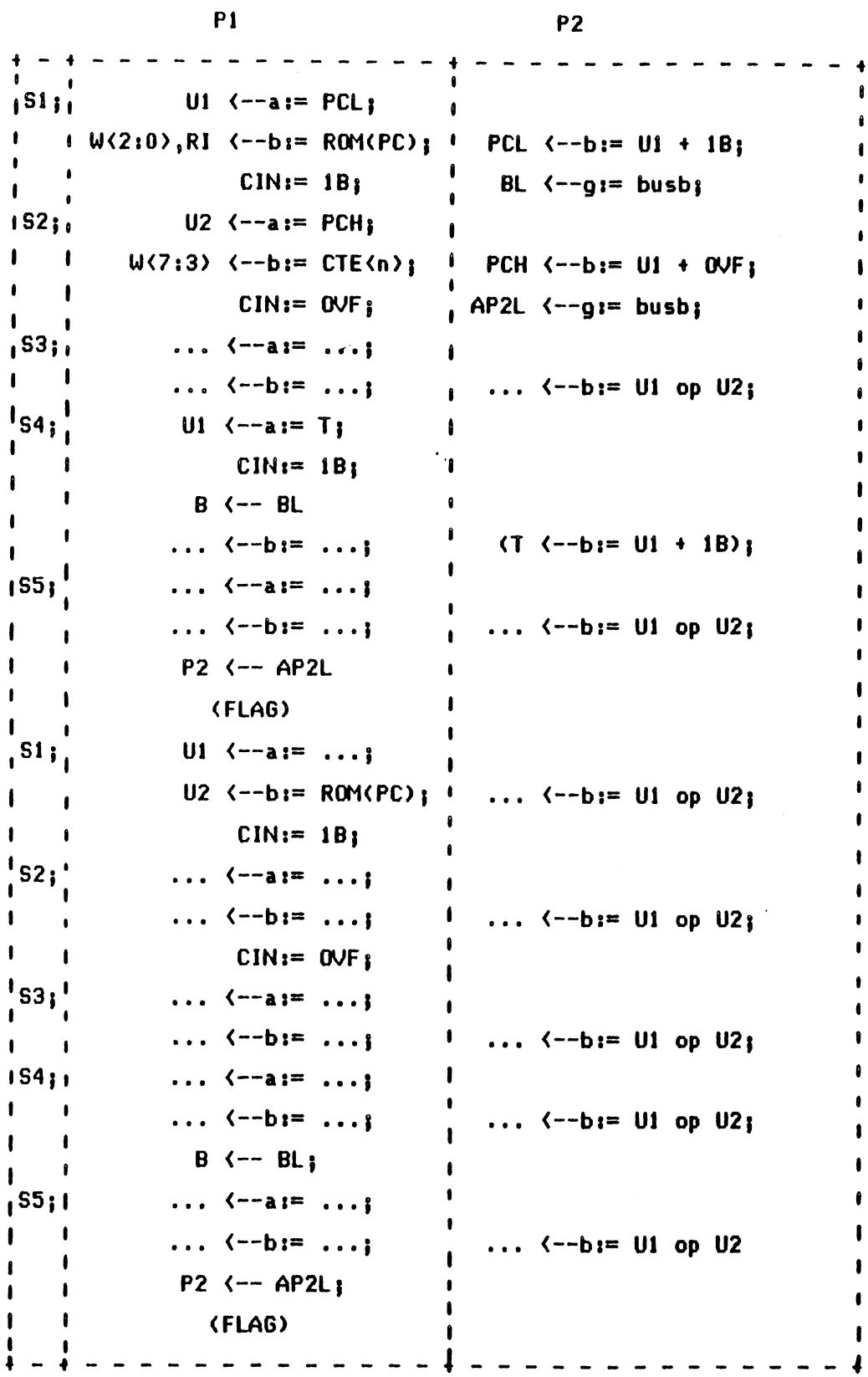


fig:III.67 Séquencement "type" d'une instruction de deux cycles

( ):= conditionnel

op:= +, ., ou, oux;

si MB = 0B, CTE(0) = 00H;

si MB = 1B, CTE(5) = 18H;

L'usage de ces "contraintes simplificatrices" permet une grande similitude dans l'exécution des instructions, beaucoup de commandes sont systématiques. Une partie contrôle très simple se dégage avec une paramétrisation importante.
(cf III 3)

c) interpréteur des instructions du 80C48

On définit l'algorithme d'interprétation en généralisant l'application du modèle de séquencement défini précédemment au jeu d'instructions du 80C48 (APL82, APL83, ALG83).

Le passage des spécifications d'un niveau donné aux spécifications de niveau inférieur - dit de base - se fait en explicitant l'algorithme d'interprétation en termes de primitives matérielles.

Les algorithmes sont écrits à l'aide de deux langages, le "PASCAL" - pour décrire et simuler le comportement de la machine -, et "IRENE" (IRE82) - pour définir les caractéristiques opérationnelles ou algorithmiques non contenues dans le "PASCAL" (transferts conditionnels ...etc).

Des instructions de la même famille (accumulateur, entrées/sorties, registre, branchements, sous-routine, transferts de données, compteur/temporisateur, contrôle) présentent de grandes similitudes, beaucoup de séquences étant identiques; on les regroupe en une seule description paramétrée dont le décodage ne porte que sur le choix de l'opération "op" et/ou la condition des bascules d'états.

Ce paramétrage diminue de façon importante les cas d'espèce et permet ainsi de simplifier la génération des équations de commandes par un nombre réduit de propriétés (cf III 3-1b).

Les instructions sont définies par des séquences d'actions qui mettent

en oeuvre une suite d'affectations (ex: X (--a:= Y); cette description spécifie les chemins de données et les opérateurs mis en jeu. La transcription en MACSIM (cf annexe 4) permet de simuler l'algorithme d'interprétation et d'en dégager le contenu de la partie contrôle et la structure du chemin de données.

Exemple:

```
procedure ACP1 (param:interger); "intructions de type: A op aRr --) A"
(* inst: ADDC A,aRr; ADD A,aRr; ANL A,aRr; ORL A;aRr; XRL A,aRr;
                                                        VER.B *)
 begin
   CYCLE(param,1);
            (* CODOP1= 01xx 1xxxB; *)
            (* CODOP2= 1101 1xxxB; *)
        DBBI:=B; "lecture ROM externe"
        busa:= PCL; U1L:= busa mod 8; U1H:= busa div 8;
        opual:= U1p0; cin:= 1; "incré. compteur prog. poids-faibles"
        busg:= DBBI;
        (* busb:= if STH(3) = 0B then ROM(PC) else busg; *)
        if STH(3) then busb:= ROM(PC)
                  else busb:= busg; "codop interne où externe"
                                    "sélect. ROM ext./ROM int. par DBF"
        RI:= busb; "codop inst. courante"
        WL:= busb mod 8; "adresse RAM poids-faibles"
   CYCLE(param,2);
        busb:= sual; PCL:= busb; "sauvegarde compt. prog. poids-faibles"
        busg:= busb; DBBO:=busg; "mém. adresse poids-faibles"
   CYCLE(param+1000,1);
        busa:= BAS30; U1L:= busa mod 8;; U1H:= busa div 8;
        opual:=U1p0; cin:= PSH(3); "incré. compt. prog. poids-forts"
        (* busb:= if STH(4)=0B then CTE(0) else CTE(5); *)
        if STH(4)=0 then busb:= CTE(0)
                    else busb:= CTE(5); "sélect. banque mémoire vive"
        WH:= busb div 8; "adresse RAM poids-forts"
   CYCLE(param+1000,2);
        busb:= sual; BAS20:= busb mod 8; BAS(3):= bit(3,busb);
        "sauvegarde compt. prog. & basc. sélect. mémoire DBF"
        busg:= busb; AP2L:= busg mod 16; "mém. adresse poids-forts"
   CYCLE(param+2000,1);
        busb:= RAM(W); U2:= busb; "adresse indir. mémoire vive"
        opual:= 0pU2; cin:= 0;
   CYCLE(param+2000,2);
        busb:= sual; WL:= busb mod 8; WH:= busb div 8; "adressage RAM"
   CYCLE(param+3000,1);
        busa:= T; U1L:= busa mod 8; U1H:= busa div 8;
```

```
        opual:= U1p0; cin:= 1; "incré. registre temporisateur"
        busb:= RAM(W); U2:= busb; "lecture premier opér."
        B:= DBB0; "adressage ROM ext., poids-faibles"
    CYCLE(param+3000,2);
        busb:= sual;
        (* T:= if PRES then busb; *)
        if PRES then T:= busb; "sauv. reg. temporisateur"
        "conditionné par débordement diviseur (%32) horloge"
    CYCLE(param+4000,1);
        busa:= A U1L:= busa mod 8; U1H:= busa div 8;
        "lecture deuxième opérande"
        (* opual:= case RI(7:4) of;
                '0111: U1pU2;
                '0110: U1pU2;
                '0101: U1.U2;
                '0100: U1ouU2:
                '1101: U1ouxU2
                                endcase; *)
        "sélec. opération instruc. courante"
        case param of
          112,96:opual:= U1pU2;
              80:opual:= U1.U2;
              64:opual:= U1ouU2;
             208:opual:= U1ouxU2
        end;
        (* cin:=if RI(7:4)= 0111 then PSH(7) else 0; *)
        if param=120 then cin:= PSH(7)
                     else cin:= 0; "charg. cond. retenue C"
        P2:= AP2L; "adressage ROM ext., poids-forts"
    CYCLE(param+4000,2);
        busb:= sual; AL:= busb mod 16; AH:= busb div 16;
        "sauv. résultat"
FININST; end;
```

fig:III.68 Description "MACSIM" de l'interpréteur de l'instruction "ADDC A,aRr"

d) simulation fonctionnelle (statique)

La simulation fonctionnelle de la description "MACSIM" (cf annexe 4) permet de valider l'algorithme d'interprétation qui réalise le jeu d'instructions du 80C48.

L'exécution des étapes d'interprétation autorise un suivi des opérandes et de leurs affectations; le concepteur peut à tout instant connaître l'état d'une variable et/ou lui affecter une valeur.

On vérifie par là la fonctionnalité des opérateurs requis, le bon enchaînement des séquences d'exécution et la syntaxe de la description. Cette analyse permet de faire apparaître les éventuels problèmes de conflit de bus ou de ressource.

L'analyse - au niveau de chaque cycle - des mots-clés distinctifs des champs de commandes permet de dresser les tables relatives aux ressources (opérandes) mises en oeuvre. On peut ainsi affecter un code à chaque champ de micro-instructions dont la valeur est explicitée par l'interpréteur (cf annexe 3).

Les instructions sont décrites sous la forme codée textuelle suivante:

CODOP= 01XX000X 1101000X; "codop instruc. courante"
INSTRUCTION= ACP1; "nom procédure"

ETATS:	S0	S1	S2	S3	S4
PHASES:	12	12	12	12	12
DBBI:=	1.	..	..	..	..
B:=	..	..	..	1.	..
DBBO:=	.1	..	..	..	..
AP2L:=	..	.1	..	..	..
P2:=	..	..	..	..	1.
busa:=	1.	2.	..	3.	4.
busb:=	12	32	42	42	.2
busg:=	12	.2	..	..	..
WL:=	1.	..	.1	..	..
WH:=	..	1.	.1	..	..
PCL:=	.1	..	..	..	..
RI:=	1.	..	..	..	..
T:=	..	..	..	.1	..
U1L:=	1.	1.	..	1.	1.
U1H:=	1.	1.	..	1.	1.
U2:=	..	..	1.	1.	1.
AL:=	..	..	..	..	.1
AH:=	..	..	..	..	.1
opual:=	1.	1.	3.	1.	2.
cin:=	1.	2.	4.	1.	3.
BAS20:=	..	.1	..	..	..
BAS(3):=	..	.1	..	..	..

--> exemple:

(S0,P1): busa:= PCL;
(S1,P1): busa:= BAS30;
(S3,P1): busa:= T;
(S4,P1): busa:= A;

fig:III.69 Exemple de description codée des séquences d'affectation relatives à l'exécution de l'instruction "ADDC A,aRr"

A ce niveau, la progression se fait pas à pas, le concepteur valide toute nouvelle affectation de champ; on réitère cette phase de simulation jusqu'à obtenir une description rigoureuse et conforme aux spécifications initiales.

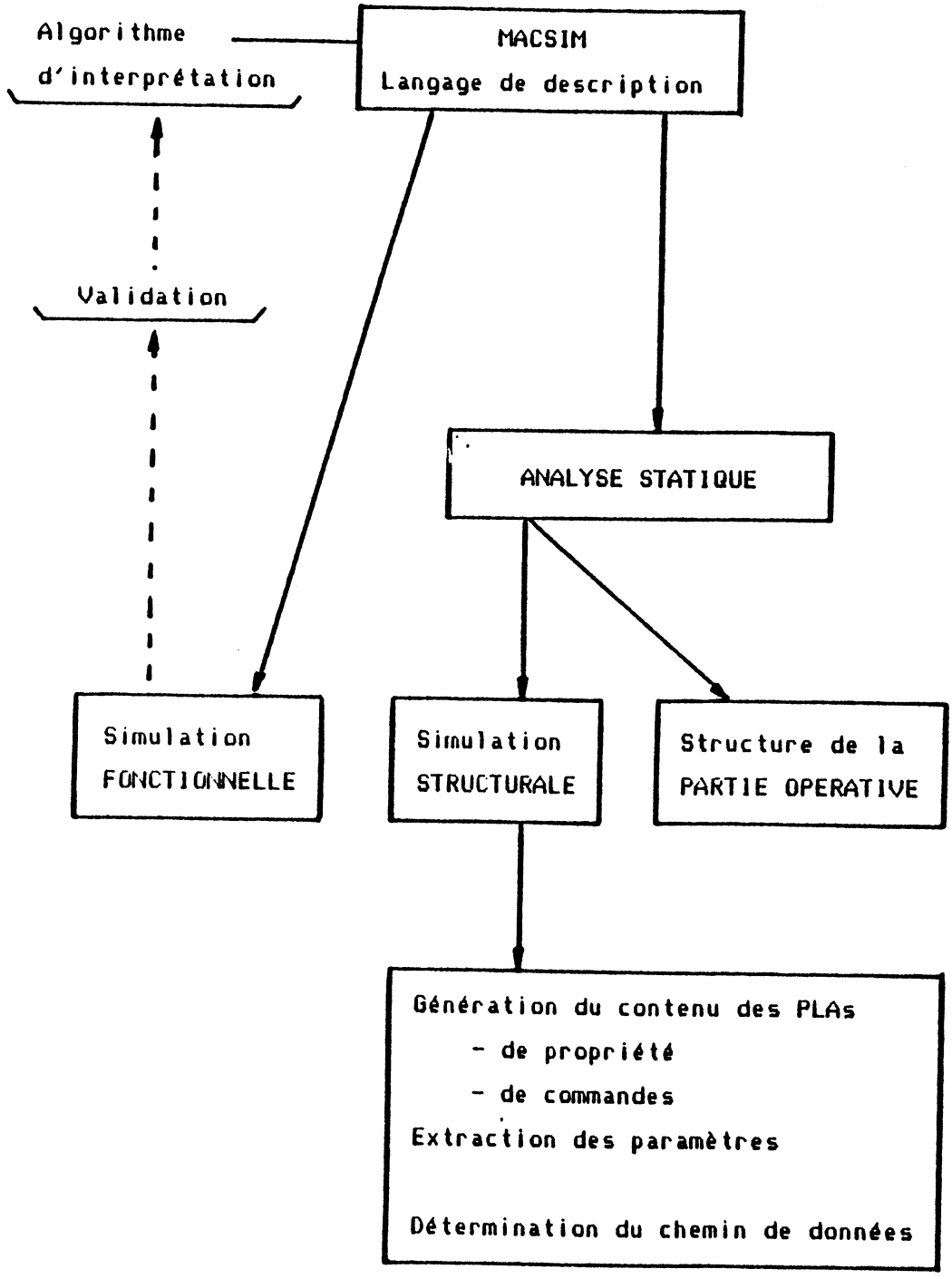


fig: III.70 Environnement de "MACSIM"

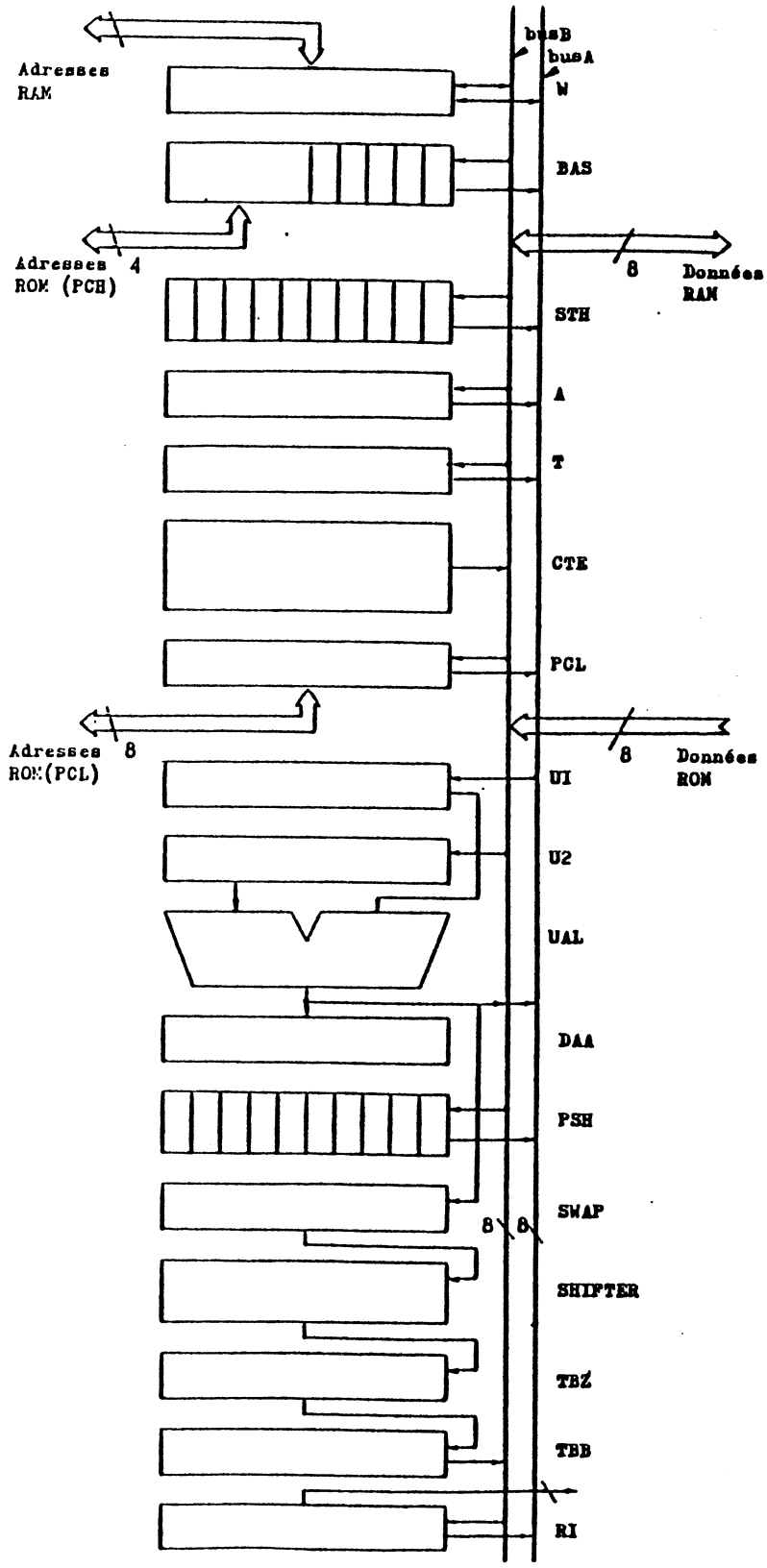
2-4 Architecture de la partie opérative

L'analyse structurale de la description "MACSIM" de l'interpréteur de phase permet d'établir un bilan des connexions caractérisant le chemin de données; on établit ainsi une relation de surjection entre l'ensemble des variables (sources) et leurs affectations (objets).
(cf annexe 3 & 4)

La grande répétitivité dans l'interprétation des instructions a permis de minimiser le nombre de types d'échange. Ces liaisons peuvent être conditionnées à l'état d'une variable interne ou d'une bascule d'état.

L'ensemble des connexions qui mettent en jeu les différents blocs fonctionnels (registres, opérateurs, bascules d'états...) définit l'architecture de la partie opérative (fig III.71).

fig:III.71 Architecture de la partie opérative du 80C48



2-5 Simulation électrique

Une simulation électrique constitue la suite logique de la conception architecturale d'un circuit (fig III.72); elle permet de déterminer le comportement électrique - niveaux logiques, temps de réponse - des ressources mises en oeuvre et de résoudre les problèmes inhérents aux interactions entre les différents blocs fonctionnels. (cf annexe 5)

Dans la pratique, on ne simule que des élémdntr logiques ne mettant en jeu qu'une cinquantaine de transistors au maximun.

Contrairement au NMOS pour lequel le rapport entre temps de montée et de descente est à peu près égal à celui des géométries des transistors, le CMOS a pour seule différence entre le temps de montée et de descente celle existant entre la mobilité des trous et des électrons; on obtient des temps de réponse identiques en dimensionnant les transistors "P" une fois et demie plus gros que les transistors "N".

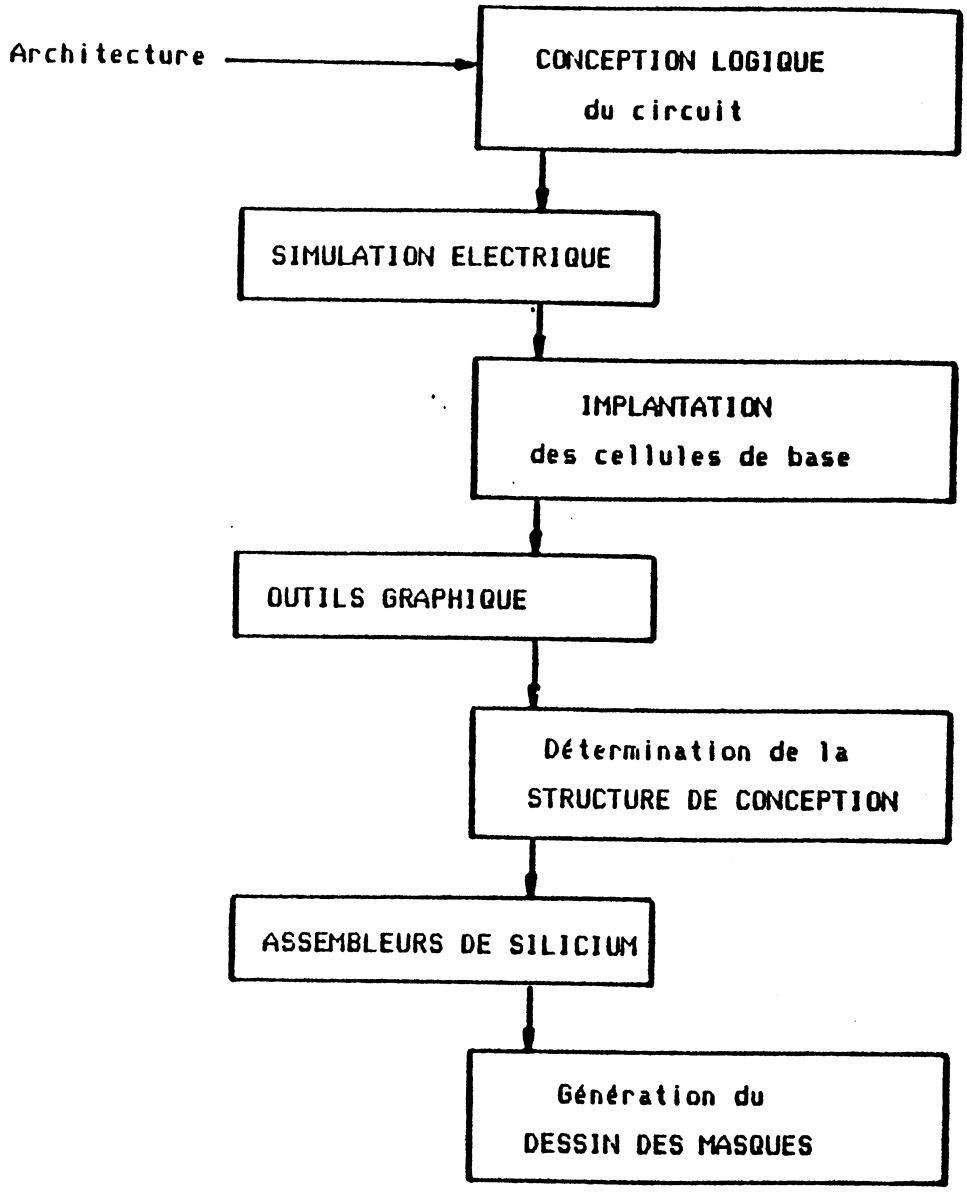


fig:III.72 Synoptique d'utilisation des outils de CAO

Les portes logiques de base sont implantées avec une géométrie minimale; seuls les points critiques - attaque de bus, amplificateurs d'entrées/ sorties etc..- que constituent les éléments de puissance font l'objet d'un dimensionnement adapté. On détermine la taille des transistors par simulations successives avec une idée de départ donnée par les règles.

exemple de simulation de points critiques:

- les transistors qui attaquent la chaîne de retenue de huit bits de l'unité arithmétique et logique sont dimensionnés pour obtenir un temps de propagation de 5 nanosecondes par bit.

- on évalue à 46 nanosecondes le temps de propagation d'un état logique le long du bus en polysilicium attaqué par le point mémoire établi précédement, contre 31 nanosecondes avec un bus métallique; ce délai est acceptable, la durée d'une phase représentant 200 nanosecondes avec un cristal de 6 MHz. Le choix d'un bus polysilicium a été fait sur des critères purement topologiques, la vitesse ne constituant pas un élément critique.

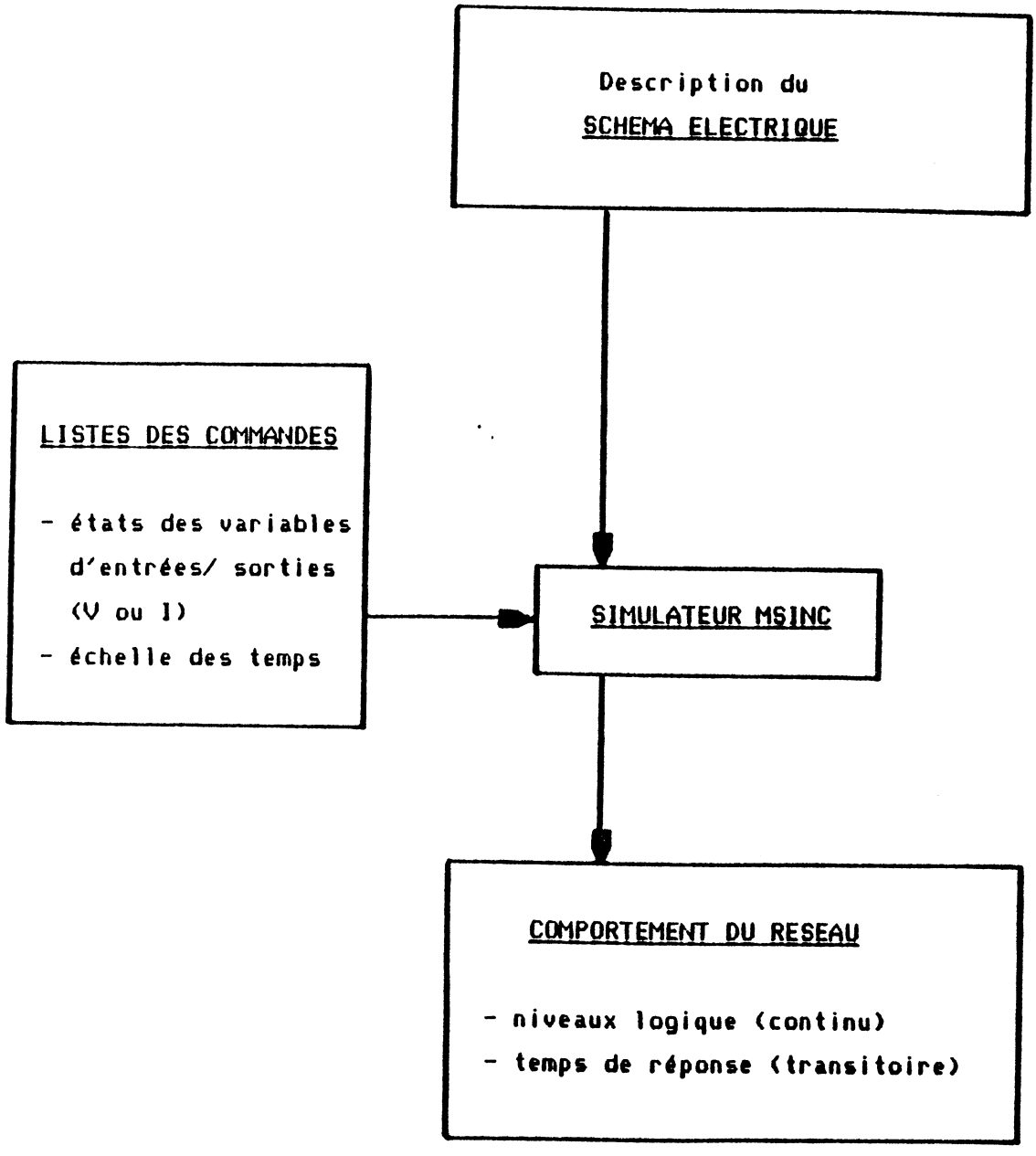


fig:III.73 Environnement du simulateur électrique MSINC

2-6 Implantation de la partie opérative

L'algorithme d'interprétation définit les fonctions logiques des différentes ressources (éléments de mémorisation, de communication, opérateurs) qui constituent la partie opérative du 80C48; leurs dessins au micron permet de les implanter.(SAH84, MAT84)

La génération du dessin des masques constitue une des étapes les plus coûteuses dans le processus d'élaboration de circuits intégrés complexes. L'expérience prouve (SUZ81) que l'utilisation d'une implantation régulière minimise les risques d'erreurs et donc la durée de la conception tout en autorisant une forte densité d'intégration.

On sait que les interconnexions sont coûteuses en surface occupée et que leurs longueurs pénalisent la vitesse de fonctionnement. Ainsi, on réduit les liaisons entre les différents blocs fonctionnels par une meilleure étude topologique de l'ensemble du circuit plutôt que de chercher à optimiser chaque élément.

Leurs positionnements respectifs doivent être établis dès que possible lors de la phase de détermination fonctionnelle. On peut alors procéder à une optimisation tant logique que topologique de la structure précédemment établie; certaines modifications peuvent remettre en cause de façon partielle l'algorithme d'interprétation.

exemples:

- on a été amené à regrouper sous forme de registres "STH" ,"BAS", "PSH" (fig III.71) les différentes bascules d'états contenues dans la partie opérative; cette nouvelle configuration intervient sur leur mode d'adressage et doit être prise en compte par l'interpréteur.

- on détermine le positionnement respectif des différents registres (ou opérateurs) selon leur degré d'inter-actions. Ainsi, le registre "PSH" contenant la retenue "C", la retenue auxiliaire "AC" et la bascule de débordement "OVF" jouxte l'UAL, de même que les différents opérateurs de test.

On définit une structure d'accueil selon la méthode dite du "bit slice"

(tranche de bit) (SUZ81); elle permet de voir une partie opérative de "n" bits comme une répétition de "n" parties de un bit.

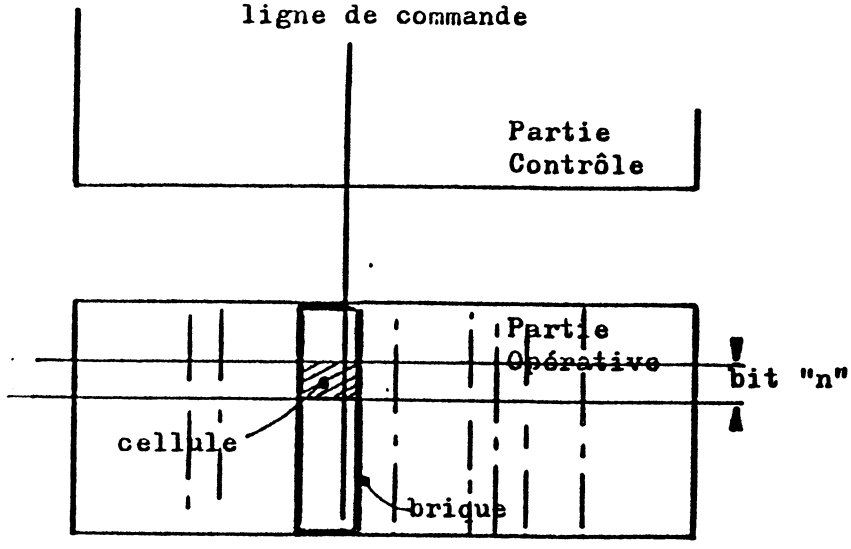


fig:III.74 Structure "bit slice"

Un bloc fonctionnel (ou brique) est défini par assemblage de cellules élémentaires (cf annexe 5 & 9). La trentaine de cellules qui ont permis de générer la partie opérative du 80C48 constitue une bibliothèque de fonctions (SAH83).

Une cellule est caractérisée par une certaine transparence vis à vis du monde extérieur; c'est à dire qu'elle permet le passage au dessus d'elle d'un certain nombre d'interconnexions où elle peut elle même venir se connecter.(fig III.75) Cette propriété permet de s'affranchir de zones de liaisons pures au sein de la partie opérative.

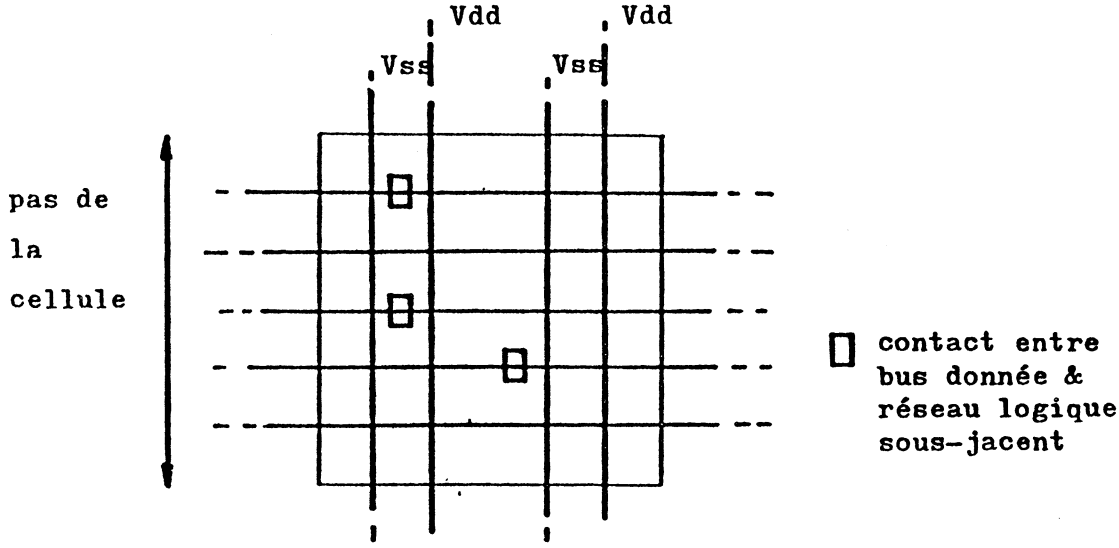


fig: III.75 Structure d'une cellule

On détermine le pas d'une cellule en implantant des briques considérées comme critiques (UAL, tampons d'entrée d'UAL, registres d'états) par le taux d'intégration qu'elles nécessitent. Une fois établie, cette hauteur sert de référence pour l'implantation de toute la partie opérative.

La méthode d'implantation choisie consiste à disposer verticalement les lignes d'aluminium et horizontalement les lignes de polysilicium; les commandes et les peignes d'alimentation seront donc en métal, le chemin de routage en polysilicium.

Ainsi, on réalise facilement un transistor "N" ou "P" en disposant un barreau de diffusion proche des lignes d'alimentation "Vss" et "Vdd", respectivement (fig III.76).

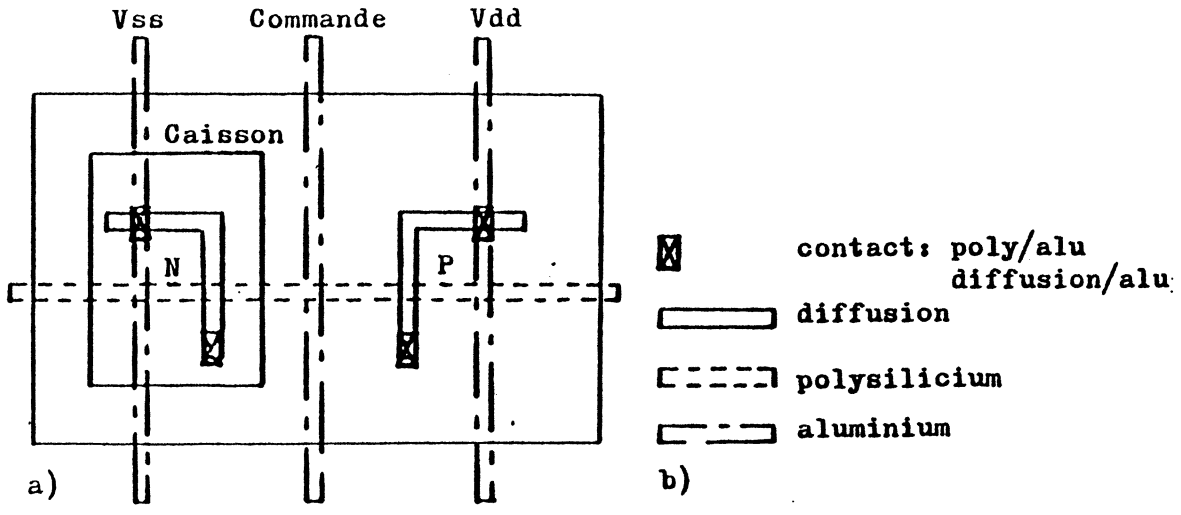


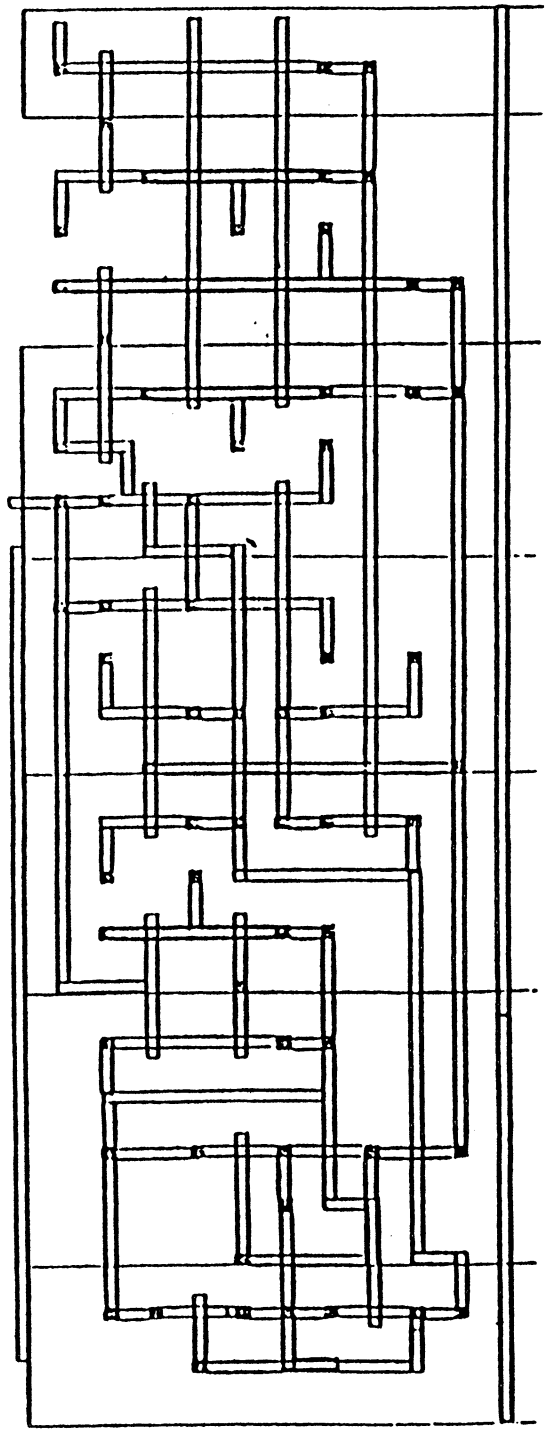
fig: III.76 Exemple d'implantation de transistors
a) représentation/ b) symbolisme

Une telle méthode facilite l'implantation tout en autorisant une grande densité d'intégration. La résistivité non négligeable de polysilicium n'entraine pas de temps de propagation critique pour le chemin de données; sa détermination à fait l'objet d'une simulation électrique.

a) implantation "stick" de la partie opérative

On réalise une première implantation de la partie opérative du 80C48 à l'aide d'un symbolisme nommé "stick" (baton); cette technique consiste à représenter de façon squelétisée les portes logiques tout en s'affranchissant des problèmes de dimensionnement des transistors.

fig:III.77 Représentation en "stick" de l'UAL



Fidèle aux règles de dessin (problème de garde, disposition topologique), cette réalisation permet une mise au point générale plus rapide et donne une idée de l'encombrement d'une figure.

On passe ensuite facilement de la représentation "stick" à la génération du dessin des masques.

L'emploi de ce symbolisme se fait au détriment de la surface occupée; en prenant comme densité de référence celle obtenue grâce au dessin au micron, on peut considérer que l'utilisation du "stick" apporte une dégradation de la densité de l'ordre de 20%.

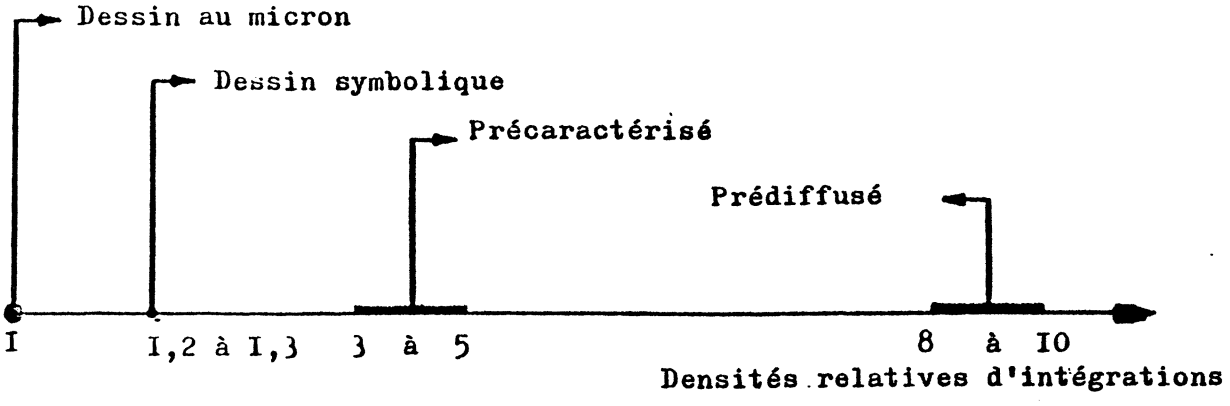


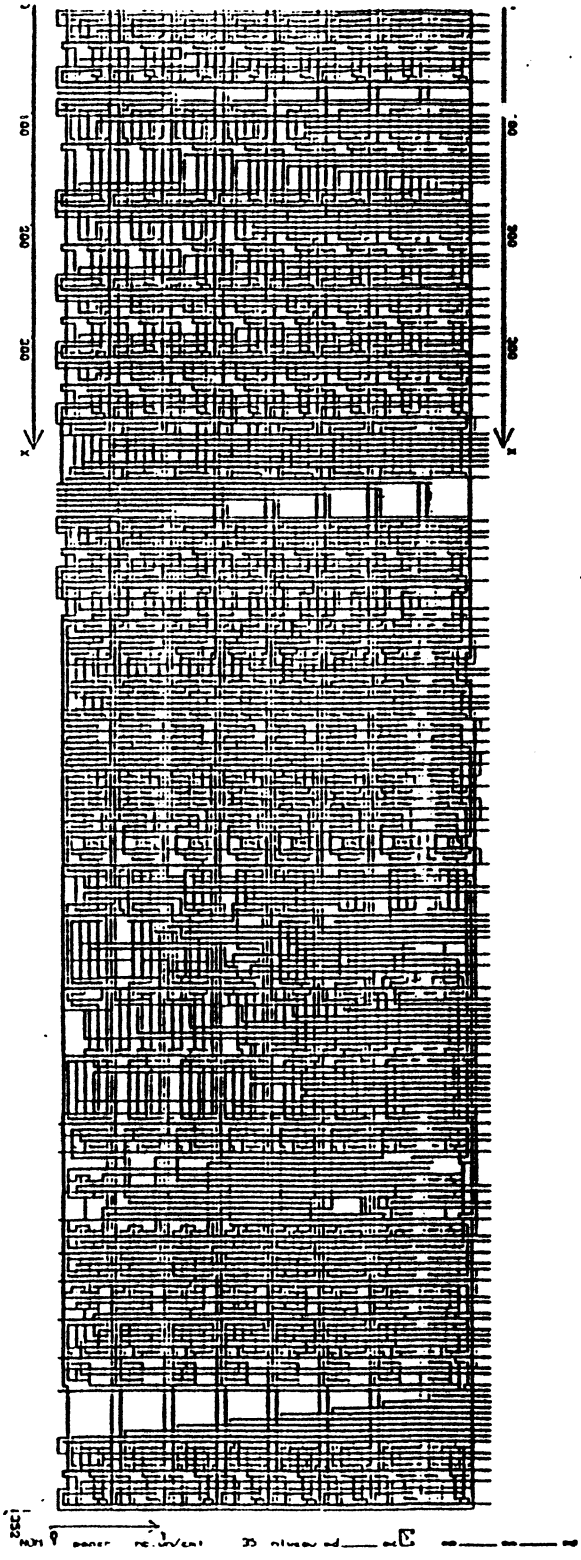
fig:III.78 Densités relatives d'intégrations obtenues suivant les méthodes d'implantations utilisées.

Cette perte de place relativement faible se justifie amplement par la fiabilité et le gain de temps qu'apporte l'introduction de cette étape intermédiaire d'implantation.

La réalisation complète de la partie opérative se fait par assemblage des briques à l'aide d'un logiciel de routage et de tassement tel que "LUBRICK" (cf annexe 9).

Cette implantation "stick" permet d'évaluer la surface du chemin de données à 0.552 mm carré (1352 x 409 microns).

fig:III.79 Représentation de la partie opérative en symbolisme "stick"



Sabot le 09/05/84 a 15:47:6 dessin no 1

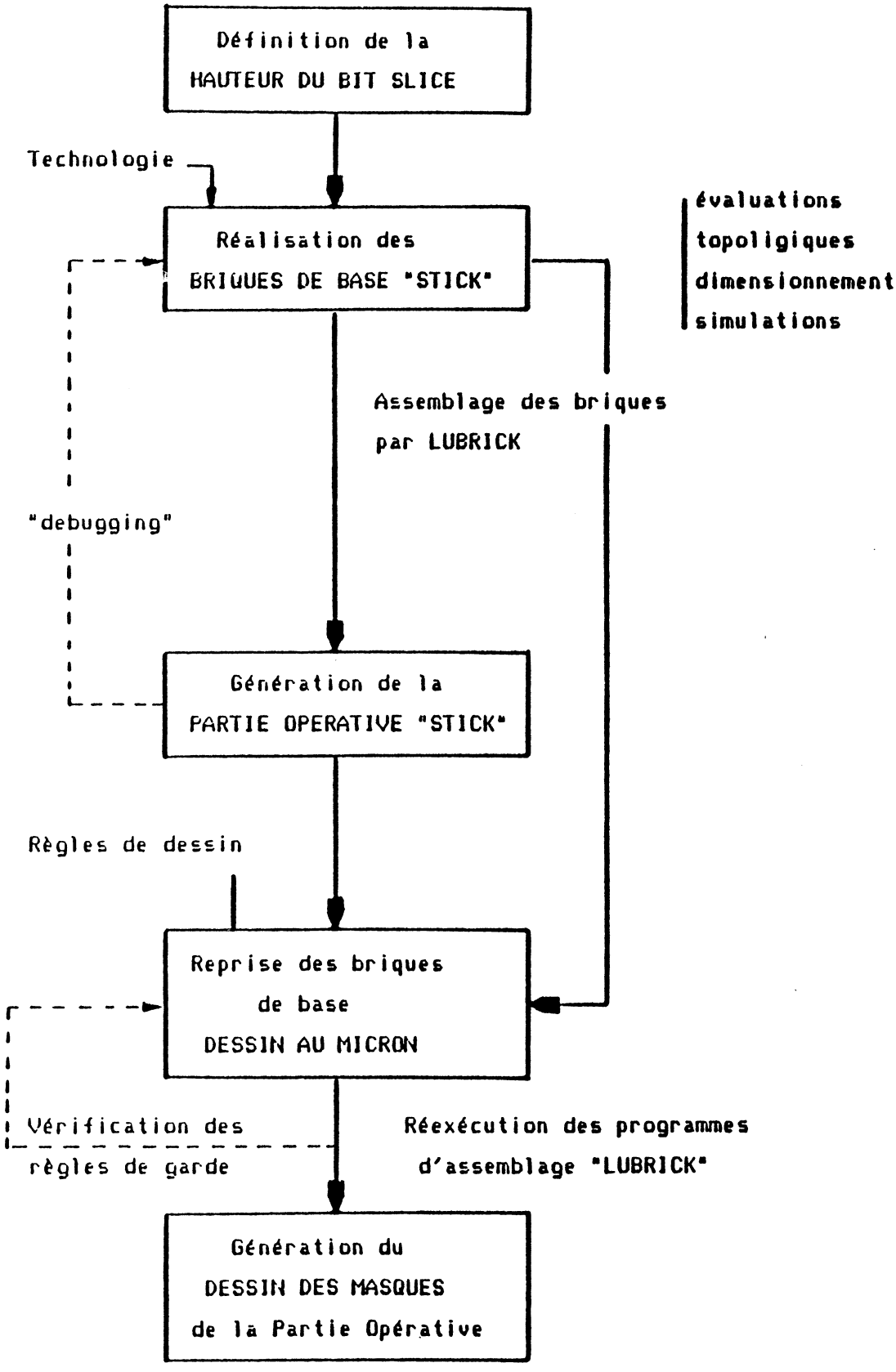


fig:III.80 Méthode d'implantation utilisant un symbolisme "stick"

b) Génération du dessin des masques de la partie opérative

On procède au dessin au micron des briques de base implantées en "stick" par application des règles technologiques (fig III.80); ces règles sont définies selon un paramètre "LAMBDA" déterminé par la technologie utilisée (ANC81,MEA).

Moins précises (donc plus coûteuses en surface) mais de formulation plus simple, leur emploi se justifie par la facilité qu'elles procurent durant la phase d'implantation, ceci sans une détérioration notable de la densité d'intégration. De surcroît, une description paramétrée permet d'assurer un suivie de la production tout en bénéficiant de l'évolution de la technologie, sans avoir à modifier l'implantation.

La partie opérative du 80C48 a été réalisée en technologie CMOS du "CMP" (Circuits Multi-Projets). Cette technologie permet de disposer d'un niveau de polysilicium et d'aluminium. Les caissons sont de type "P". Seul les contacts alu/diffusion ou alu/polysilicium sont autorisés moyennant quoi les connexions poly/diffusion sont réalisées par un pont d'aluminium.

On réalise le passage du dessin en "stick" au dessin au micron par "dilatation" des connexions de polysilium, diffusion et aluminium. L'implantation se fait de façon manuelle sur une grille dont les pas sont établis après avoir dessiné des briques considérées comme critiques (UAL, registres tampon d'UAL, registres d'états); ces études permettent de choisir un pas de sept lambdas (4 pour les connexions et 3 pour un inter-alu) et cinq pour le polysilicium (2 pour les connexions et 3 pour un inter-poly).

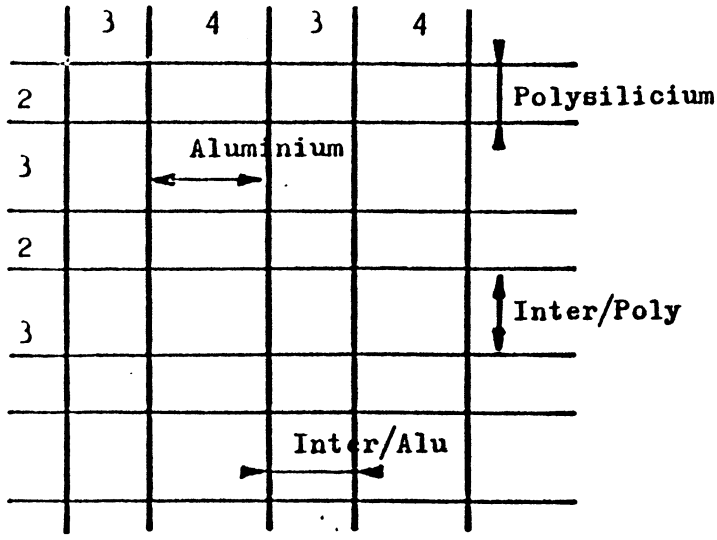
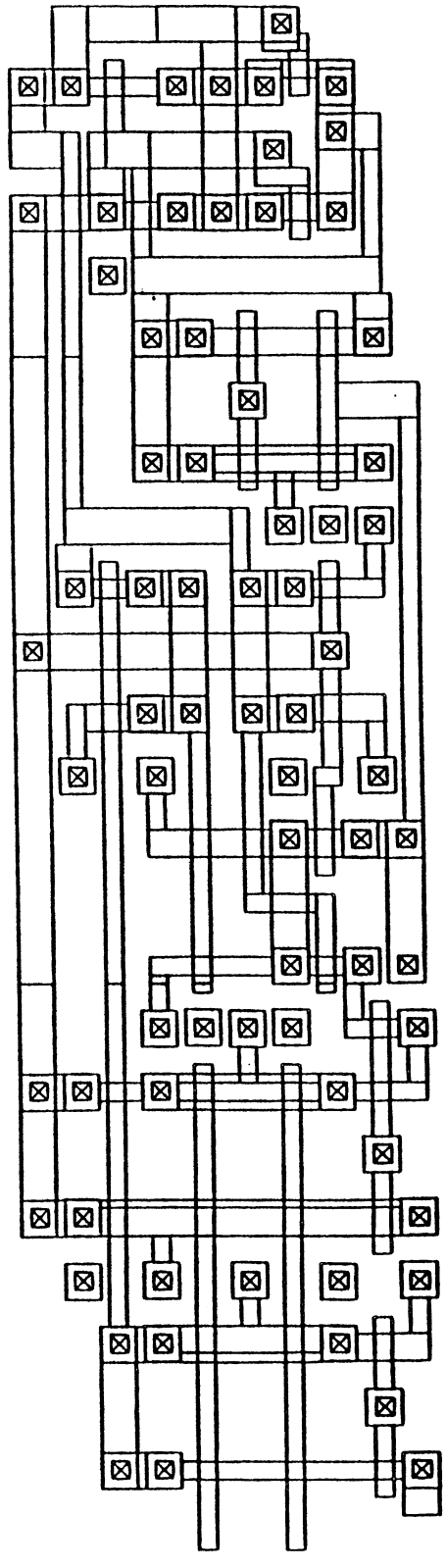


fig:III.81 Grille d'implantation pour dessin au micron

La hauteur du "bit slice" est choisie avec précaution pour offrir une implantation optimale en terme de densité; sa valeur est déterminée par la porte la plus complexe de l'UAL. Ainsi, toutes les briques de base sont implantées sur une hauteur de douze pas de polysilicium.

Les canaux de routage (bus "alpha" et "bêta") sont élargis au maximun en tenant compte du fait qu'ils sont longs et que l'augmentation de la capacité est largement compensée par la diminution de la résistance, et donc du temps de réponse. En technologie CMOS, les réseaux logiques sont réalisés avec des transistors de tailles minimales; on dimensionne les portes de puissance - attaques de bus, entrées/sorties - après simulation électrique (cf annexe 5).

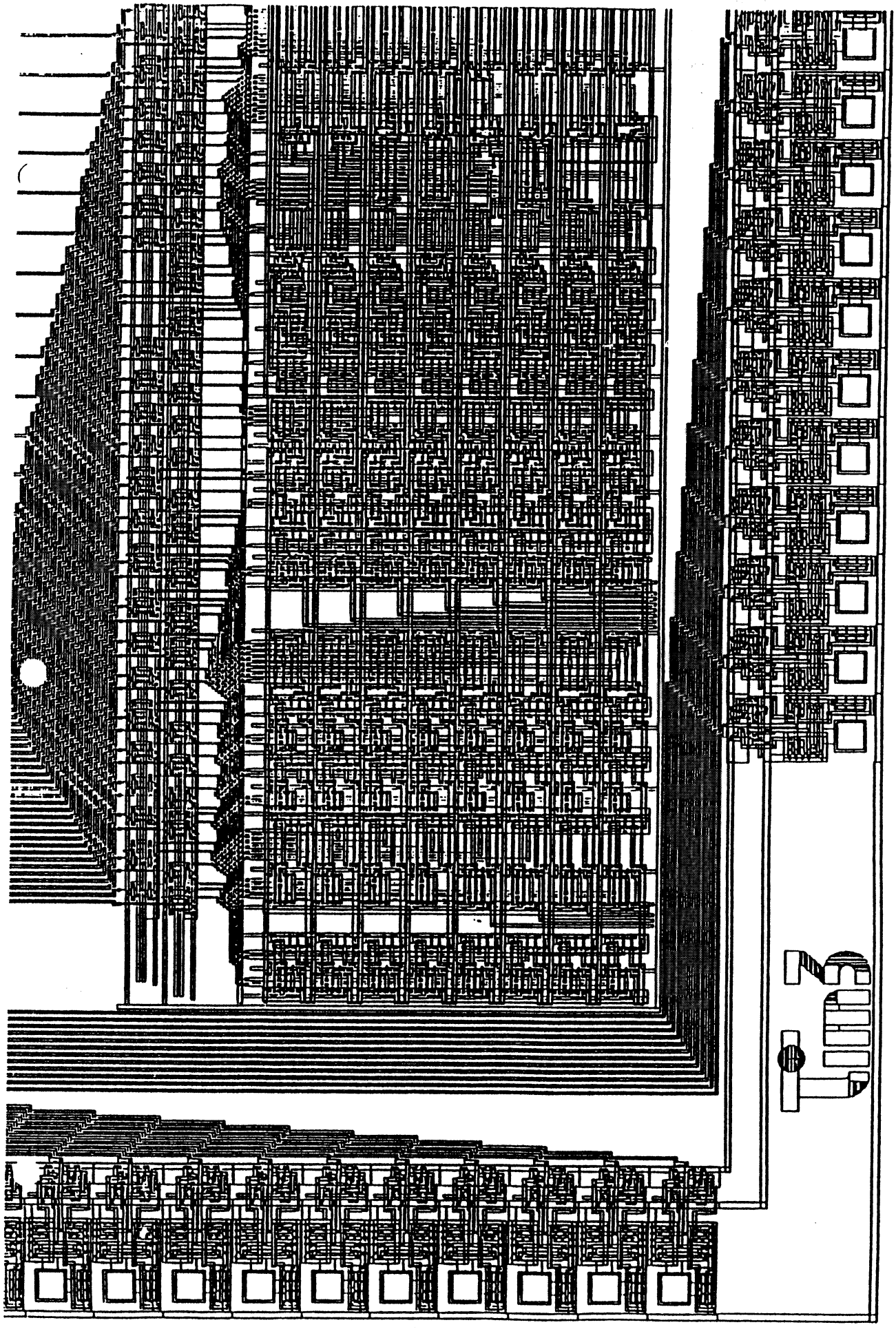
fig:III.82 Représentation au micron de l'UAL

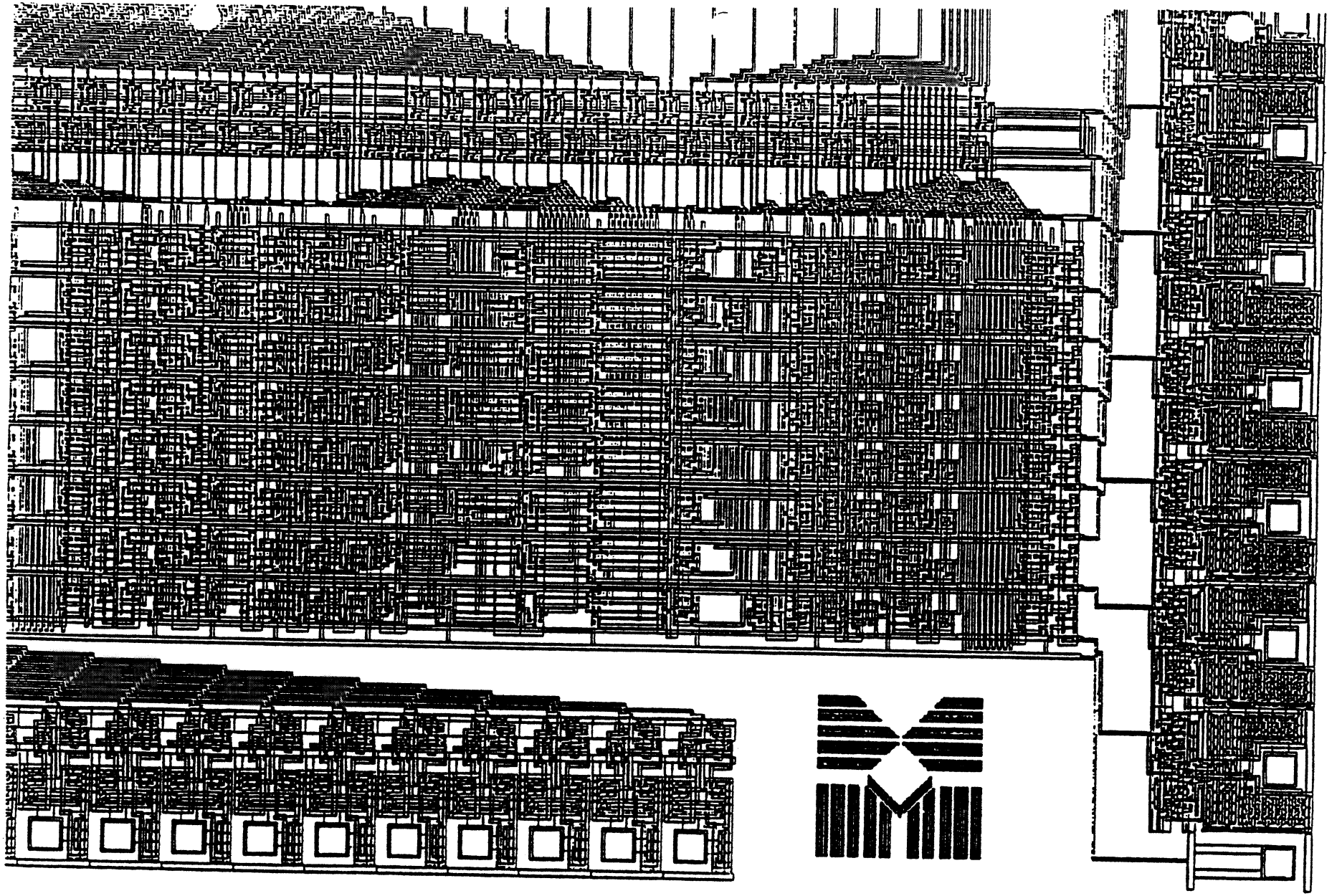


Les briques de base ainsi dessinées sont digitalisées; le programme de digitalisation génère un fichier "LUCIE" (cf annexe 6) dont la traduction permet de visualiser le dessin sur console graphique.

Après vérification de la conformité avec les règles de garde, l'exécution "LUBRICK" de la description assemble la partie opérative qui exécute le jeu d'instruction du 80C48.

fig:III.83 Implantation au micron de la partie opérative
qui exécute le jeu d'instructions du 80C48





CMP 84

1000 1200 1400 1600 1800

On remarque la grande régularité de la structure ainsi établie. La surface globale de la partie opérative du 80C48 est estimée à 0.963 mm carré avec une technologie de 2.5 microns (1800 x 535 microns) pour un total de 1888 transistors, soit une densité de 1960 transistors au mm carré.

La différence de valeur avec l'évaluation de surface précédente se justifie par le fait que l'on n'a pas conservé le pas de grille de l'implantation "stick" pour respecter les règles de dessin au "LAMBDA".

La conception de circuits intégrés VLSI requiert:

- une bonne méthodologie de travail,
- une bonne approche topologique,
- des outils informatiques appropriés,

afin d'obtenir un dessin des masques dans des temps acceptables.

L'utilisation de la méthode "bit slice" permet d'obtenir une implantation régulière pour la partie opérative. Une telle structure se prête à l'utilisation d'outils de CAO, source de fiabilité et de rapidité; la difficulté à implanter des circuits de plus en plus complexes ne peut être contournée qu'en allant dans ce sens.(ANC82)

Des programmes de routage permettent d'assembler des briques tout en minimisant le surface de silicium (cf annexe 9). De même, on est passé de la représentation "stick" au dessin au micron en redessinant les cellules de bases; une telle étape ouvre la voie à l'automatisation, après définition du pas d'implantation.

La vérification du dessin des masques ainsi généré est réalisée grâce à des outils de mise au point tel que "COMFOR" (extraction de schéma électrique) ou "VERDICT" (vérification de règles de dessin); partant tous deux de fichiers source "LUCIE", ils permettent, d'une part, de vérifier la fonction logique réalisée, d'autre part, de certifier le respect des contraintes technologiques. (cf annnexe 8 & 9)

Le premier prototype obtenu (juillet 84) a fait l'objet de tests fonctionnels sous microscope électronique; pour ce faire, le chemin de données et les lignes de contrôle sont directement activés, après amplification, à des plots d'entrées/sorties.(MIC83)

2-7 Plan de masse

L'étude topologique du circuit est réalisée en parallèle avec sa conception logique et non comme une étape finale; on étudie, dès que possible, une fois définies les spécifications fonctionnelles, la structure du chemin de données et l'emplacement relatif des différents blocs.

Le plan de masse évolue parallèlement avec la finesse de la description; pour être significatif, il doit être accompagné de valeurs métriques afin de permettre l'évaluation de l'encombrement, de l'imbrication et de la connexion des différentes unités.

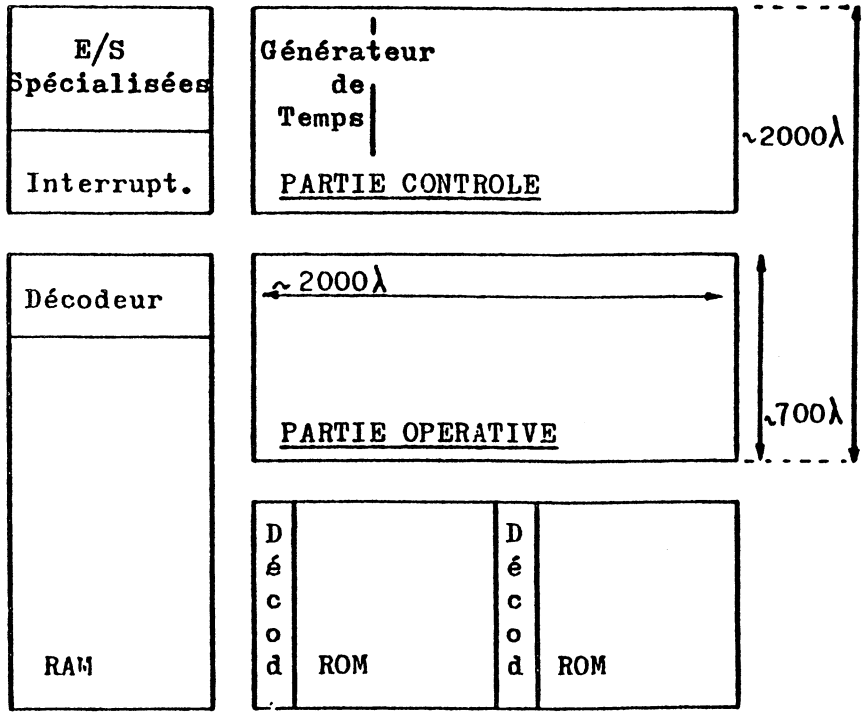
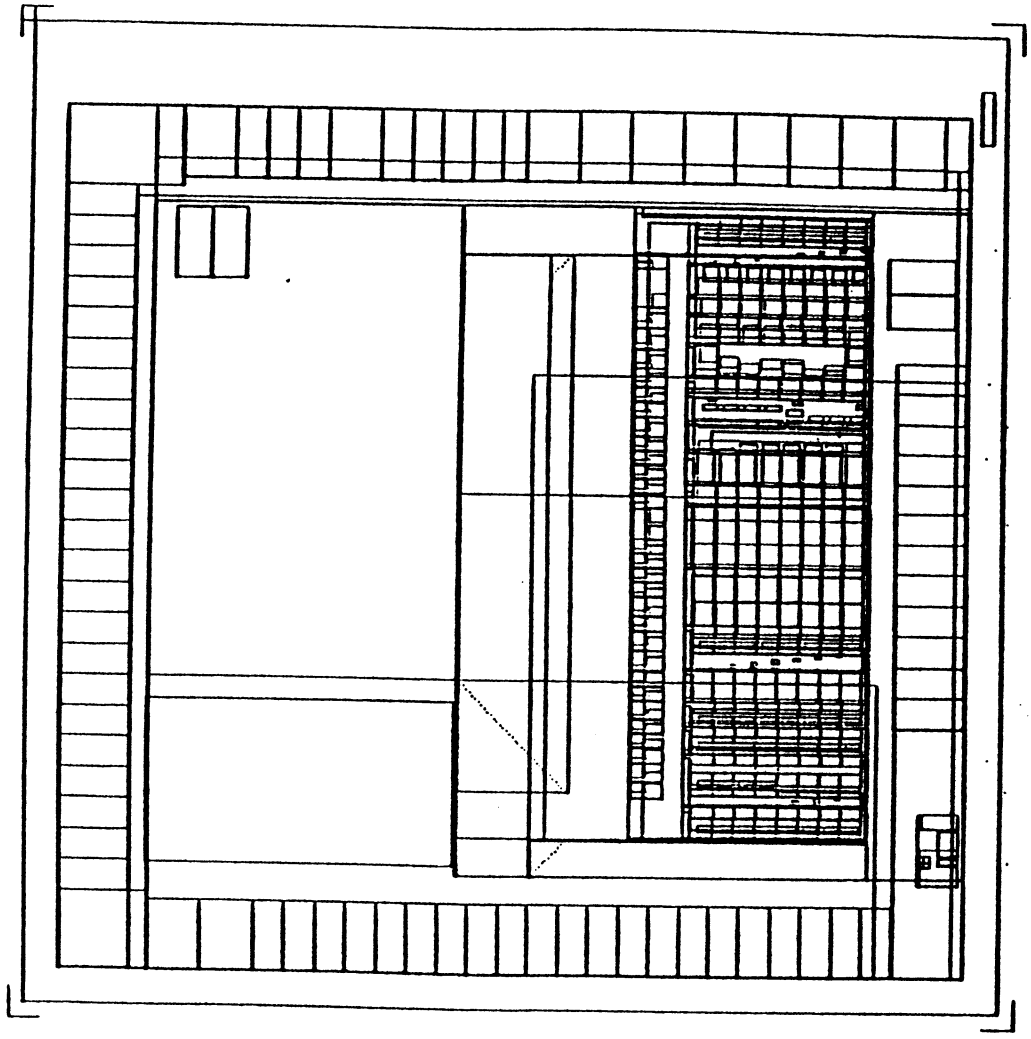


fig:III.84 Organisation du 80C48

On remarque l'agencement des registres "W", "PCL /PCH" de la partie opérative, mis en correspondance avec les décodeurs de la mémoire vive et de la mémoire morte, respectivement.



PLAN DE MASSE DU CIRCUIT

III 3 Réalisation de l'unité de contrôle et de séquencement

3-1 Choix d'une organisation

a) Généralités:

L'implémentation de la partie opérative nous permet de faire un choix fonctionnel et topologique de la partie contrôle.

Pour un microprocesseur de la complexité du 80C48, la réalisation de la partie contrôle avec un seul PLA n'est pas acceptable car la surface du circuit devient trop importante. La séparation partielle des deux fonctions - séquencement et génération de commandes - améliore les caractéristiques de la réalisation.(OBR82)

Le déroulement d'une instruction est décrit par deux ensembles: d'un côté, l'ensemble des commandes à activer pour cette instruction, de l'autre côté, l'ensemble des instants de leur activation. Il en découle que pour réaliser l'algorithme d'interprétation, il faut avoir: un dispositif qui élabore les commandes à activer, un autre qui délivre les instants définissant le temps et un troisième qui validera les commandes au bon moment. (fig III.85)

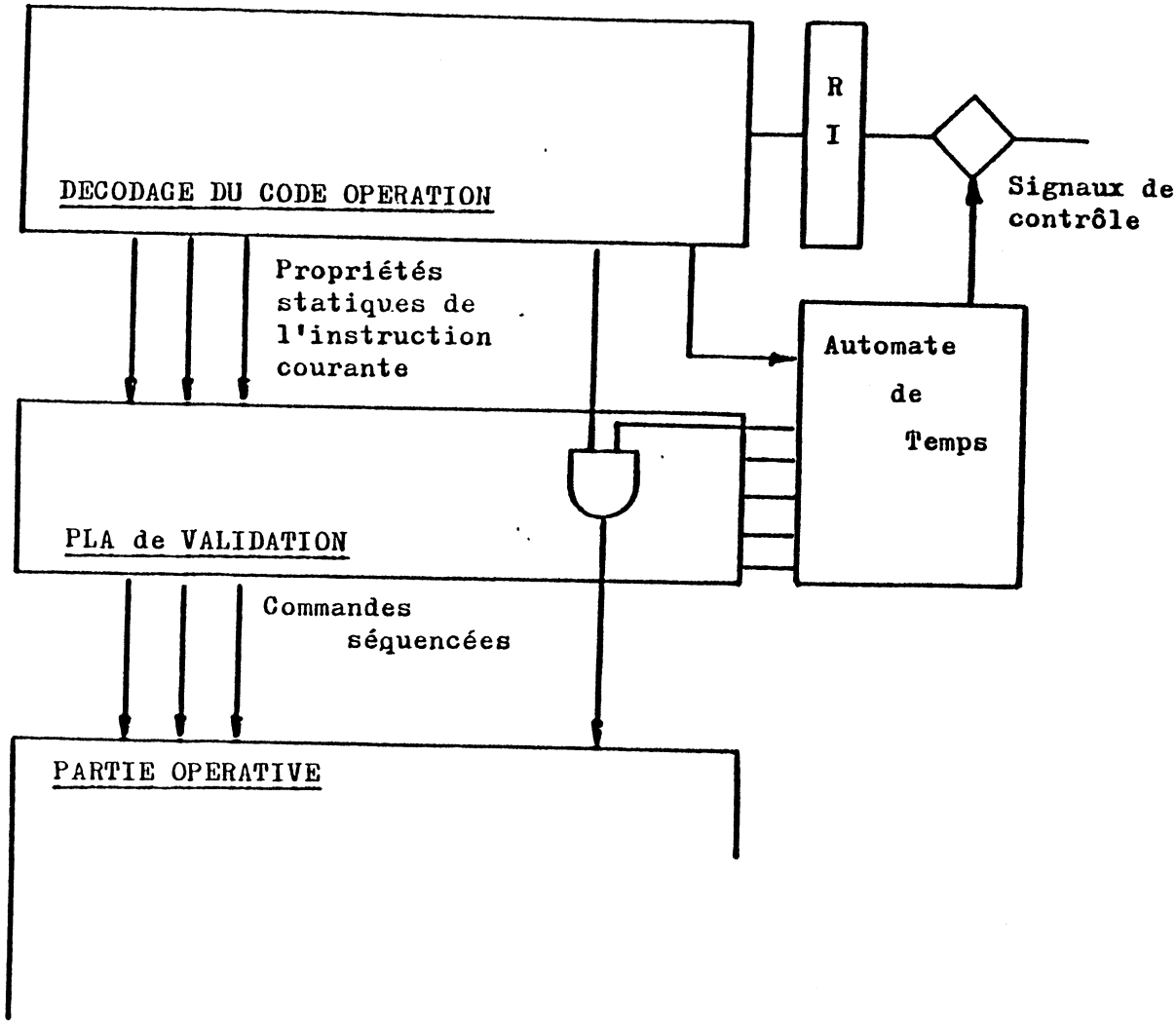


fig:III.85 Partie contrôle utilisant un générateur de temps

b) Décodage du code opération:

On se propose d'étudier une partie contrôle dans laquelle la fonction de décodage est réalisée à l'aide de plusieurs PLAs:

- le PLA de propriétés
- le PLA de génération
- le PLA de paramétrage

(fig III.86)

b1) le PLA de propriétés:

Dans cette solution, l'idée principale consiste à diminuer le nombre d'états de l'algorithme grâce au conditionnement de la génération des

commandes; on utilise des informations caractéristiques, extraites directement du code opération et invariantes tout au long de l'interprétation d'une instruction.

L'algorithme d'interprétation est analysé sous l'angle de la recherche des états semblables. On comprend, par semblables, les états qui génèrent les même commandes.

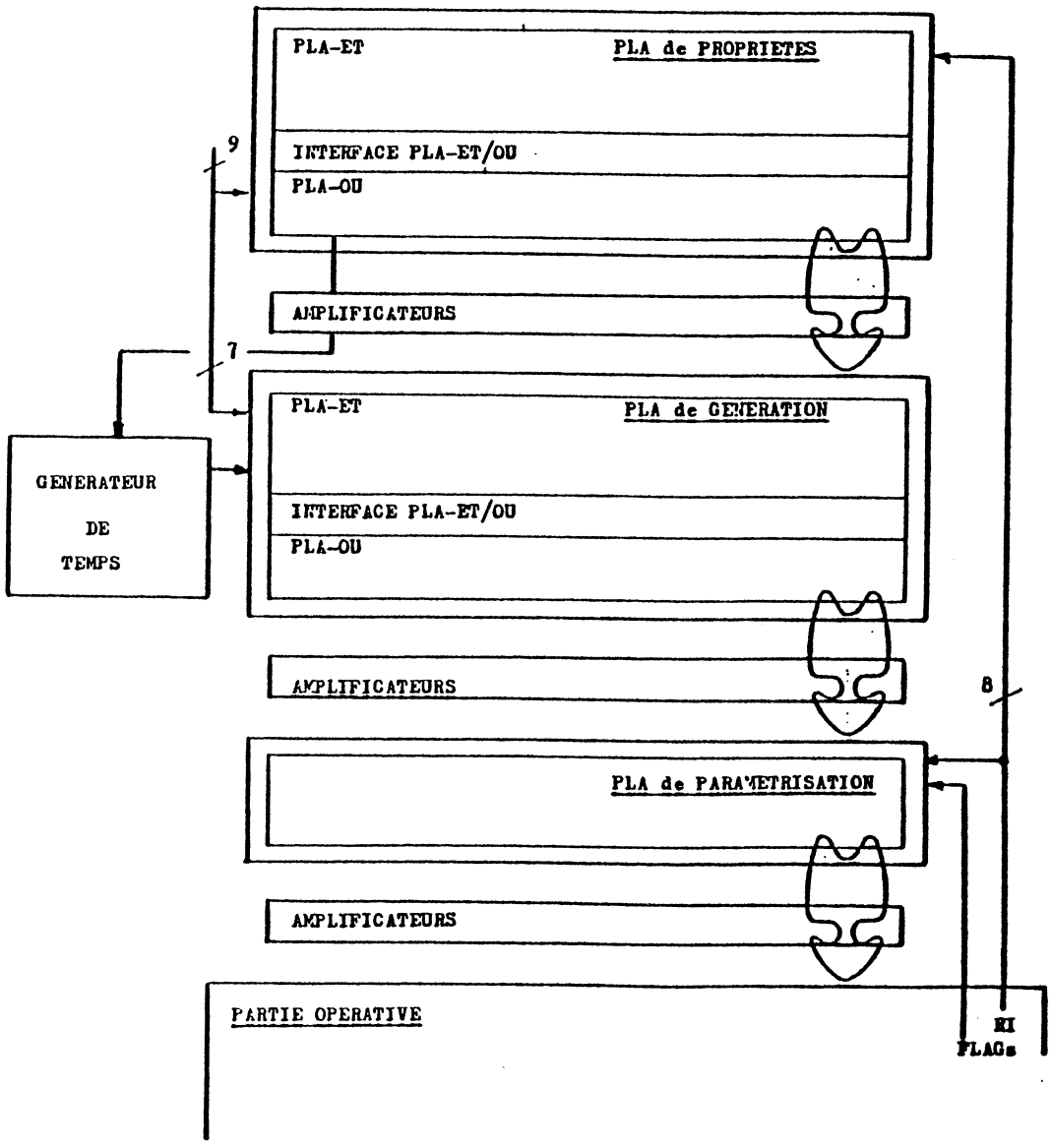
Ainsi, toutes les instructions pour lesquelles la commande "Ci" est activée à l'instant "Tj" possèdent la propriété "Pij".(SCH85)

Le rôle du PLA de propriété est de décoder le code opération de l'instruction pour en extraire les propriétés.

Le but recherché - le plus petit nombre de propriétés englobant le plus grand nombre d'actions - est atteint en séquençant de façon systématique l'algorithme d'interprétation.

Une structure de contrôle avec générateur de temps impose des actions systématiques durant les phases de décodage du code opération; en conséquence, les étapes "S1" et "S2" de l'algorithme d'interprétation sont identiques pour toutes les instructions, la paramétrisation n'intervenant qu'à partir de l'état "S3" (cf III 2-3 b3).

fig:III.86 Schéma fonctionnel de la partie contrôle multi-PLAs



b2) Le PLA de paramètres:

Les paramètres concernent un ensemble de commandes agissant sur des unités fonctionnelles de la partie opérative durant l'interprétation d'une instruction donnée.

Cet ensemble de commandes peut être généré statiquement à partir du code opération et de l'état de certaines bascules distribuées dans la partie opérative, et ce, dès la lecture de l'instruction.

Le transfert conditionnel "bus:= if COND then A else B" signifie que l'on affecte au bus la variable "A" lorsque la condition "COND" est vraie, et "B" dans le cas contraire.

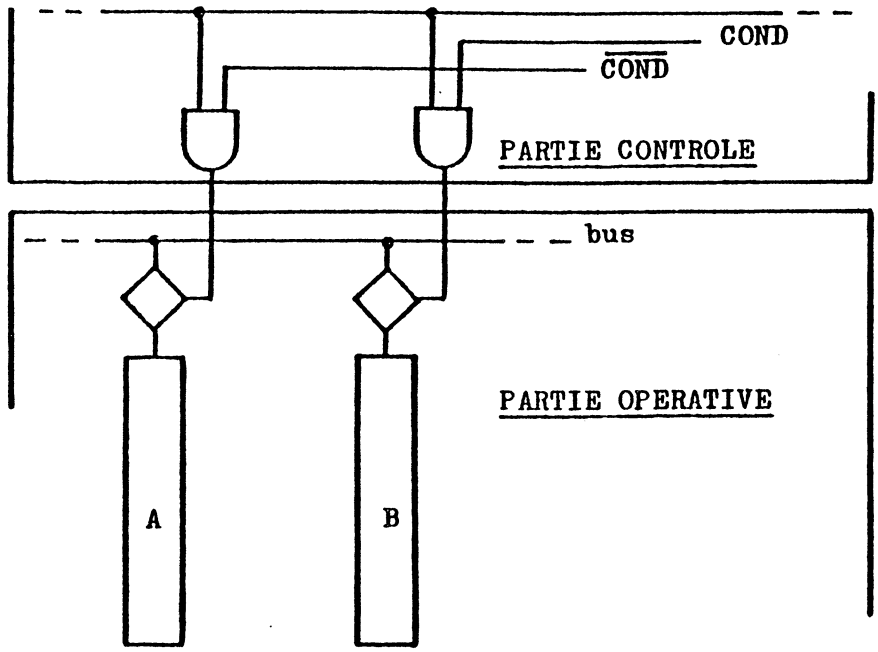


fig:III.87 Réalisation matérielle d'un transfert conditionné à l'état d'une bascule "COND"

L'utilisation des PLAs de paramètres permet de diminuer le nombre de signaux que doit générer le PLA de commandes, et également de diminuer le nombre des étapes de séquencement.

c) Le séquencement des instructions:

c1) le générateur de temps:

Le générateur de temps est un automate qui assure la progression de l'interprétation de l'instruction ainsi que l'identification de chaque instant "Ti" d'exécution grâce aux signaux de séquencement qu'il génère.

Pour cela, il prend en compte non seulement les caractéristiques de l'instruction exécutée (10 ou 20 phases) mais aussi les signaux de contrôle et l'état de la machine (interruptions...etc).

On l'implémente sous la forme d'un compteur "Jonhson" constitué de trois bascules maître/esclave rebouclées par un inverseur (fig III.88) et générant les six états suivants: 000/ 100/ 110/ 111/ 011/ 001. Le cycle machine du 80C48 ne nécessite que cinq états; un circuit combinatoire simple assure la remise à zéro du compteur suite à l'activation du cinquième état.

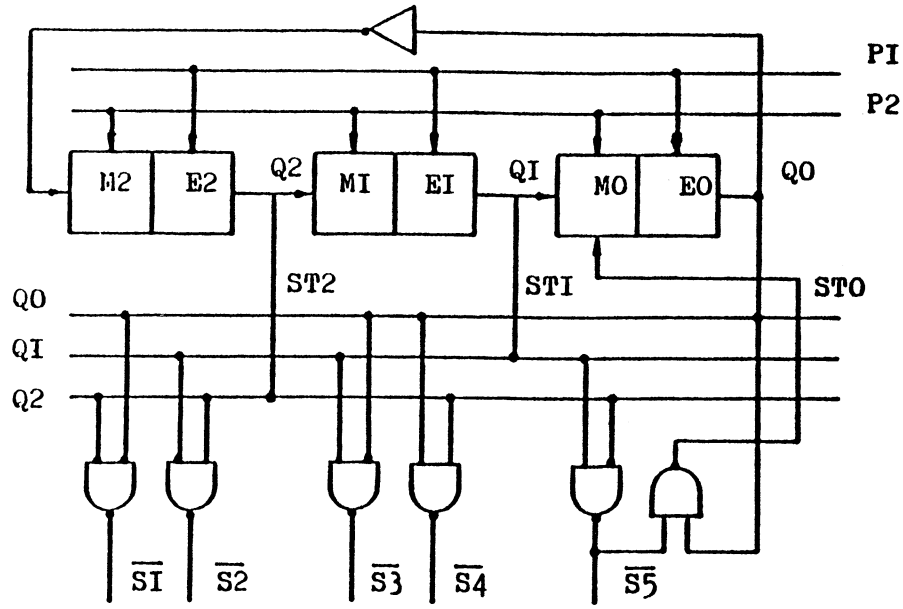


fig:III.88 Le générateur de temps (compteur "Jonhson")
Logique combinatoire de décodage des états

Un réseau logique à base de portes "NAND" décode l'état des bascules et génère les cinq signaux de temps qui composent le cycle machine du 80C48.

ETATS	ST2	ST1	ST0	EQUATIONS
S1	0	0	0	(ST2*.ST0*)*
S2	1	0	0	(ST2.ST1*)*
S3	1	1	0	(ST1.ST0*)*
S4	1	1	1	(ST0.ST2)*
S5	0	1	1	(ST2.ST1)*

fig: III.89 Equations logiques des états

On réalise une bascule maître/esclave par la mise en série de deux points mémoires (fig III.90); le signal de contrôle "Reset" réinitialise l'élément maître. La synchronisation est assurée sur les phases "P1" - bascule maître - et "P2" - bascule esclave -.

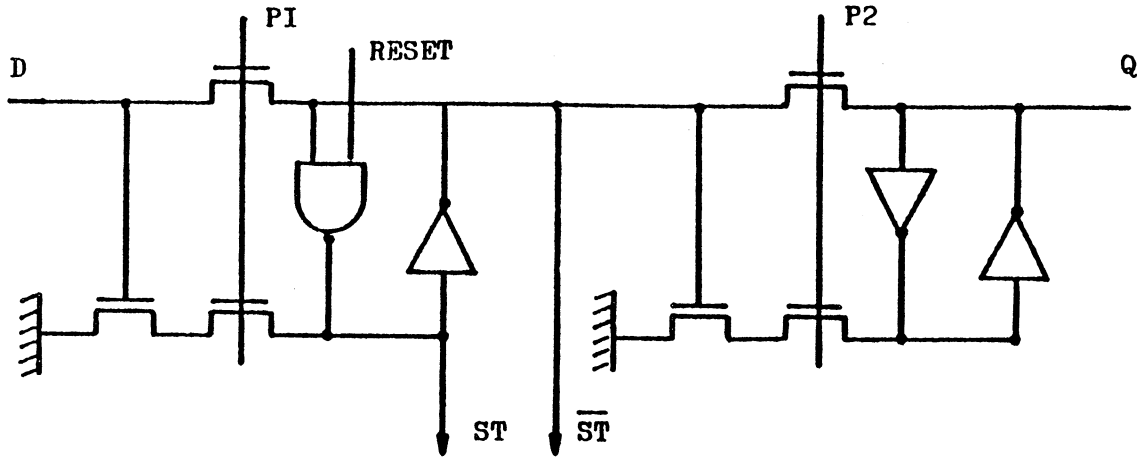


fig:III.90 Réalisation d'une bascule maître/esclave

c2) le PLA de génération de commandes:

Les commandes sont conditionnées par les propriétés. Le PLA de génération reconnait l'état de l'algorithme - phase dans le cycle - et les propriétés, permettant ainsi d'obtenir des équations sous la forme de combinaisons temps/ fonctions:

$$Cj = \sum (Ti \times Pk)$$

"Cj": commande
"Ti": temps
"Pk": propriété

Le signaux ainsi générés relèvent de l'une des trois catégories suivantes:

- les commandes qui agissent directement sur la partie opérative, telles les connexions bus/registre ..etc. Elles sont transparentes vis à vis du PLA de paramètres.

- les commandes conditionnelles qui nécessitent une paramétrisation (cf III 3-1); elles sont explicités sous la forme:

$$Cj = \sum (Ti \times Pk \times CONDp)$$

pour laquelle "CONDp" signifie une condition déterminée à partir du code instruction courant ou une variable d'état.

- les commandes qui activent plusieurs ressources contenues dans la partie opérative (contrôle de l'UAL, de l'opérateur de décalage..etc); paramétrisation et décodage sont alors parfois confondus en un seul PLA.

Le fait d'avoir utilisé un PLA de paramètres a permis de diminuer considérablement le nombre de commandes produites par la matrice "OU" du PLA de génération; de plus, cette matrice est facilement optimisable.

c3) les interruptions

Le 80C48 ne présente qu'un seul niveau d'interruption; durant le traitement d'une interruption, toute nouvelle demande sera ignorée et mémorisée grâce à des bascules implantées dans la partie contrôle. (fig III.92) Le retour au programme principal se fait après exécution d'une instruction "RETR".

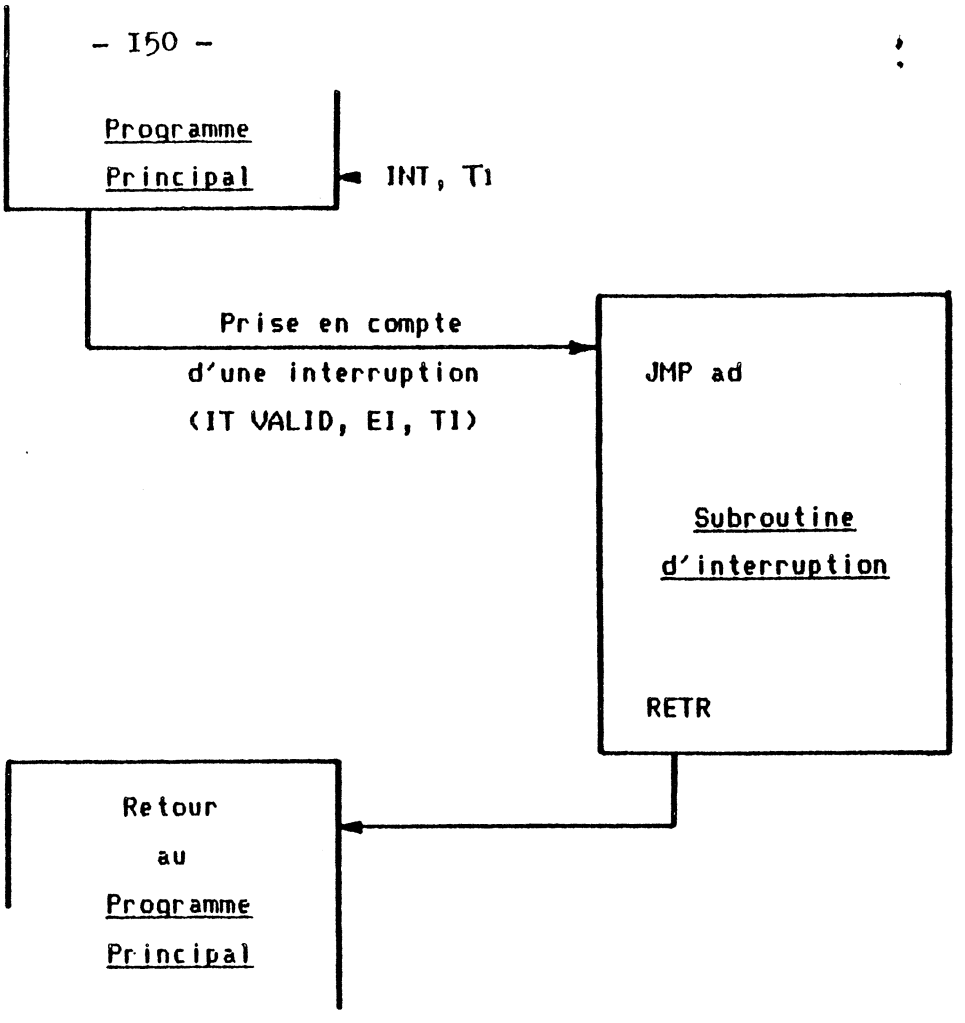


fig:III.92 Déroutement d'un programme suite à l'activation d'une ligne d'interruption

On échantillonne les lignes d'interruption - "INT", "T1", "Reset" - à chaque cycle machine durant la phase active de "ALE" (dernier cycle de l'instruction courante pour les instructions doubles). Une fois détectée, l'interruption est traitée par un algorithme spécifique "IPO" ("Interrupt after Power On") dont le rôle principal est de sauvegarder l'état courant de la machine - compteur de programme, partie poids forts du registre d'état "PSW"- et de forcer l'accès au mot "V0" (Reset), "V3" (interruptions externes) ou "V7" (interruptions internes) de la mémoire morte. Une instruction de saut inconditionnel adresse alors le sous-programme relatif au traitement de l'interruption courante.(fig III.93)

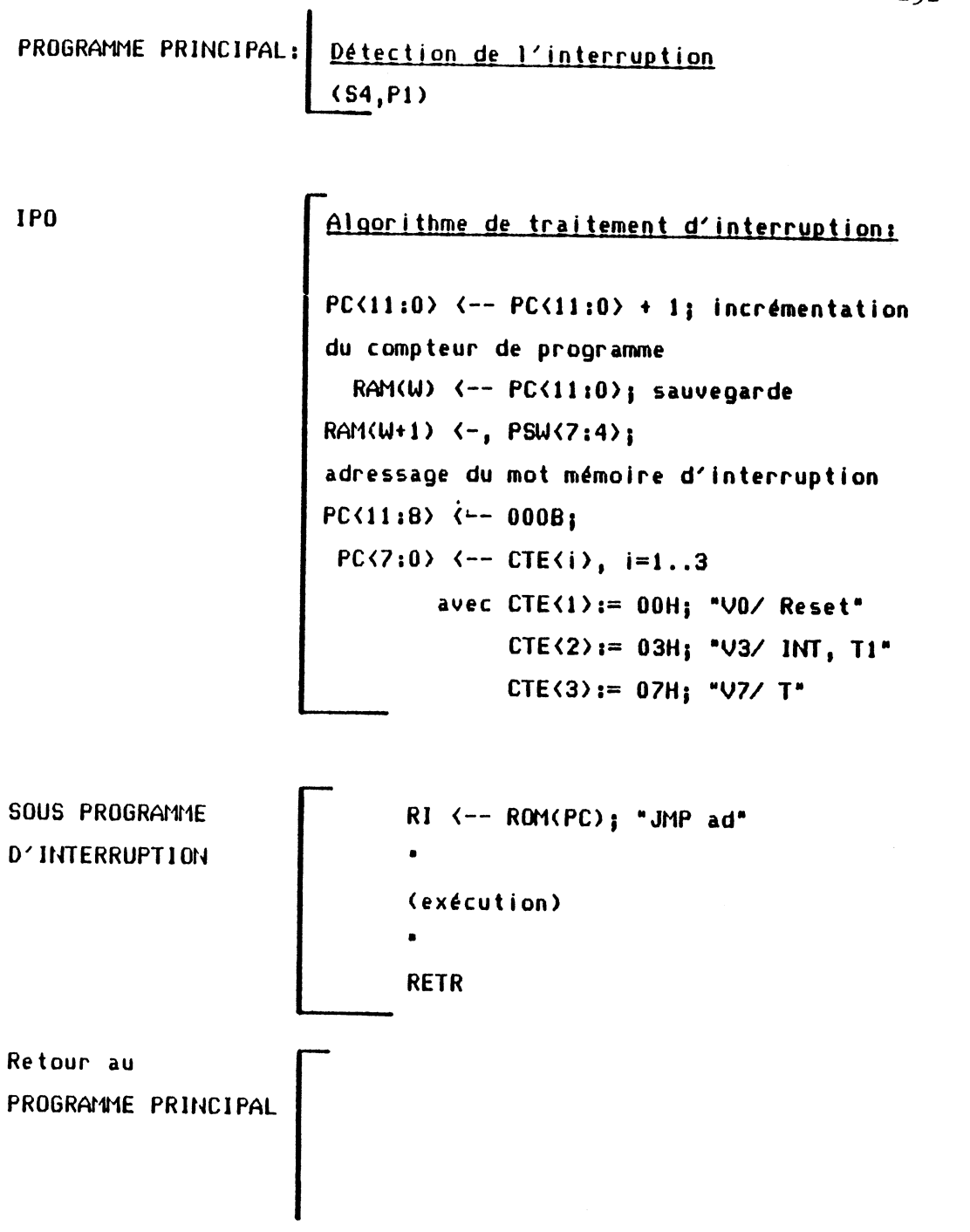


fig:III.93 Prise en compte et traitement d'une interruption

Remarque: on note la très grande analogie de la procédure avec le déroulement de l'instruction d'appel de sous-programme "CALL".(APL82)

Un réseau de portes logiques (fig III.94) valide une requête d'interruption et active la routine correspondante.

a) les interruptions d'origine externe:

générées à partir des broches "INT" ou "RESET", elles sont prioritaires par rapport à une requête d'origine interne.

La ligne "INT" (INTerrupt) est échantillonnée pendant la phase active de "ALE" durant le dernier cycle de l'instruction courante; sa valeur est mémorisée par la bascule "INT RQ" (INTerupt ReQuest). La prise en compte est conditionnée à l'exécution de l'instruction "EN I" (ENable Interrupt) qui active la bascule "INT EN" (INTerrupt ENable).

Toute requête d'interruption externe sera sans effet si elle fait suite à l'exécution de l'instruction "DIS I" (DISable Interrupt) ou à l'activation de la ligne "RESET".

b) les interuptions d'origine interne:

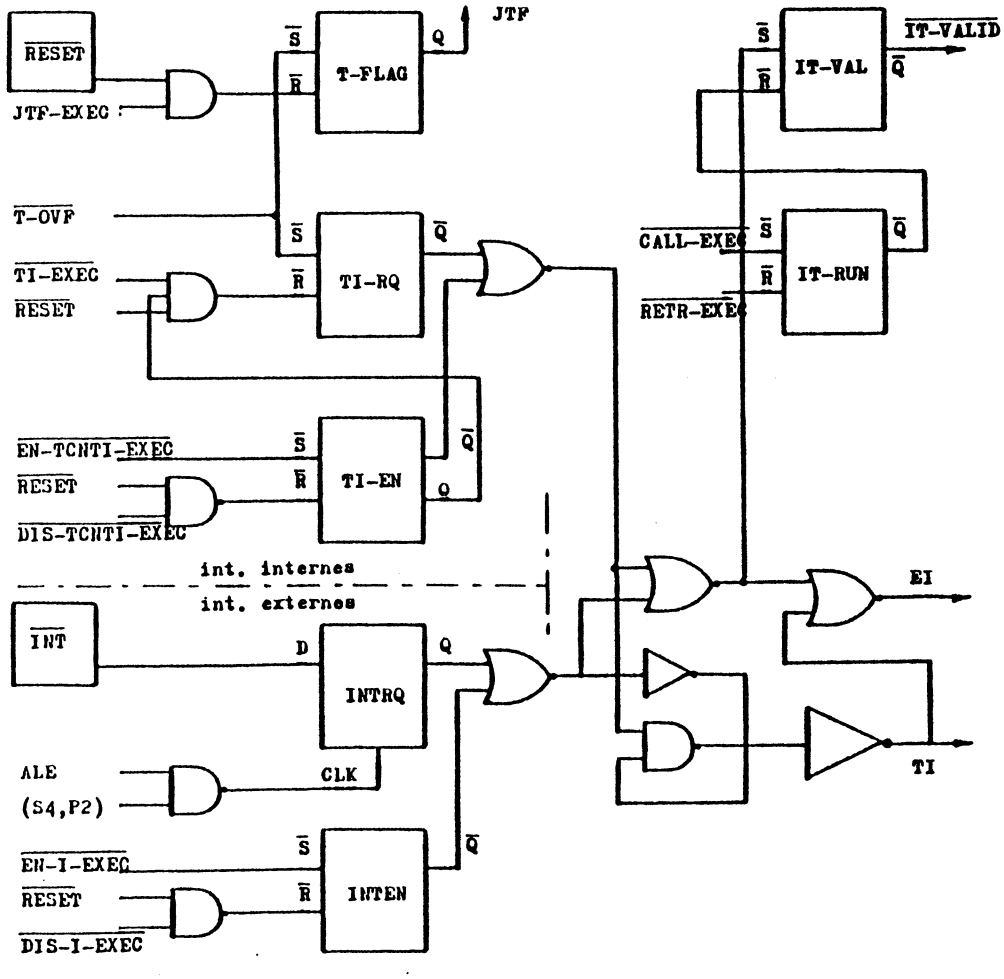
elles font suite au débordement du compteur/temporisateur "T OVF" (Timer OVerFlow) dont l'état est mémorisé dans les bascules "TI RQ" (Timer Interrupt ReQuest) et "T FLAG" (Timer FLAG). Leur prise en compte est conditionnée par la bascule "TI EN" (Timer Interrupt ENable) activée suite à l'exécution de l'instruction "EN TCNTI" (ENable Timer CouNTer Interrupt).

On réinitialise la bascule "TI RQ" par activation de la ligne "RESET" ou suite à l'exécution de l'instruction "DIS TCNTI" (DISable Timer CouNTer Interrupt).

La bascule "T FLAG" conditionne la branchement durant l'exécution de l'instruction de branchement "JTF" (Jump on Timer Flag).

Un réseau de portes logiques réalise un circuit de priorité qui délivre les deux commandes "EI" (External Interrupt) et "TI" (Internal Interrupt); celles-ci sont exclusives et ne peuvent donc être au niveau haut simultanément. Une bascule "IT VAL" (Interupt VALid) indique que l'on est en attente d'interruption; elle sera prise en compte dès la fin de l'instruction courante.
Un appel à la procédure d'interruption la réinitialise.

fig:III.94 Logique "anarchique" de traitement des interruptions



Le compteur/temporisateur peut aussi travailler en compteur d'évènement externe lorqu'il est connecté à la broche "T1" (cf III 1-1j). Il se produira une incrémentation une fois tous les trois cycles au maximun sur une transition du niveau haut au niveau bas de la broche d'entrée (fig III.94a).

Lorsqu'il fonctionne en temporisateur, on réalise une incrémdntation tous les 32 cycles machine grâce à un compteur modulo 32.

Le réseau logique suivant assure la commutation entre ces deux modes de fonctionnement - la sélection est conditionnée par l'exécution de l'une des instructions "STRT T" (STaRT Timer) ou "STRT CNT" (STaRT CouNTer);

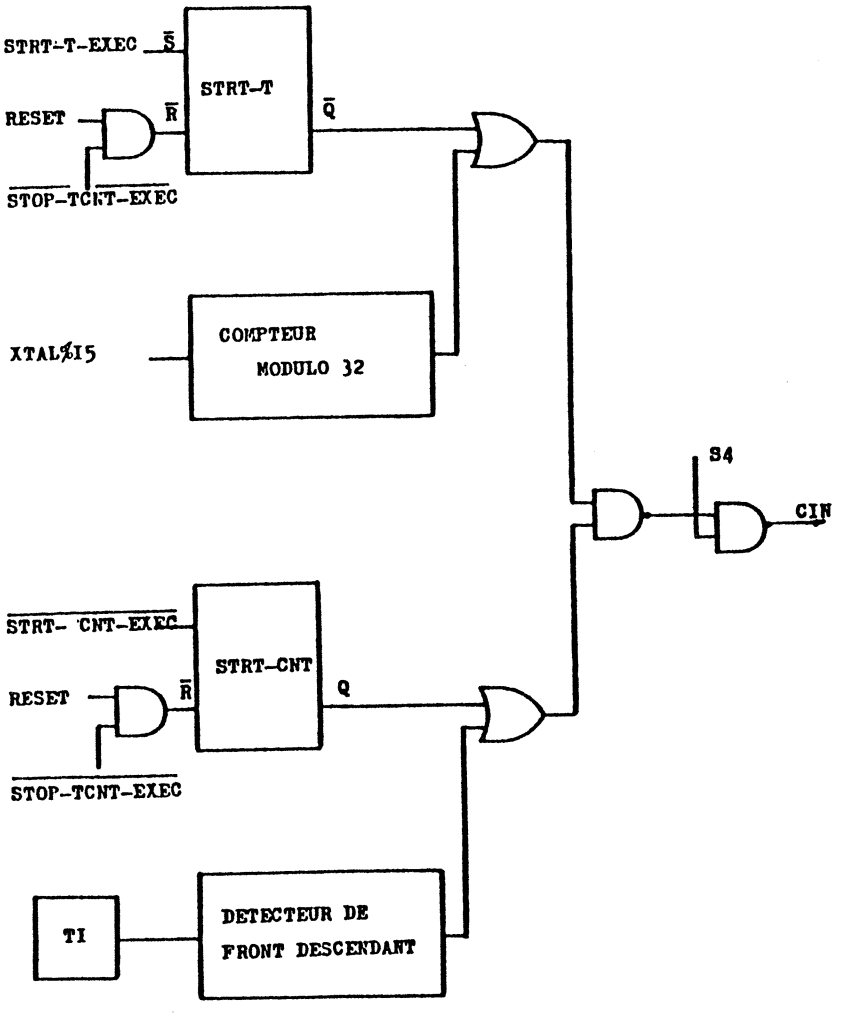


fig:III.94a Logique de génération de la retenue entrante pour l'incrémentation du compteur/temporisateur

désactivation par l'instruction "STOP TCNTI" (STOP Timer CouNTer Interrupt) ou par l'activation de ligne "RESET" - et génère la retenue entrante "CIN" qui assure l'incrémentation du regitre "T" (cf III 2-3b):

(S4,P1) U1 <--a:=T;
(S4,P2) T <--b:= U1 + CIN; avec CIN = (0,1)B

On réalise de façon simple un détecteur de front descendant par une bascule maître-esclave associée à une porte inverseur et un NOR (fig III.94b)

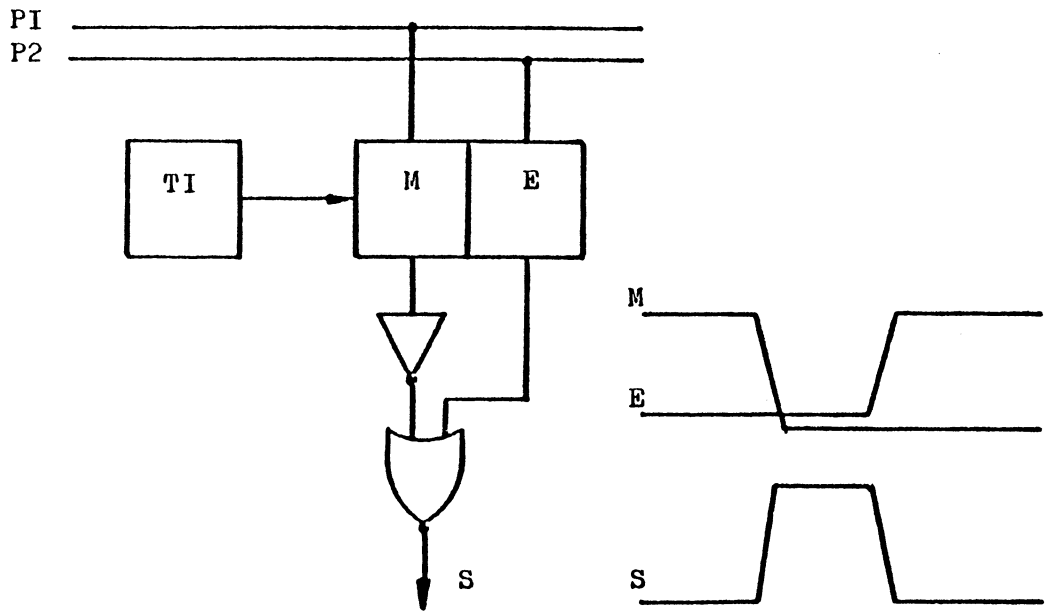


fig:III.94b Réalisation d'un détecteur de front descendant

3-2 Génération des PLAs

On procède à la génération du contenu des PLAs de la partie contrôle à l'aide des logiciels d'analyse, d'expansion et de synthèse de la description "MACSIM"; ces trois programmes correspondent aux trois phases de traitement de l'algorithme d'interprétation et communiquent entre eux par l'intermédiaire de fichiers.(SIM83, OBJ84)

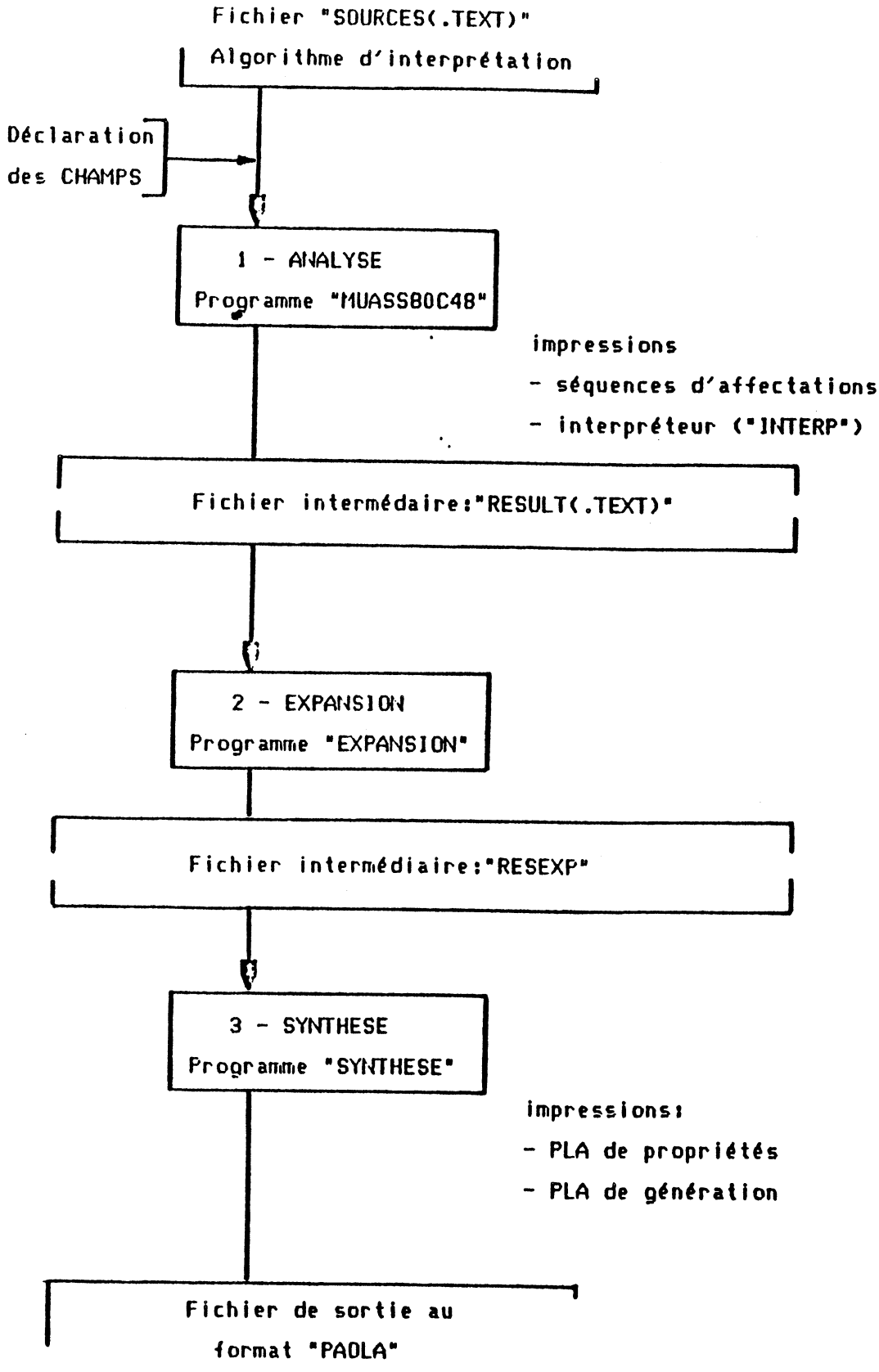


fig:III.95 Etapes d'exécution relatives au programme de génération "MACSIM" du contenue des PLAs

a) analyse (simulation) structurelle:

A partir de la forme textuelle de la description et de la déclaration des "CHAMPS" (objet), le programme d'analyse "MUASS8048" génère les tables (tables de points d'entrées, de longueur de séquences, de micro-instructions..etc) nécessaires à l'interprétation des micro-instructions ,et dresse le bilan de toutes les connexions relatives à la partie opérative (fichier "INTERPRETEUR", cf annexe 3).

b) expansion des champs:

La phase d'expansion permet d'établir le bilan des affectations relatives aux champs - parmi les ressources composant la partie opérative - corrélées à l'instant de leur activation; la paramétrisation du code opération autorise le regroupement par famille des instructions similaires.

c) synthèse:

L'exécution du programme de synthèse permet de générer le contenu des PLAs de propriétés et de génération.

c1) génération du PLA de propriétés

La recherche des états semblables dans l'ensemble des descriptions des familles d'instructions (fichier "RESEXP") autorise une extraction des propriétés; cette analyse permet d'en dénombrer 147, dont deux sont relatives au séquencement. Codées sur deux bits, elles signifient le nombre d'instants associé à chaque famille d'instructions.

On dresse une table de correspondance entre les instructions - référencées par leur code opération "CODOP(i)" - et les propriétés "P(j)" mises en oeuvre; un jeu d'instructions est décrit par un ensemble de 256 bits, la valeur "Cij = 1" du bit "i" indique que l'instruction "CODOP(i)" possède la propriété correspondante "P(j)". On peut donc en déduire la valeur des codes opérations à décoder (par expansion des bits à "1").

De même, une instruction est définie par un ensemble de propriétés CODOP(i) = Cij x P(j). Son code opération ne doit être décodé qu'une seule fois; un regroupement par colonnes permet d'associer un ensemble de propriétés relatives à une instruction.

On définit une matrice de 16 entrées - huit bits de code opérations complémentés - et 147 sorties; après réduction, elle est constituée de 69 monômes, représentant une surface de:

Si = 69 * (16 + 147) = 11247 unités.

L'implémentation est réalisée suite à une optimisation topologique (cf 3-3a)par le systhème PAOLA (CHU84, cf annexe n°10).

c2) génération du PLA de commandes

L'exécution du programme "synthèse" réalise la combinaison des propriétés avec l'instant relatif à leur activation, et génère un fichier constitué par les équations qui définissent la PLA de commandes; les ressources mises en oeuvre sont explicitées par l'interpréteur. (Note: les descriptions "MACSIM" ne sont pas directement exploitables par "PAOLA"; destinées au concepteur qui doit pouvoir les interpréter, elles doivent faire l'objet d'une traduction pour devenir compatibles).

```
CHAMPS
   .
   .
  etc.
  5  1   C10=(T10.P9)+(T18.P11)
  6  1   C11=(T4.P1)+(T14.P16)+(T15.P13)+(T16.P17)+(T18.P120)+(T10.P21)
     2   C12=(T14.P12)
  7  1   C13=(T9.P19)+(T19.P8)
     2   C14=(T19.P5)
     3   C15=(T10.P9)+(T18.P11)
     4   C16=(T19.P9)
     5   C17=(T6.P20)+(T10.P11)
     6   C18=(T19.P12)
  8  1   C19=(T10.P9)+(T18.P11)
     2   C20=(T6.P20)+(T10.P21)
  9  1   C21=(T10.P9)+(T18.P11)
 10  1   C22=(T18.P11)
  etc..
   .
   .
```

fig: III.96 Exemple de description du PLA génération

Les caractéristiques du PLA de génération sont les suivantes:

- 152 entrées (dont 147 en provenance du PLA de propriétés et 5 du générateur de temps),
- 205 monômes,
- 136 sorties.

Le codage des instants "Ti" est réalisé par une variable de sept bits composée des champs suivants:

Ti:= /nombre de cycles(1)//états(5)//phase(1)/

Le nombre de cycles contenu dans l'instruction courante est codé sur un bit ("0":= 10 cycles, "1":= 20 cycles); le codage de "1" parmi "n", généré par un compteur "Jonhson", représente les états de "00001" pour "S1" à "10000" pour "S5". La phase est codée sur un bit ("0":= P1 et "1":= P2), d'où la liste:

Codage des instants:

0	0000010
1	0000011
2	0000100
3	0000101
4	0001000
5	0001001
6	0010000
7	0010001
8	0100000
9	0100001
10	1000010
11	1000011
12	1000100
13	1000101
14	1001000
15	1001001
16	1010000
17	1010001
18	1100000
19	1100001

Précédant à sa génération, le PLA de commandes a fait l'objet d'une optimisation topologique.

3-3 Contraintes topologiques et implantation de la partie contrôle

Une bonne imbrication des différents blocs permet de minimiser les zones d'interconnexions, et par conséquent, la surface globale occupée ainsi que le temps de propagation.

Une génération "PAOLA" (cf annexe 10) est réalisée en appliquant des contraintes topologiques et dimensionnelles aux unités, des cellules spécifiques assurant l'interfaçage entre les différents éléments.

a) optimisation topologique des PLAs

Ainsi, on réalise une meilleure distribution des commandes en alignant la longueur du PLA de génération - le plus gros - sur celle de la partie opérative, soit une longueur de 2000 lambdas environ.

Les deux matrices "OU" - PLAs de générations et de propriétés - ont leurs sorties vers le bas; les entrées sont disposées à droite et en haut des matrices "ET" des PLAs de propriétés et de génération, respectivement.

On réalise une optimisation en surface après avoir défini la longueur des matrices; elles sont alors étendues dans le sens de la hauteur.

Ainsi, la duplication de 136 monômes du PLA de propriétés a permis de porter de 147 à 41 les colonnes générant les sorties. Si la surface globale de la matrice est sensiblement la même,

So = 205 * (16 + 41) = 11685 unités (contre 11247)

le PLA est adapté à la topologie générale; de surcroît, les conditions nécessaires au tracé - nombre de sorties inférieur au nombre de monômes - sont aussi atteintes. Dans le cas initial, la réalisation n'aurait pas été possible.

L'optimisation du PLA de génération est réalisée avec une priorité à l'ordonnancement des monômes sur la matrice "OU". Après exécution, les 152 entrées ont été ramenées à 43 colonnes et les 136 sorties à 34 niveaux, soit une surface de

So = 205 * (43 + 34) = 15785 unités (contre 59040)

On réalise ainsi un gain de surface de près de 73%!

Les caractéristiques dimensionnelles de la cellule de base, évaluées à 7x7 lambdas-carré, permettent de chiffrer la surface relative des diverses matrices; le nombre de canaux d'écartement - monômes fictifs nécessaires au routage des sorties et aux rappels de masse - représente 30% du nombre total de sorties.

PLAs	Nb de rectangles	Dimensions (lambdas)
PROPRIETES	PRO-ET = 5271	117 x 2032
	PRO-OU = 11258	337 x 2035
GENERATION	GEN-ET = 7672	313 x 2035
	GEN-OU = 5329	278 x 2063

fig:III.97 Caractéristiques dimensionnelles des matrices de PLAs de propriétés et de génération du 80C48

b) réalisation d'interfaces

b1) les étages amplificateurs

Destinés à reformater les signaux et autoriser l'attaque de charges plus importantes (ex: matrices de PLAs, lignes de commandes..etc), ils sont composés d'amplificateurs délivrant la valeur logique présente à l'entrée ainsi que son complément.

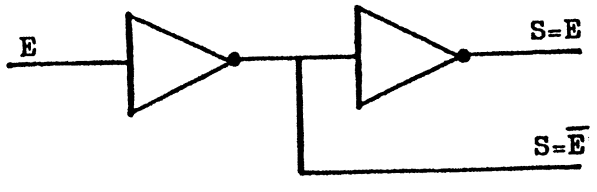


fig:III.98 Schéma fonctionnel d'un étage d'amplificateurs

b2) interfaçage: matrice ET/OU

L'interface PLA-ET/PLA-OU a pour rôle de séquencer et d'amplifier les signaux produits par les lignes de monômes de la matrice "ET".

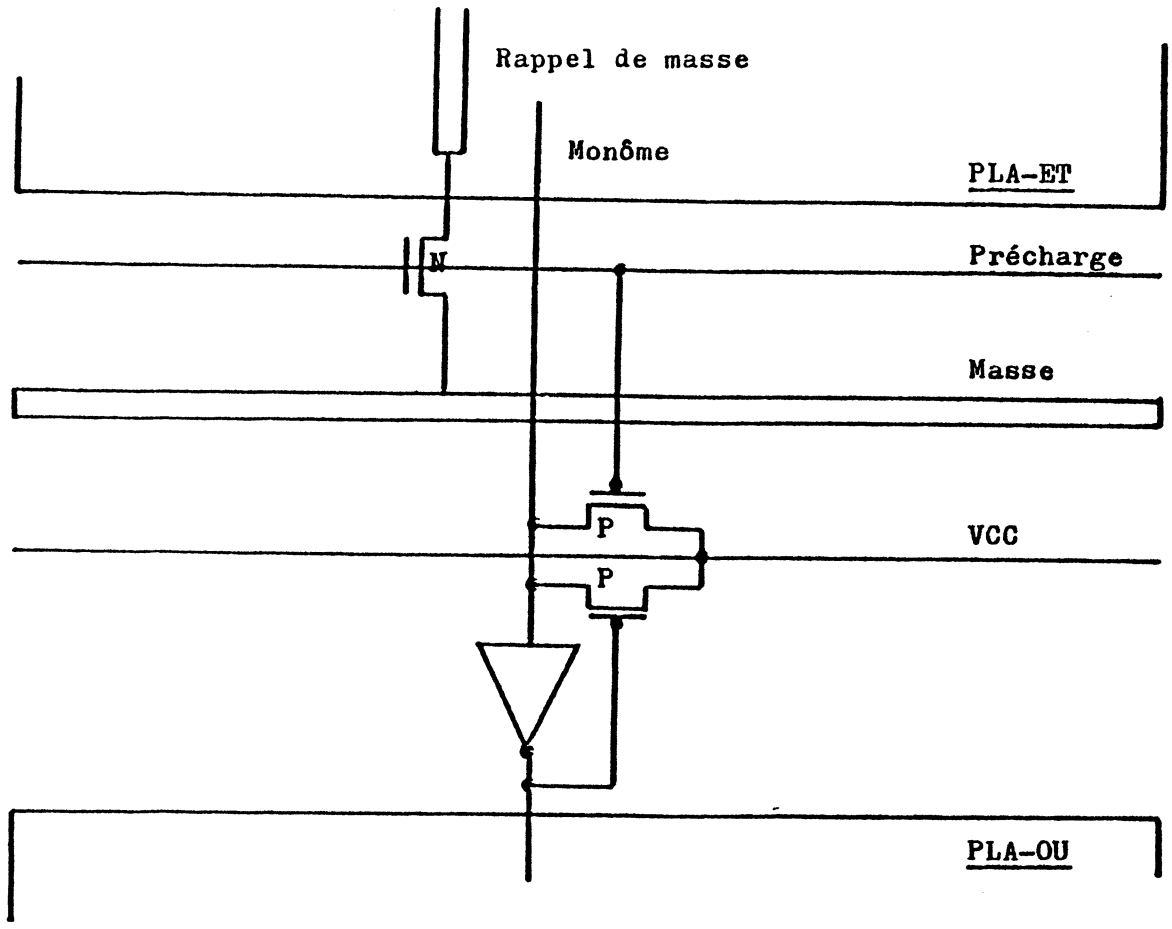


fig:III.99 Schéma logique d'une interface de matrice ET/OU pour PLA

c) conclusion

Le 8048 d'Intel exécute des instructions d'une durée finie de 1 ou 2 cycles; cette particularité nous permet d'implémenter une partie contrôle utilisant un générateur de temps, la structure mise en oeuvre étant particulièrement adaptée aux contraintes.

Il eût été facile d'augmenter les performances - diminution du temps d'exécution des instructions - par des ruptures de séquences conditionnelles; en effet, l'algorithme d'interprétation laisse apparaître beaucoup d'étapes inactives.

Une telle réalisation ne permet pas de conserver la compatibilité avec les circuits existant sur le marché; de surcroit, l'illégularité induite dans l'algorithme d'interprétation entraîne une complexification de la

partie contrôle (augmentation du nombre de propriété) matérialisée par un accroissement de la surface du silicium nécessaire.

L'architecture de la partie contrôle retenue est une architecture classique de type hiérarchisée; sa conception selon une approche multi-PLAs, avec utilisation des PLAs de propriétés et de paramètres permet d'obtenir une structure optimisée dont la réalisation est possible avec l'aide d'outils.

L'implémentation de tous les blocs fonctionnels de cette partie contrôle témoigne d'une grande souplesse; les différentes matrices s'emboîtent bien, car leurs dimensions sont "petites".

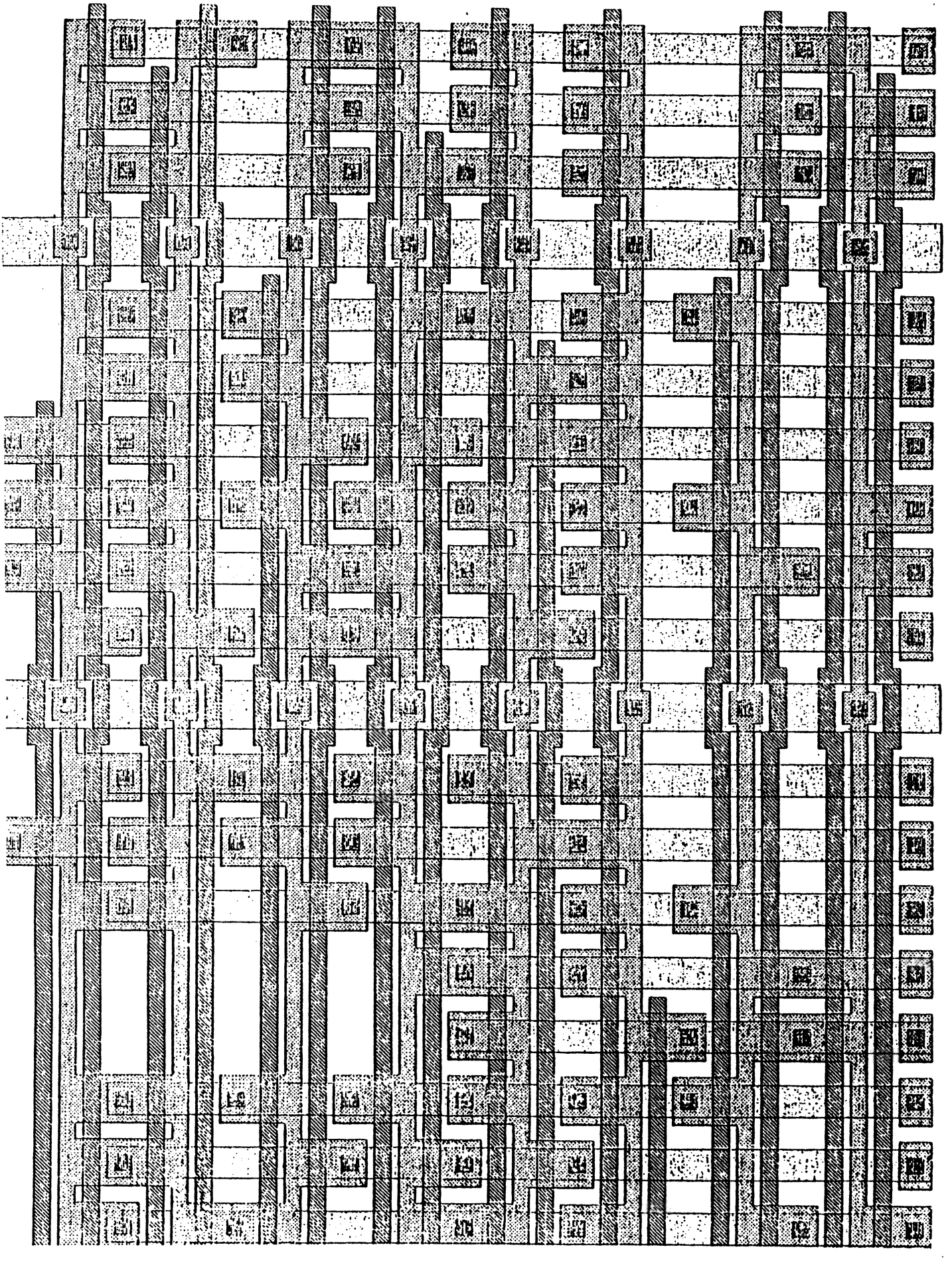
On obtient ainsi une structure compacte.

Son élaboration est possible par la mise en oeuvre d'outils de CAO tels que "MACSIM" et "PAOLA", logiciels développés par l'Equipe de Recherche en Architecture d'Ordinateurs de l'IMAG/TIM3. Intégrés sous un même système, leur contribution doit permettre l'élaboration d'outils de génération automatique de partie contrôle pour circuits VLSI.

"Extrait" de la matrice "OU" du PLA de Propriétés

```
hiplot -r aprop
Window: xmin = 1982.00, xmax = 0.00, ymin = 0.00, ymax = 110.00 (lambda-units)
Scale: 0.0089 inches (2.26 mm) per lambda-unit, 111.2 lambda-units per inch (0.443 lambda-units per mm)

prop
```

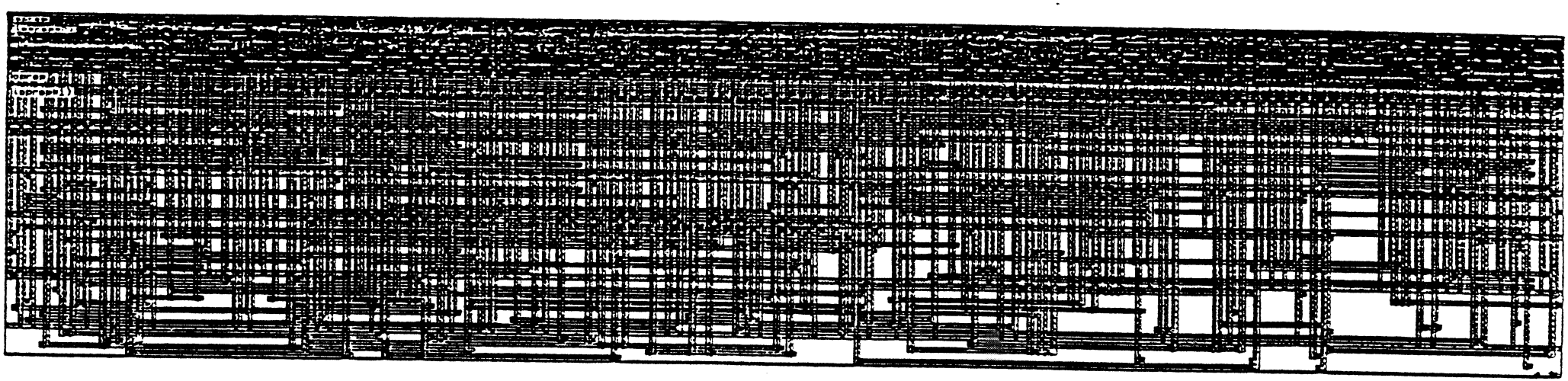


PLA DE PROPRIETES

ouerdani:Thu May 3Ø 13:4Ø:58 1985

phlplot prop.phl
Window: xmin = Ø.ØØ, xmax = 496.ØØ, ymin = -2.ØØ, ymax = 19Ø4.ØØ (lambda-units)
Scale: Ø.ØØ551 inches (Ø.14Ø mm) per lambda-unit, 182 lambda-units per inch (7.15 lambda-units per mm)

prop.phl

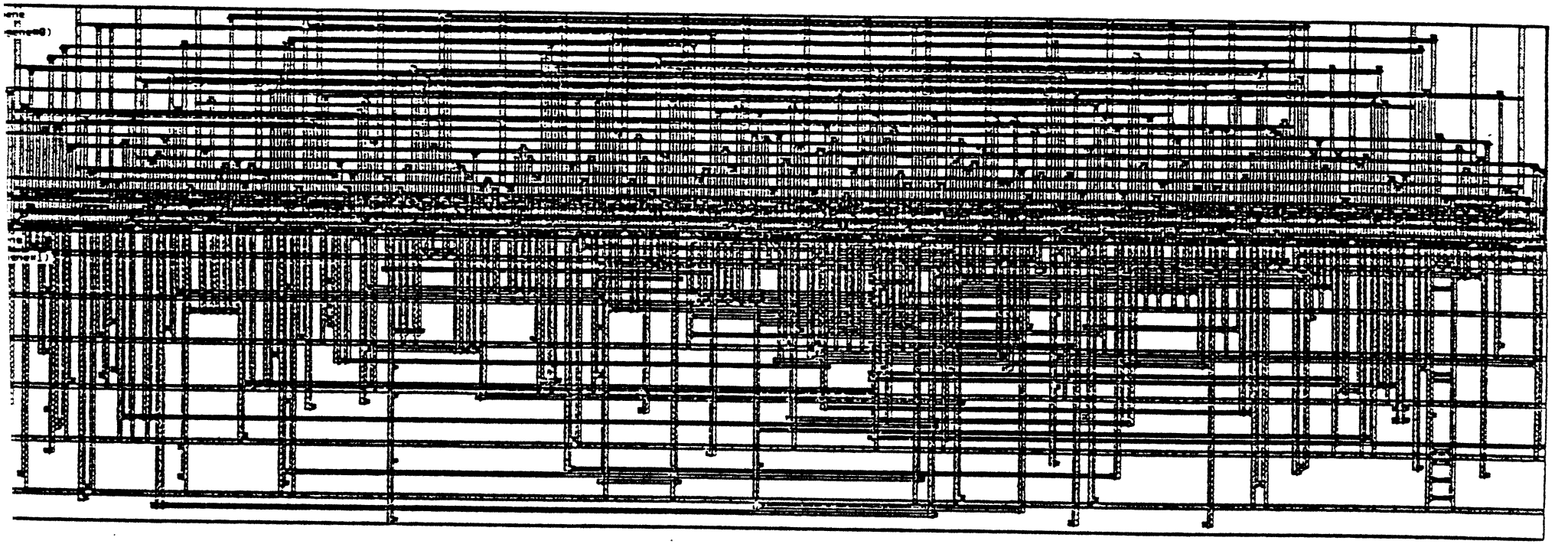


PLA DE GENERATION

jerdani:Thu May 30 11:43:36 1985

ilplot gene
indow: xmin = 0.00, xmax = 672.00, ymin = -2.00, ymax = 1904.00 (lambda-units)
cale: 0.00551 inches (0.140 mm) per lambda-unit, 182 lambda-units per inch (7.15 lambda-units per mm)

ene



III 4 Conclusion

L'étude et la réalisation du 80C48 nous a permis de tester et de valider les différents outils (LUBRICK, MACSIM et PAOLA) d'aide à la conception mis en oeuvre dans l'approche descendante CAPRI.

Nous avons montré comment établir un algorithme d'interprétation supporté par une architecture à deux bus et séquencée par deux phases d'horloge; la décomposition en transferts élémentaires des instructions de base a été réalisée de façon manuelle et fut l'objet d'un travail important, pour aboutir à une description finale. Une bonne analyse n'en est pas moins fondamentale pour l'obtention d'une architecture régulière ("bit slice"), facilement implémentable. On peut toutefois regretter l'absence de logiciel spécifique à cet égard (recherche du chemin de donnée, mise en évidence des chaines critiques, ...etc); un outil est actuellement en étude au sein du laboratoire.

Les schémas logiques ont été réalisés en circuiterie CMOS statique.

L'implémentation des cellules s'est faite selon une structure ordonnée matricielle qui autorise l'utilisation du dessin symbolique "STICK"; cette approche permet d'avoir rapidement une évolution topologique et dimensionnelle du circuit, indépendamment de critères technologiques précis; le passage du "STICK" au dessin au micron se fait par simple dilatation de la grille de support.

Les cellules de base ont été assemblées par programme grâce à LUBRICK; cette technique d'implantation régulière permet effectivement un travail plus rapide et plus sûr, sachant qu'une intervention sur une cellule de base ne remet pas en cause la description globale du circuit. La perte de surface induite par une telle méthode se justifie amplement au regard du gain réalisé sur le coût de l'étude du circuit.

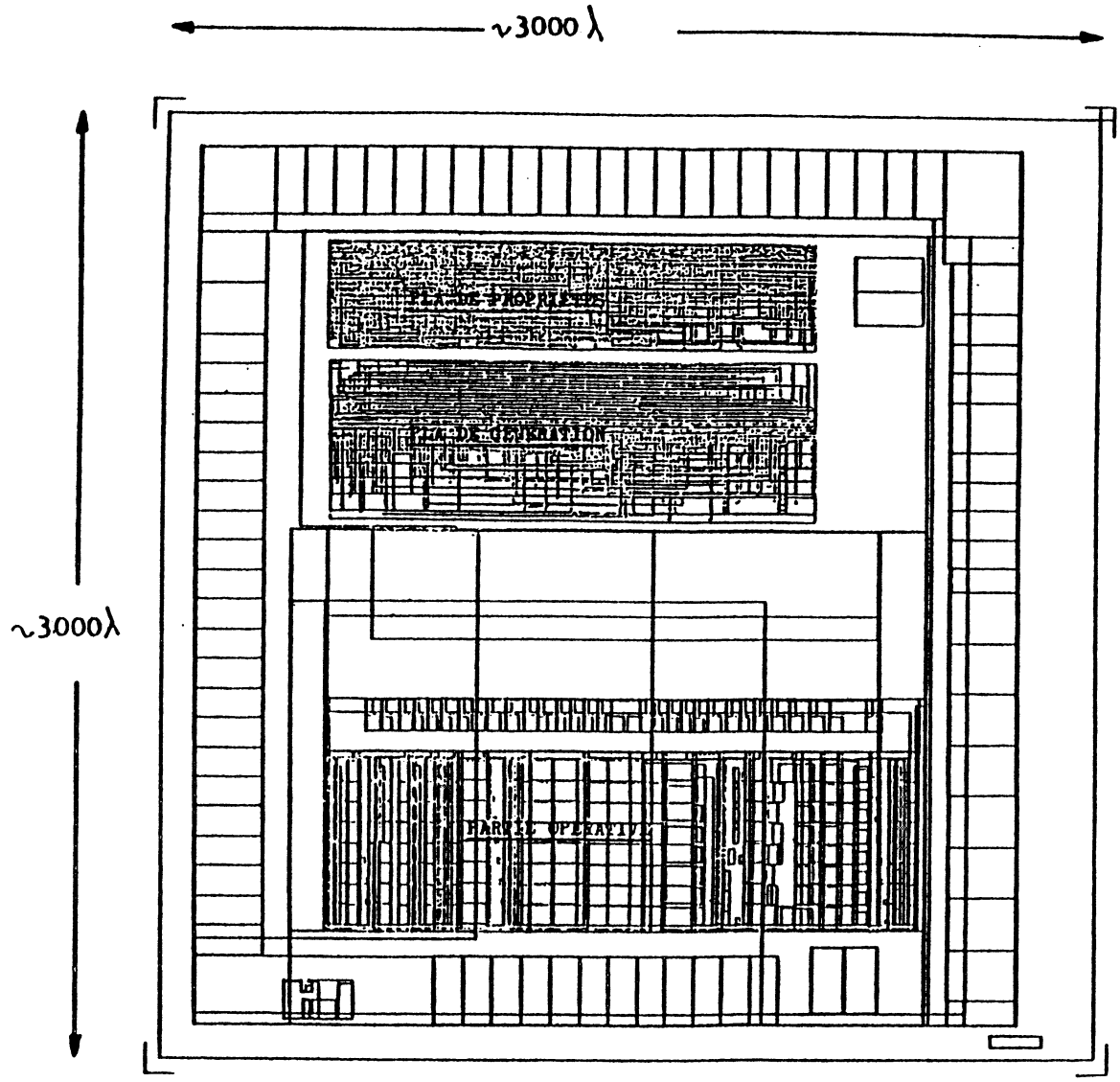
Une analyse par MACSIM de l'algorithme d'interprétation à permis de générer de façon automatique le contenu des PLAs sous une forme exploitable; leur implantation avec optimisation topologique de surface a été réalisée par PAOLA. Le 80C48 s'est révélé bien adapté à acceuillir une struture de contrôle à deux matrices (PLA de propriétés, PLA de commandes) avec un générateur de temps, chaque instruction ayant - conformément aux spécifications - une durée d'exécution finie de dix ou vingt phases.

La réalisation du 80C48 nous a permis de valider la méthode de conception descendante CAPRI qui, partant de la spécification

fonctionnelle d'un circuit permet d'aboutir à une implémentation matérielle d'un style donné.

Cette étude ne fait que confirmer l'idée que la maîtrise de la complexité des circuits futurs passe par l'utilisation de structures régulières dont la régularité est facilement exploitable par des outils de génération automatique; en d'autres termes, les VLSI de demain ne pourront plus être vus comme une suite de portes logiques mais plutôt comme un ensemble de blocs fonctionnels dont le développement, l'implémentation et le test seront facilités par l'emploi d'outils spécifiques.

Le rôle du concepteur "n'est plus réduit" qu'à une prise de décision au moment de la définition de l'architecture interne; il peut être conforté dans son choix par une exploration rapide des différentes possibilités offertes, lui permettant ainsi de développer de façon automatique la meilleure des solutions.



PLAN DE MASSE DU 80C48 (CMOS)

BIBLIOGRAPHIE:

(ANC74) ANCEAU François
"Contribution à l'étude des systèmes hiérarchisés de ressources dans l'architecture des machines informatiques."
Thèse d'état, INPG, décembre 1974.

(ANC78) ANCEAU François
"Principes et mise en oeuvre des microprocesseurs"
Cours polycopié de l'ENSIMAG, janvier 1978.

(ANC80) ANCEAU François
"Architecture and design of Von Neumann microprocesseurs".
Ecole d'été de Louvin la Neuve, 1980.

(ANC81) ANCEAU François
"Paramètre électriques pour le projet CMP".
Rapport interne IMAG, 1981.

(ANC82) ANCEAU François & REIS Ricardo
"Complex integrated circuit design strategy".
Journal of Solid State.
Circuits Vol SC-17 n°3, june 1982.

(ANC83) ANCEAU François
"A design methodology and a silicon compiler for the VLSI circuits specified by algorithms".
Third CALTECH conference on Very Large Scale Integration.
Edited by Randal Bryant - Computer Science Press, march 1983.

(ALG83) BERTRAND François
"Algorithme d'interprétation des microprocesseurr 8048/8041".
Rapport interne IMAG, juin 1983.

(APL82) BERTRAND François
"Application d'une démarche descendante au 8648".
Rapport interne IMAG, septembre 1982.

(APL83) BERTRAND François
"Application d'une démarche descendante au 8748 (suite)".
Rapport interne IMAG, mars 1983.

(AUM) AUMIAUX M.
"Fonction logiques et arithmétiques binaire".
Logique binaire et ordinateurs.
Edition Masson & Cie

(CAP82) ANCEAU François & SHOELLKOPF Jean Pierre
"CAPRI:a silicon compiler for VLSI circuits spécified by algorithms".
VLSI82 Bristol University, july 1982.

(CHU84) CHUQUILLANQUI Samuel
"Une nouvelle approche pour l'optimisation topologique et l'automatisation du dessin des masques de PLAs complexes".
Thèse de Docteur-Ingénieur à l'INPG, octobre 84.

(DEM82) BERTRAND François & SAHBATOU Djamel
"Démarche descendante appliquée aux micropocesseur 8048".
Rapport de DEA IMAG, juin 1982.

(DES) JESPERS P. & SEQUIN H C. & VAN DE WIELE F.
"Design methodology for VLSI circuits".
Applied Science n° 47.

(DUR79) DURET Alain
"Participation à la conception et la réalisation en LSI de la partie opérative d'une machine intégrée".
Thèse de Docteur 3ième cycle à l'INPG, décembre 1979.

(GAL84) GALLAIS Rémy
"Participation à la réalisation d'un microprocesseur 16 bits interprétant le P Code".
Thèse d'Ingénieur CNAM Grenoble, mars 1984.

(INT41) INTEL
"UPI-41A User's manuel".

(INT48) INTEL
"MCS-48 Family of single chip microcomputers".
User's manuel.

(IRE82) MARINE Souheil & ANCEAU François & JAHIDI K.
"IRENE: un langage de description de circuits intégrés logiques".
Rapport de recherche IMAG n° 356.

(JER83) JERRAYA Ahmed
"COMFOR: une nouvelle approche pour la vérification des masques des circuits integrés".
Thèse de Docteur-Ingénieur à l'INPG, novembre 1983.

(LAT79) "VLSI design methodology, the problem of the 80's for microprogram design."
16th Design Automation Conference 1979.

(LUC81) JERRAYA Ahmed & GUYOT Alain
"Présentation de LUCIE"
Rapport interne IMAG, juin 1981.

(MAT84) MATHERON Christophe
"Implantation de la partie opérative du 80C48".
Rapport de DEA IMAG, juin 1984.

(MEA) MEAD Carver & CONWAY Lynn
"Introduction to VLSI systems".
Addison Wesley publishing company.

(MEI) MEINADIER Jean Pierre
"Structure et fonctionnement des ordinateurs"
Edition Larousse, série Informatique.

(MIC83) LAURENT Jacques & COURTOIS Bernard
"Présentation et utilisation d'un outil de test de VLSI par faisceau d'électrons".
Rapport de recherche IMAG n° 374, mai 1983.

(OBJ84) OBJOIS Philippe
"Etude de l'implantation de la partie contrôle du 80C48".
Rapport de DEA IMAG, juin 1984.

(OBR82) OBREBSKA Monika
"Etude comparative de différentes méthodes de conception des parties contrôle des microprocesseurs".
Thèse de Docteur-Ingénieur, INPG juin 1982.

(REI83) REIS Ricardo
"Evaluateur topologique prédictif pour la génération automatique des plans de masse de circuits VLSI"
Thèse de Docteur-Ingénieur, INPG juin 83.

(SAH83) SAHBATOU Djamel
"Conception de la partie opérative CMOS compatible au 80C48 et 80C41".
Rapport interne IMAG, novembre 83.

(SAH84) SAHBATOU Djamel
"Une méthode de conception de microprocesseurs CMOS: application au 80C48 (Intel)."
Thèse de Docteur-Ingénieur INPG, novembre 1984.

(SCH77) SCHOELLKOPF Jean Pierre
"Machine PASC-HLL: définition d'une architecture pipeline pour une unité centrale adaptée au langage Pascal.
Thèse de Docteur 3ième cycle à l'INPG, 1977.

(SCH83) SCHOELLKOPF Jean Pierre
"LUBRICK: a silicon assembler and its application to datapath design for FISC".
VLSI 83, Trondheim, august 1983 (R.R. IMAG mars 1983).

(SCH85) SCHOELLKOPF Jean Pierre
"SILICIEL: contribution à la compilation du silicium et à l'architecture des circuits intégrés"
Thèse d'Etat, INPG, avril 1985.

(SIM83) BERTRAND François & SCHOELLKOPF Jean Pierre
"Simulation statique (comportementale) du 80C48".
Rapport interne IMAG, juillet 1983.

(SUZ81) SUZIM Altamiro Amadeu
"Etude des parties opératives à éléments modulaires pour processeurs monolithiques"
Thèse de Docteur-Ingénieur à l'INPG, novembre 1981.

(VLS82) ANCEAU François
"VLSI Processeur Architecture and design."
VLSI82 Bristol University, july 1982.

ANNEXES

ANNEXE 1: SYMBOLES ET ABREVIATIONS UTILISEES

1) Ressources matérielles

L'implémentation du 80C48 nécessite la mise en oeuvre des ressources matérielles suivantes:

a) éléments de mémorisation

W(8) registre de "travail" et d'adressage de la mémoire vive
W(8):= /WH(5)//WL(3)/

BAS(8) bascules d'états, compteur de programme - poids forts -
BAS(8):= /F1(1)//IBF(1)//B(1)//I(1)//PCH(4)/

STH(8) registre STH - poids forts -, bascule de sélection de banques mémoire et pointeur de pile du mot d'états programme - pois faibles -.
STH(8):= /STH74(4)//DBF(1)//S2(1)//S1(1)//S0(1)/

A(8) registre accumulateur
A(8):= /AH(4)//AL(4)/

T(8) registre temporisateur

PCL(8) compteur de programme - poids faibles -
PC(12):= /PCH(4)//PCL(8)/

U1(8) registre tampon d'UAL
U1(8):= /U1H(4)//U1L(4)/

U2(8) registre tampon d'UAL

PSH(8) bascules (du mot d'états programme - poids forts - et d'interruption)
PSH(8):= /CY(1)//AC(1)//F0(1)//BS(1)//OVF(1)//T1(1)/
/T0(1)//TF(1)/

RI(8) registre d'instruction

DP2L(8) registre "données" du port P2

AP2L(4) registre "adresses" du port P2

P1L(8) registre tampon du port P1

DBBOUT registre de sortie du port B

DBBIN registre d'entrée du port B

RAM(8) mémoire vive (adressée par W)

ROM(8) mémoire morte (adressée par PC)

DBF(1) bascule de sélection ROM externe ou interne

T-OVF(1) bascule de débordement du timer

EXT-I(1) sélection des interruptions externes
INT-I(1) sélection des interruptions internes
JTF(1) bascule d'activation de la logique de branchement
TB(1) bascule de test de l'état d'un bit
TZ(1) bascule de test du zéro

b) les opérateurs

CTE(8) générateur de constantes
UAL(8) unité arithmétique et logique
DAA(8) unité d'ajustement décimal
SWAP(8) opérateur de permutation poids-faibles/poids-forts
SHIFTER opérateur de décalage
TBZ(8) test du zéro
TBB(8) test binaire

2) Lexique des abbréviations

a	préfixe d'adressage
A	accumulateur
AC	retenue auxiliaire
ad	adresse de la mémoire programme
AP2L	registre "adresse" du port "P2"
B	port "données/adresses" (noté aussi "P0")
BS	sélection de banques
BAS	bascules et compteur de programme - poids forts -
CAO	conception assistée par ordinateur
CY	retenue
CLK	horloge
CMP	circuit multi projet
CNT	compteur
CPU	unité centrale
CTE	générateur de constantes
d	données
DAA	ajustement décimal

DBBIN	registre d'entrée du port "B"
DBBOUT	registre de sortie du port "B"
DBF	bascule de sélection de banque
DP2L	registre "données" du port "P2"
ENTO	sélection de la broche "T0" en horloge externe
F0,F1	drapeaux
I	bascule d'interruption
IBF	bascule d'interruption ("buffer full")
INT-I-E	fin d'exécution d'un traitement d'interruption interne
JTF-E	fin d'exécution d'une routine de branchement
OVF	bascule de débordement ("overflow")
P1L	registre tampon du port "P1"
PC	compteur de progrmme
PCL	compteur de programme - poids faibles -
PCH	compteur de programme - poids forts -
Pi	ports externes du système étendu (i:=4,..,7)
PRES	entrée du registre temporisateur (configuration "timer")
Pp	ports internes (p:=0,..,4)
PSH	registre de bascules (mot d'états programme - poids faibles - et d'interruption)
PSW	mot d'états programme
RAM	mémoire vive
ROM	mémoire morte
Rr	mot de mémoire vive
S2,S1,S0	pointeur de pile du mot d'états programme
SC	sélection du compteur
ST	sélection du registre temporisateur
SHIFTER	opérateur de décalage
SL	décalage à gauche
SP	pointeur de pile
SR	décalage à droite
STH	registre (registre "STH" - poids forts -, bascule de sélection de banque mémoire et pointeur de pile du mot d'états programme
SWAP	permutation
T	temporisateur
TBB	test d'un bit

TBZ	test du zéro
T0,T1	bascules
TF	drapeau de temporisateur
U1,U2	registres tampon d'UAL
U1H	registre tampon d'UAL - poids forts -
U1L	registre tampon d'UAL - poids faibles -
UAL	unité arithmétique et logique
W	registre d'adressage de la mémoire vive
WH	registre d'adressage de la mémoire vive - poids forts -
WL	registre d'adressage de la mémoire vive - poids faibles -
<--	affectation
:=	identification
<y>	champ de "y" bits
<x:y>	champ de "y" bits à partir du bit de rang "x"
xD	"x" codé décimal; (x:=0,..,9)
xxH	"xx" codé hexadécimal; (x:=0,..,15)
xxxxB	"xxxx" codé binaire; (x:=0,1)
(x)	variable "x"
((x))	variable adressée par la variable "x"
x(y)	élément de "x" paramétré par "y"
x//y	concaténation du champ "x" avec le champ "y"
*	commentaire

ANNEXE 2: LE JEU D'INSTRUCTION DU 80C48

(Mnémonique/ Codops/ Description symbolique)

Mnémonique	Codops	Description	
Instructions accumulateur:			
ADD A,Rr	0110 1rrr	A <-- A + Rr	Rr:=0..8
ADD A,aRr	0110 000r	W <-- Rr	Rr:=0,1
		A <-- A + RAM(W)	
ADD A,id	0000 0011	A <-- A + ROM(PC)	
	aaaa aaaa		
ADDC A,Rr	0111 1rrr	A <-- A + Rr + PSW(7)	Rr:=0..8
		PSW(7) <-- 0	
ADDC A,id	0001 0011		
	aaaa aaaa	A <-- A + ROM(PC) + PSW(7)	
		PSW(7) <-- 0	
ADDC A,aRr	0111 000r		Rr:=0,1
		W <-- Rr	
		A <-- A + RAM(W) + PSW(7)	
		PSW(7) <-- 0	
ANL A,Rr	0101 1rrr	A <-- A . Rr	Rr:=0..8
ANL A,aRr	0101 000r	W <-- Rr	Rr:=0,1
		A <-- A . RAM(W)	
ANL A,id	0101 0011		
	aaaa aaaa	A <-- A . ROM(PC)	
ORL A Rr	0100 1rrr	A <-- A ou Rr	Rr:=0..8
ORL A,aRr	0100 000r	W <-- Rr	Rr:=0,1
		A <-- A ou RAM(W)	
ORL A,id	0100 0011		
	aaaa aaaa	A <-- A ou ROM(PC)	
XRL A,Rr	1101 1rrr	A <-- A oux Rr	Rr:=0..8
XRL A,aRr	1101 000r	W <-- Rr	Rr:=0,1
		A <-- A oux RAM(W)	
XRL A,id	1101 0011		
	aaaa aaaa	A <-- A oux ROM(PC)	
INC A	0001 0111	A <-- A + 1	
DEC A	0000 0111	A <-- A - 1	
CLR A	0010 0111	A <-- 0	

CPL A	0011 0111	A <-- A*	
DAA	0101 0111		
		A(3:0)D <-- A(3:0)B	
		A(7:4)D <-- A(7:4)B	
SWAP A	0100 0111	A(7:4) <-- A(3:0)	
		A(3:0) <-- A(7:4)	
RL A	1110 0111	A(n+1) <-- A(n)	n:= 0..6
		A(0) <-- A(7)	
RLC A	1111 0111	A(n+1) <-- A(n)	n:= 0..6
		A(0) <-- PSW(7)	
		PSW(7) <-- A(7)	
RR A	0111 0111	A(n) <-- A(n+1)	n:= 0..6
		A(7) <-- A(0)	
RRC A	0110 0111	A(n) <-- A(n+1)	n:= 0..6
		A(7) <-- PSW(7)	
		PSW(7) <-- A(0)	

Instructions d'entrées/sorties:

IN A, Pp	0000 10pp	A <-- Pp	Pp:=0..4
OUTL Pp,A	0011 10pp	Pp <-- A	Pp:=0..4
ANL Pp,id	1001 10pp		
	aaaa aaaa	Pp <-- Pp . ROM(PC)	Pp:=0..4
ORL Pp,id	1000 10pp		
	aaaa aaaa	Pp <-- Pp ou ROM(PC)	Pp:=0..4
INS A,B	0000 1000	A <-- B	
OUTL B,A	0000 0010	B <-- A	
ANL B,id	1001 1000		
	aaaa aaaa	B <-- B . ROM(PC)	
ORL B,id	1000 1000		
	aaaa aaaa	B <-- B ou ROM(PC)	
MOVD A,Pi	0000 11ii	A(3:0) <-- Pi	Pi:= 4..7
		A(7:4) <-- 0H	
MOVD Pi,A	0011 11ii	Pi <-- A(3:0)	Pi:= 4..7
ANLD Pi,A	1001 11ii	Pp <-- Pi . A(3:0)	Pi:= 4..7
ORLD Pi,A	1000 11ii	Pi <-- Pi ou A(3:0)	Pi:= 4..7

Instructions registre:

INC Rr	0001 1rrr	Rr <-- Rr + 1	Rr:=0..8
INC aRr	0001 000r	W <-- Rr	Rr:=0,1
		RAM(W) <-- RAM(W) + 1	
DEC Rr	1100 1rrr	Rr <-- Rr - 1	Rr:=0..8

Instructions de branchement:

JMP ad	aaa0 0100		
	aaaa aaaa		
		PCH(2:0) <-- aaa	
		PCH(3) <-- DBF	
		PCL <-- ROM(PC)(3:0)	
JMPP aA	1011 0011	PCL <-- A	
		PCL <-- ROM(PC)	
DJNZ Rr,ad	1110 1rrr		Rr:=0..8
	aaaa aaaa	Rr <-- Rr - 1	
		PCL <-- ROM	si Rr > 0
JC ad	1111 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si PSW(7)=1
JNC ad	1110 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si PSW(7)=0
JZ ad	1100 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si A=0
JNZ ad	1001 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si A>0
JT0 ad	0011 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si T0=1
JNT0 ad	0010 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si T0=0
JT1 ad	0101 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si T1=1
JNT1 ad	0100 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si T1=0
JF0 ad	1011 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si PSW(5)=1
JF1 ad	0111 0110		
	aaaa aaaa	PCL <-- ROM(PC)	si F1=1
JTF ad	0001 0110		

	aaaa aaaa	PCL <-- ROM(PC) si TF=1
JNI ad	1000 0110	
	aaaa aaaa	PCL <-- ROM(PC) si I=0
JBb ad	bbb1 0010	
	aaaa aaaa	PCL <-- ROM(PC) si A<b>=1; b=0,1

Instructions de sous-routine:

CALL ad	aaa1 0100	
	aaaa aaaa	RAM(SP) <-- PCL
		RAM(SP) <-- PCH
		RAM(PC) <-- PSW<7:4>
		PSW<2:0> <-- PSW<2:0> + 1
		PCH<2:0> <-- aaa
		PCH<3> <-- DBF
		PCL <-- ROM(PC)
RET	1000 0011	
		PSW<2:0> <-- PSW<2:0> - 1
		PCH <-- RAM(SP)
		PCL <-- RAM(SP)
RETR	1001 0011	
		PSW<2:0> <-- PSW<2:0> - 1
		PSW<7:4> <-- RAM(SP)
		PCH <-- RAM(SP)
		PCL <-- RAM(SP)

Instructions de transferts de données:

MOV A,Rr	1111 1rrr	A <-- Rr	Rr:=0..8
MOV A,aRr	1111 000r	W <-- Rr	Rr:=0,1
		A <-- RAM(W)	
MOV A,id	0010 0011		
	aaaa aaaa	A <-- ROM(PC)	
MOV Rr,A	1010 1rrr	Rr <-- A	Rr:=0..8
MOV aRr,A	1010 000r	W <-- Rr	Rr:=0,1
		RAM(W) <-- A	
MOV Rr,id	1011 1rrr		Rr:=0..8
	aaaa aaaa	Rr <-- ROM(PC)	
MOV aRr,id	1011 000r		Rr:=0,1

	aaaa aaaa	W <-- Rr	
		RAM(W) <-- ROM(PC)	
MOV A,PSW	1100 0111	A <-- PSW	
MOV PSW,A	1101 0111	PSW <-- A	
XCH A,Rr	0010 1rrr	A <-- Rr	Rr:=0..8
		Rr <-- A	
XCH a,aRr	0010 000r	W <-- Rr	Rr:=0,1
		A <-- RAM(W)	
		RAM(W) <-- A	
XCHD A,Rr	0011 000r	A(3:0) <-- RAM(W)(3:0)	Rr:=0,1
		RAM(W)(3:0) <-- A(3:0)	
MOVX A,aRr	1000 000r	W <-- Rr	Rr:=0,1
		A <-- RAMX(W)	
MOVX aRr,A	1001 000r	W <-- Rr	Rr:=0,1
		RAMX(W) <-- A	
MOVP A,aA	1010 0011	PCL <-- A	
		A <-- ROM(PCL)	
MOVP3 a,aA	1110 0011	PCL <-- A	
		PCH(2:0) <-- 011B	
		A <-- ROM(PCL)	

Instructions compteur/temporisateur:

MOV A,T	0100 0010	A <-- T
MOV T,A	0110 0010	T <-- A
STRT T	0101 0101	ST <-- 1B
STRT CNT	0100 0101	SC <-- 1B
STOP TCNT	0110 0101	ST,SC <-- 0B
EN TCNT I	0010 0101	EN TCNT <-- 1B
DIS TCNT I	0011 0101	DISTCNT <-- 0B

Instructions de contrôle:

EN I	0000 0101	ENI <-- 1B
DIS I	0001 0101	DISI <-- 1B
SEL RB0	1100 0101	PSW(4) <-- 0B
SEL RB1	1101 0101	PSW(4) <-- 1B
SEL MB0	1110 0101	DBF <-- 0B
SEL MB1	1111 0101	DBF <-- 1B

ENTO CLK |0111 0101| ENTO <-- 1B

ANNEXE 3: LES CONNEXIONS DU 80C48

```
(*** INTERPRETEUR ***)
case champ( 1) of
        '1':DBBI:=B;
        '2':DBBI:=if RI<1:0> = 00 then B
end;
case champ( 2) of
        '1':B:=DBBO;
        '2':B:=if RI<1:0> <> 00 then DBBO;
        '3':B:=if RI<1:0> = 00 then DBBO;
        '4':B:=if OU then DBBO
end;
case champ( 3) of
        '1':DBBO:=busg;
        '2':DBBO:=if RI<1:1> then busg
end;
if champ( 4)='1' then P1L:=if RI<1:0> = 01 then busg;
if champ( 5)='1' then P1:=if RI<1:0> = 01 then P1L;
case champ( 6) of
        '1':AP2L:=busg mod 16;
        '2':AP2L:=if OU then busg mod 16
end;
case champ( 7) of
        '1':P2:=AP2L;
        '2':P2:=if RI<1:0> <> 00 then AP2L;
        '3':P2:=if RI<1:0> = 10 then DP2L;
        '4':P2:=if RI<1:0> <> 00 then DP2L;
        '5':P2:=DP2L;
        '6':P2:=if ou then AP2L
end;
case champ( 8) of
        '1':DP2L:=if RI<1:0> = 10 then busg;
        '2':DP2L:=busg
end;
if champ( 9)='1' then P3L:=if RI<1:0> = 11 then busg;
if champ(10)='1' then P3:=if RI<1:0> = 11 then P3L;
```

```
case champ(11) of
        '1':busa:=PCL;
        '2':busa:=BAS30;
        '3':busa:=T;
        '4':busa:=A;
        '5':busa:=16*PSH74;
        '6':busa:=W;
        '7':busa:=16*STH74;
        '8':busa:=RI;
        '9':busa:=RAM(W);
        'A':busa:=8*STH73;
        'B':busa:=STH20;
        'C':busa:=16*PSH74 + BAS30;
        'D':busa:=16*PSH74 + PSH(3) + STH20;
        'E':busa:=16*BAS74
end;
case champ(12) of
        '1':busb:=if STH(3)='0' then ROM(PC) else busg;
        '2':busb:=sual;
        '3':busb:=if STH(4)='0' then PLA(0) else PLA(5);
        '4':busb:=RAM(W);
        '5':busb:=PLA(3);
        '6':busb:=PLA(1);
        '7':busb:=busg;
        '8':busb:=if RI(1:0) = 00 then busg;
        '9':busb:=PLA(2);
        'A':busb:=PLA(4);
        'B':busb:=case RI(4) of;
        'C':busb:=if RI(2:1) = 00 then sual;
        'D':busb:=PLA(7);
        'E':busb:=case INTER of;
        'F':busb:=PLA(6)
end;
```

```
 case champ(13) of
        '1':busg:=DBBI;
        '2':busg:=busb;
        '3':busg:=case RI(1:0) of;
        '4':busg:=if RI(1:0) then DBBI;
        '5':busg:=P2
        '6':busg:=P1
end;
case champ(14) of
        '1':WL:=busb mod 8;
        '2':WL:=if RI(0) = 0 then busb mod 8
end;
case champ(15) of
        '1':WH:=busb div 8;
        '2':WH:=if RI(0)= 0 then busb div 8
end;
if champ(16)='1' then PCL:=busb;
if champ(17)='1' then PCH:=busb mod 8;
if champ(18)='1' then RI:=busb;
case champ(19) of
        '1':T:=if PRES then busb;
        '2':T:=busb
end;
if champ(20)='1' then U1L:=busa mod 8;
if champ(21)='1' then U1H:=busa div 8;
if champ(22)='1' then U2:=busb;
case champ(23) of
        '1':AL:=busb mod 16;
        '2':AL:=if TDAL then busb mod 16;
        '3':AL:=if TDAH then busb mod 16;
        '4':AL:=if RI(1:0) = 00 then busb mod 16
end;
```

```
case champ(24) of
        '1':AH:=busb div 16;
        '2':AH:=if TDAL then busb div 16;
        '3':AH:=if TDAH then busb div 16;
        '4':AH:=if RI(1:0) = 00 then busb div 16
end;
case champ(25) of
        '1':RAM(W):=busb;
        '2':RAM(W):=busa;
        '3':RAM(W):=busa mod 16
end;
case champ(26) of
        '1':opual:=U1p0;
        '2':opual:=case RI(7:4) of;
        '3':opual:=0pU2;
        '4':opual:=U1p1;
        '5':opual:=0p0;
        '6':opual:=U1oux1;
        '7':opual:=U1pU2;
        '8':opual:=U1p0sw;
        '9':opual:=if RI(4)='0' then U1p0sl else U1p0slc;
        'A':opual:=if RI(4)='1' then U1p0sr else U1p0src;
        'B':opual:=case RI(4) of;
        'C':opual:=U1p0sr;
        'D':opual:=case COMBI1 of;
        'E':opual:=U1pU2sl;
        'F':opual:=case INTER of;
        'G':opual:=case Reset of
end;
```

```
case champ(27) of
        '1':cin:=1;
        '2':cin:=PSH(3);
        '3':cin:=if RI(7:4) = 0111 then PSH(7) else 0;
        '4':cin:=0;
        '5':cin:=if RI(7:4) = 0001 then PSH(7) else 0;
        '6':cin:=if TZ='1' then 0 else 1;
        '7':cin:=case COMBI1 of;
        '8':cin:=case INTER of;
        '9':cin:=if Reset then 0 else 1
end;
case champ(28) of
        '1':BAS20:=busb mod 8;
        '2':BAS20:=if OU then busb mod 8
end;
case champ(29) of
        '1':BAS(3):=bit(3,busb);
        '2':BAS(3):=if OU then bit(3,busb)
end;
case champ(31) of
        '1':BAS(5):=if RI(5,1) = 01 then bit(5,busb);
        '2':BAS(5):=if RI(4,1) = 11 then bit(5,busb)
end;
case champ(33) of
        '1':BAS(7):=if RI(5) = 1 then bit(7,busb);
        '2':BAS(7):=if RI(5,1) = 10 then then bit(7,busb)
end;
case champ(37) of
        '1':PSH(3):=bit(3,busb);
        '2':PSH(3):=if OU then bit(3,busb)
end;
if champ(38)='1' then PSH(4):=bit(4,busb);
case champ(39) of
        '1':PSH(5):=bit(5,busb);
        '2':PSH(5):=if RI(5,1) = 00 then bit(5,busb)
end;
if champ(40)='1' then PSH(6):=bit(6,busb);
```

```
case champ(41) of
        '1':PSH(7):=if RI(4)='1' then bit(7,busb);
        '2':PSH(7):=if RI(4)='0' then bit(7,busb);
        '3':PSH(7):=bit(7,busb)
end;
if champ(42)='1' then STH20:=busb mod 8;
if champ(43)='1' then STH(3):=bit(3,busb);
if champ(44)='1' then STH74:=busb div 16;
if champ(45)='1' then TSEL:=case RI(7:4) of;
if champ(46)='1' then CSEL:=case RI(7:4) of;
if champ(47)='1' then ETCI:=bit(0,busb);
if champ(48)='1' then EI:=bit(1,busb);
```

ANNEXE 4: MACSIM

(Langage de simulation fonctionnelle)

MACSIM (MAChine SIMulator) est un outil d'aide à la conception élaboré par l'équipe d'Architecture d'Ordinateur de l'IMAG (SCH85).

Il s'agit d'un langage de description d'algorithme d'interprétation associé à des programmes de simulation qui permet de caractériser le chemin de données et la partie contrôle de toute machine de type microprocesseur.

Sa mise en oeuvre comprend les étapes suivantes:

- description MACSIM de l'algorithme d'interprétation,
- simulation fonctionnelle,
- analyse structurale.

a) description MACSIM de l'algorithme d'interprétation

L'algorithme d'interprétation générant les séquences est écrit sous la forme de programme "PASCAL"; les instructions se décomposent en cycles qui spécifient les actions.

Le début de chaque instruction est repéré par le mot clef "procédure<nom de l'instruction>" associé au code opération "CODOP" qui lui correspond; le mot clef "CYCLE", suivi d'un paramètre indiquant le numéro du cyle, caractérise les séquences d'exécution. "FININST" indique la fin de l'instruction courante.

Exemple:

```
procédure(nom de l'instruction)
CODOP;
      CYCLE(1);
        action(s) à effectuer
      CYCLE(2);
        action(s) à effectuer
      CYCLE(3);
        action(s) à effectuer
      .
      .
      CYCLE(n);
        action(s) à effectuer
FININST; end;
```

Syntaxe de description d'une instruction

Les affectations (ex: X (--a:= Y) qui caractérisent une séquence sont décrites en termes d'identificateurs qui prennent en compte le chemin de données.

exemple:

```
busa:= Y; X:= busa; * affecter à "X" la valeur "Y"   *
Z:= busb mod 8;     * affecter à la variable "Z" les *
                    * trois bits de poids faibles du *
                    * bus "bêta" *
```

b) Simulation fonctionnelle

L'exécution du programme "MACSIM" permet de simuler l'algorithme d'interprétation en validant:

- la syntaxe de la description
- le bon enchainement des séquences,
- la fonctionnalité des opérations requises.

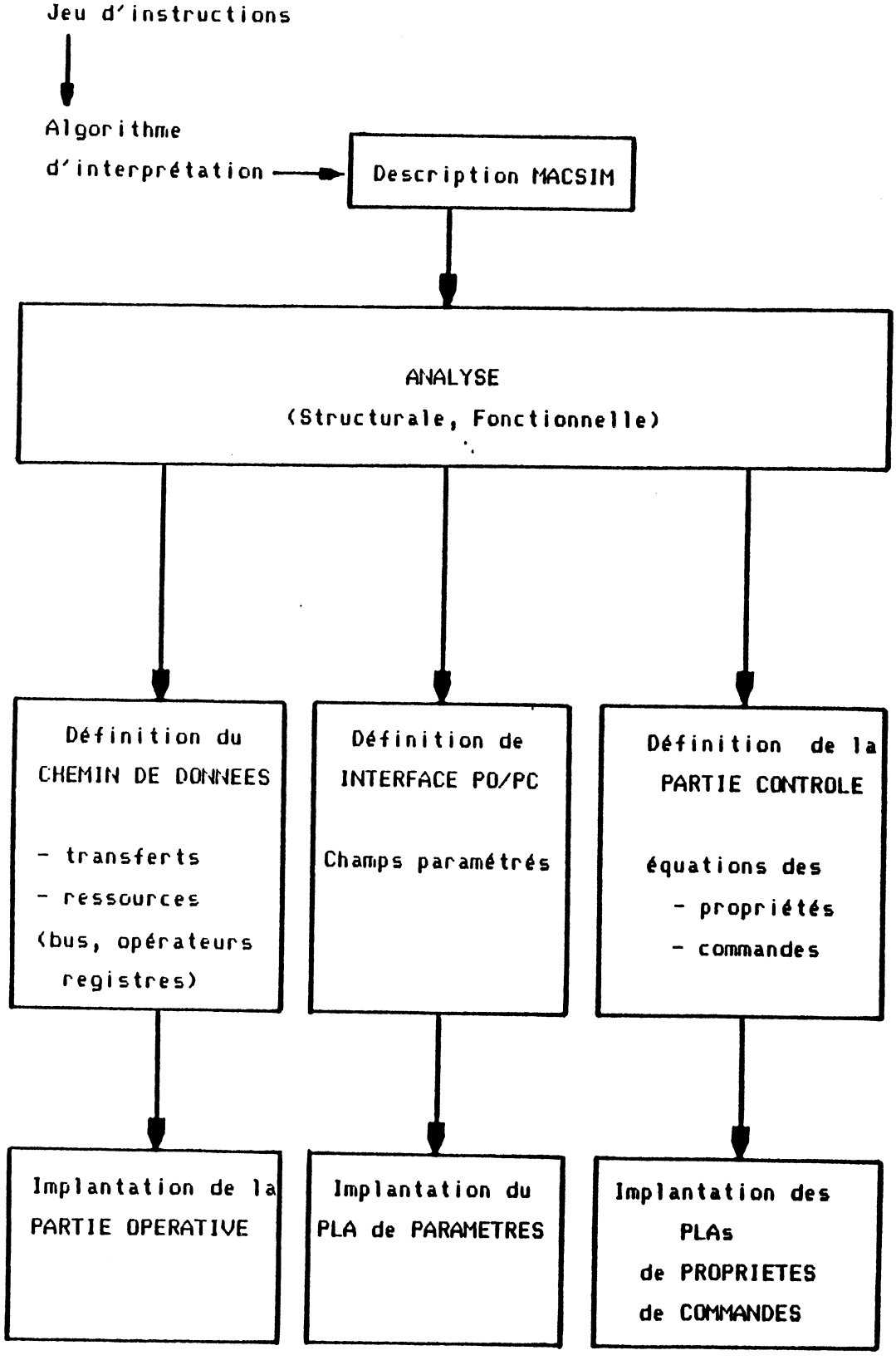
Son déroulement s'éffectue cycle à cycle ou instruction par

instruction; le concepteur peut à tout instant connaître l'état d'une variable et/ou lui affecter une valeur.

c) Analyse structurale

L'analyse de la description "MACSIM" permet de dégager les propriétés relatives à chaque instruction en vue de caractériser:

- la structure du chemin de données (bilan des transferts, ressources nécessaires...etc),
- la partie contrôle (exécution des champs paramétrés, génération des équations des paramètres et des commandes).



Organigramme d'exécution MACSIM

ANNEXE 5: MSINC

(Simulateur électrique)

"MSINC" est un programme développé par "STANFORD UNIVERSITY" qui permet d'analyser le comportement électrique d'un réseau; il s'agit d'un simulateur à modèle enregistré tel que "SPICE" car on ne peut modifier les équations de calcul comme dans des logiciels du type "IMAG3" ou "ASTEC3".

1) Description

Une description "MSINC" comporte une succession de cartes relatives à l'une des cinq catégories suivantes:

a) la carte titre

c'est la première et elle ne sert qu'à nommer le programme

b) les cartes commentaires

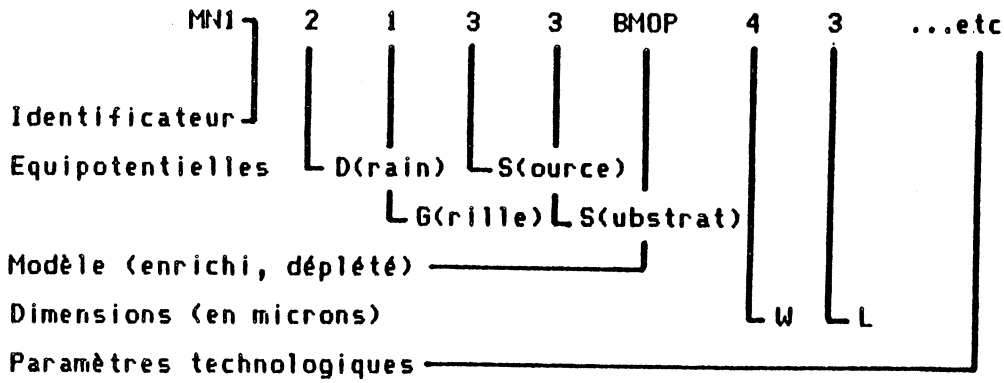
débutent par le caractère "*"

c) Les cartes ressources

elles nomment les différents composants - transistors, capacités, résistances, selfs, diodes - qui composent le réseau.

La description fait apparaître les connexions ainsi que les valeurs caractéristiques relatives à chaque élement.

exemple: description d'un transistor



d) Les cartes modèles

caractérisent le comportement des transistors (tension de seuil, capacité/ unité de surface, mobilité des porteurs...etc) en fonction de valeurs technologiques.

exemple: Description d'un transistor de type "N"

```
MODEL BMON NMO VTO=0.9 UB=600 ,1,0.03,/3.5E4 TOX=470
+DNB=5E15 CJUN=2.5E-4 XW=0.5 XJ=0.45 XN=0.5 GDS=3
```

e) les cartes de controle

fixent:

- les valeurs des équipotentielles,
- tensions d'alimentation "Vcc",
- sources de tension "V" ou de courant "I"

représentant des signaux de formes impulsionnelles, exponentielles, sinusoidales ou en escaliers,

- les conditions initiales (ex: charges d'une capacité),
- les échelles de temps: pas de simulation,

durée de l'exploration,

- les variables à visualiser,

etc ...

Des options sont prises par défaut.

2) Exemple de description et de simulation sur le point mémoire

On veut étudier le comportement électrique - seuil de basculement, temps de réponse - du point mémoire.(fig A5.1)

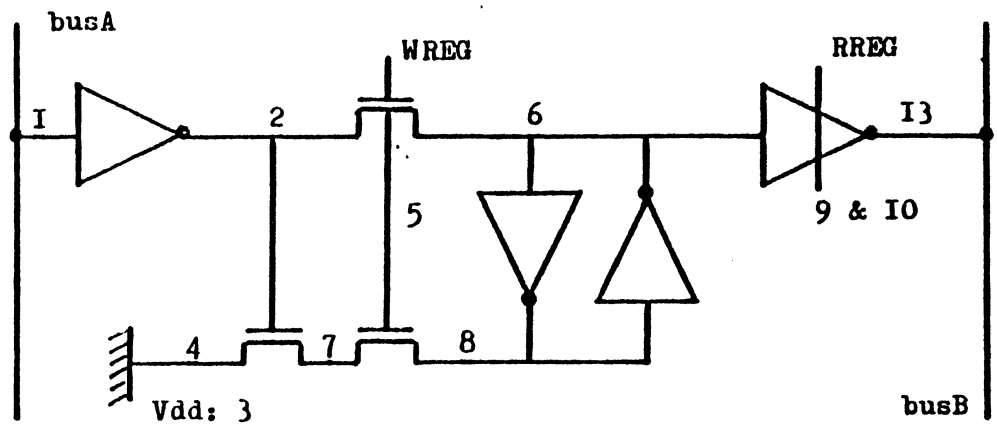


fig:A5.1 Schéma logique du point mémoire

La description "MSINC" de la cellule fait suite au schéma électrique

" x " définition des noeuds d'intéraction

"W/L" largeur et longueur du transistor.

L'élément de charge est constitué par la résistance et la capacité que représente le bus "busB".

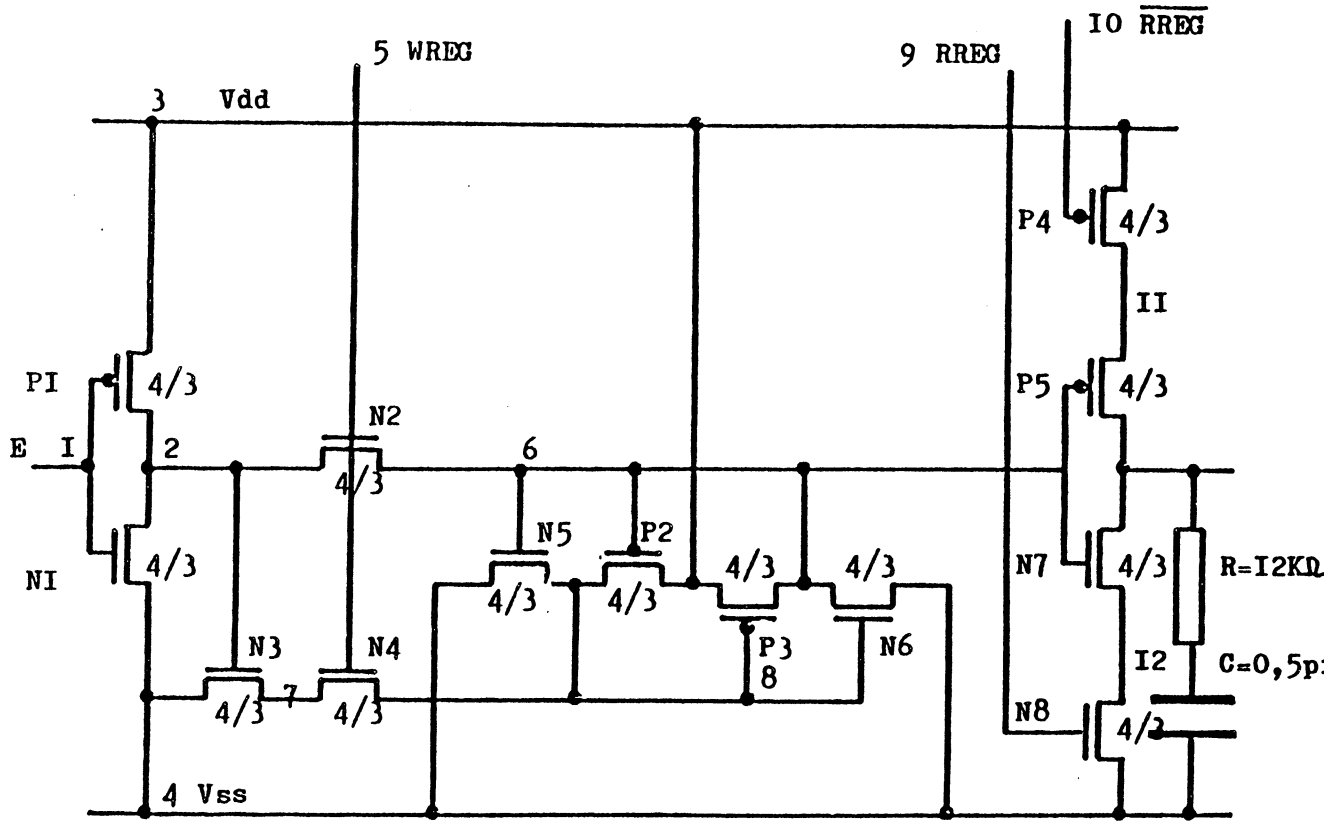


fig:A5.2 Représentation électrique du point mémoire (faisant apparaître les noeuds d'équipotentielles et les dimensions des transistors)

```
*                    POINT MEMOIRE                                    *
*********************************************************************
THLE 5N 200N                                                          *
*********************************************************************
*                                                                     *
*                                                                     *
*                         TRANSISTORS P                               *
*                                                                     *
MP1   3   1   2   3   BMOP   4   3   0.4   0.4   32   32
MP2   3   6   8   3   BMOP   4   3   0.4   0.4   32   32
MP3   3   8   6   3   BMOP   4   3   0.4   0.4   32   32
MP4   3  10  11   3   BMOP   4   3   0.4   0.4   32   32
MP5  11   6  13   3   BMOP   4   3   0.4   0.4   32   32
*                                                                     *
*                         TRANSISTORS N                               *
*                                                                     *
MN1   2   1   4   0   BMON   4   3   0   0   32   32
MN2   2   5   6   0   BMON   4   3   0   0   32   32
MN3   7   2   4   0   BMON   4   3   0   0   32   32
MN4   8   5   7   0   BMON   4   3   0   0   32   32
MN5   8   6   4   0   BMON   4   3   0   0   32   32
MN6   6   8   4   0   BMON   4   3   0   0   32   32
MN7  13   6  12   0   BMON   4   3   0   0   32   32
MN8  12   9   4   0   BMON   4   3   0   0   32   32
*********************************************************************
*                         CAPACITE                                    *
C 14   0   0.5
*                         RESISTANCE                                  *
R 13 14   12K
*********************************************************************
*                         ALIMENTATIONS                               *
VCC     3   4   5
VE      1   4   0   0N   0    5N   5    10N   5    90N   0    94N   0   200N
VVREG   5   4   0   0N   0   20N   5    25N   5    90N   0    95N   0   200N
VRREG   9   4   0   0N   0  100N   5   105N   5   160N   0   165N   0   200N
VRBAR  10   4   5   0N   5  100N   0   105N   0   160N   5   165N   5   200N
*********************************************************************
*                         CONTROLE                                    *
```

```
OPT NP
PLOT  5  1  6
PLOT  9 13  8
END
**************************************************************
```

fig:A5.3 Description MSINC du point mémoire

Le résultat de la simulation est disponible sur écran graphique ou tracé papier.

fig:A5.4a Simulation électrique du point mémoire a)

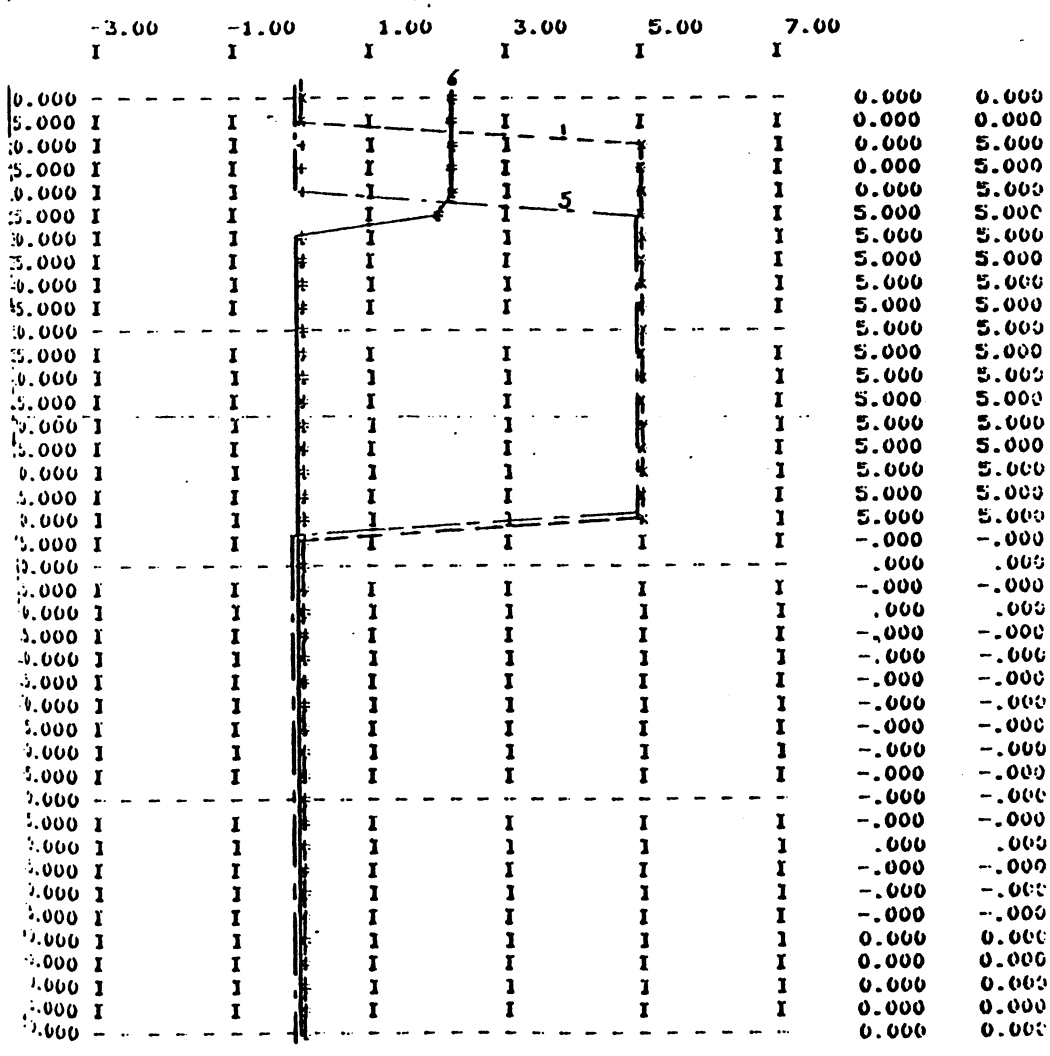
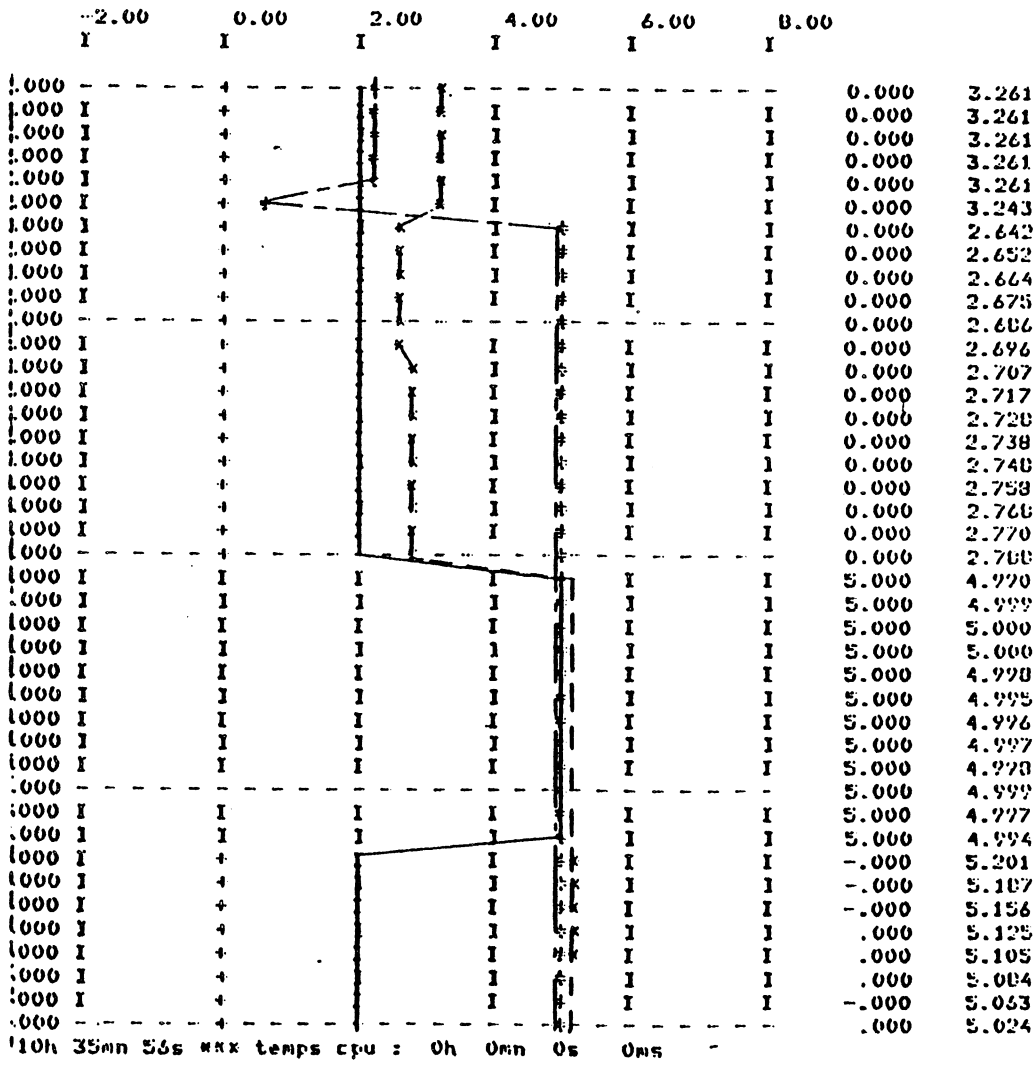


fig:A5.4b Simulation électrique du point mémoire b)



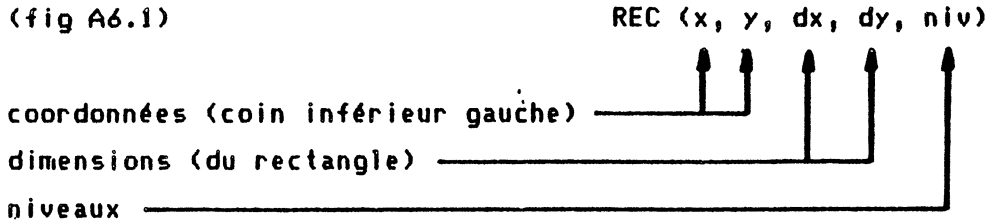
ANNEXE 6: LUCIE

(Langage Universitaire de Conception de Circuit Intégrés pour l'Enseignement)

Le système "LUCIE" (LUC81) est constitué d'un langage de description et d'un ensemble de programmes associés permettant de définir et de manipuler les masques de circuits intégrés.

1) Le langage "LUCIE"

L'élément de base du langage est le rectangle dont la syntaxe est la suivante:
(fig A6.1)



Le 80C48 est réalisé dans une technologie CMOS à caisson "P" faisant appel à six niveaux de masque:

"MD"; desoxydation N+
"MB"; désoxydation P+
"MP"; polysilicium
"MM"; métal
"MC"; contact
"MJ"; caisson P

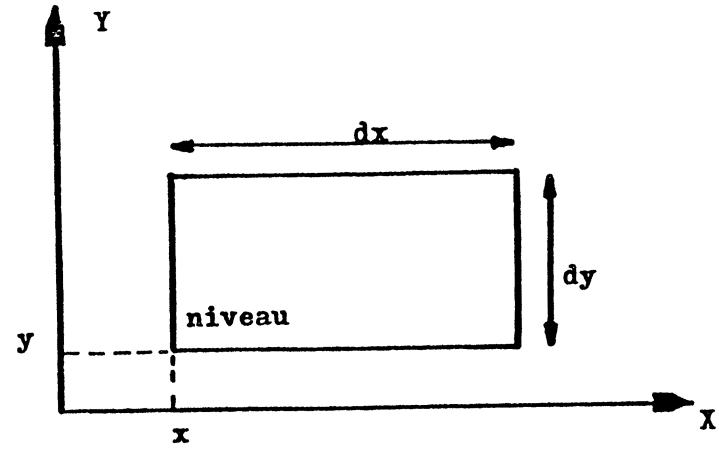


fig:A6.1 Description d'un rectangle en "LUCIE"

Une figure est définie comme étant une association d'éléments de base (rectangles) et/ou d'autres figures; on lui affecte un nom qui permet de la référencer dans une description de figure plus englobante. Toute description "LUCIE" est ainsi ramenée, par imbrications successives, à une suite de rectangles.

Syntaxe de génération, de déclaration et d'appel de figures:

- Génération de figure:

syntaxe: FIG <nom de la figure> <génération de figure>
rec(x, y, dx, dy, niv)
•
•
• <description sous forme d'une suite de
• rectangles ou de figures>
•
•
FFIG <identificateur de fin de figure>

- Déclaration de figure:

syntaxe: FIG <nom de la figure> (x, y)

La première occurrence d'une figure constitue sa déclaration, les autres, son placement aux coordonnées (x, y) indiquées. Une déclaration de figure suivie de coordonnées permet de placer cette figure en un lieu déterminé.

Une description est constituée par des imbrications successives; chaque figure est accessible, suite à sa déclaration, pour un placement ultérieur.

- Appel de figure externe:

syntaxe: FIGEXT (nom de la figure) (x,y)

Un appel de figure externe permet d'utiliser une figure précédemment établie - et traduite - dans une nouvelle description.

2) Les opérateurs de figures

Trois opérateurs permettent de manipuler les figures; ils réalisent des fonctions de répétition, de symétrie et de rotation. On associe à chaque opérateur une paire de mot-clés de début et de fin de procédure.

syntaxe:

REP ..... (suite d'éléments..... FREP (répétition)
SYM .... - rectangles/ figures - ..... FSYM (symétrie)
ROT .....affectés par l'opérateur)..... FROT (rotation)

- la fonction de répétition

syntaxe:

REP (n, x, dx) (répéter "n" fois suivant l'axe des "x" avec un pas de "dx")
REP (n, y, dy) (répéter "n" fois suivant l'axe des "y" avec un pas de "dy")

L'opérateur de répétition permet de dupliquer "n" fois suivant l'axe des "x" ou des "y" une description - figure/ rectangle - enregistrée. (fig A6.2a)

- la fonction de symétrie

syntaxe:

SYM (x, dx) (symétrie selon l'axe des "x")
SYM (y, dy) (symétrie selon l'axe des "y")

L'opérateur de symétrie permet de générer la "symétrie miroir" d'une figure, les variables "dx" et "dy" spécifient son encombrement. (fig A6.2b)

- la fonction de rotation

syntaxe:

ROT(+, dx) (rotation dans le sens direct avec translation de "dx")

ROT(-, dy) (rotation dans le sens indirect avec translation de "dy")

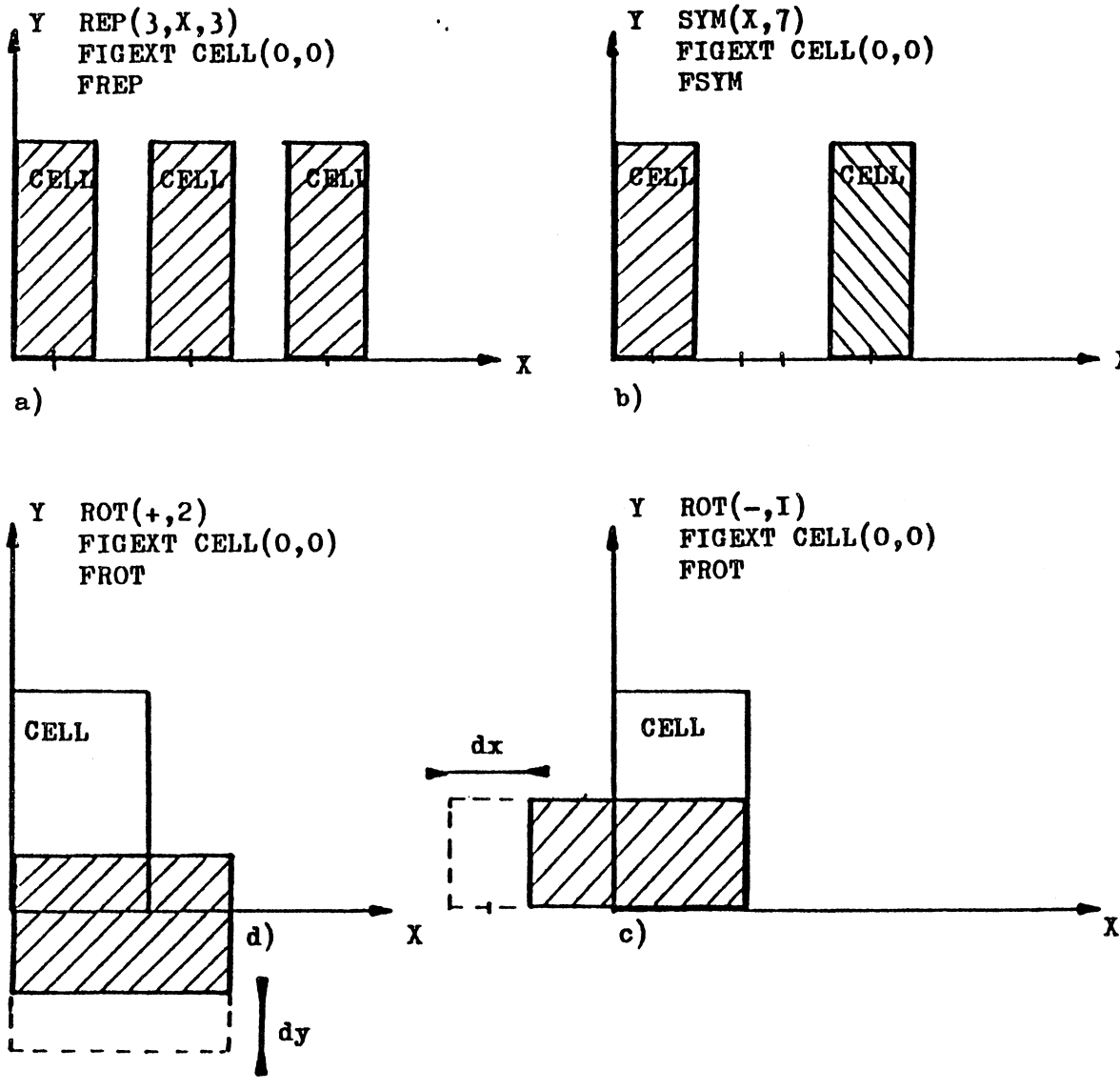
L'opérateur de rotation permet de faire tourner une figure dans le sens direct (ou indirect) par rapport à une abscisse absolue et lui appliquer une translation de "dx" (ou "dy") selon l'axe considéré. (fig A6.2c & A6.2d).

fig:A6.2 Fonctions réalisées par les opérateurs de "LUCIE"

a) répétition/ b) symétrie

c) rotation dans le sens direct

d) rotation dans le sens indirect



3) L'éditeur graphique

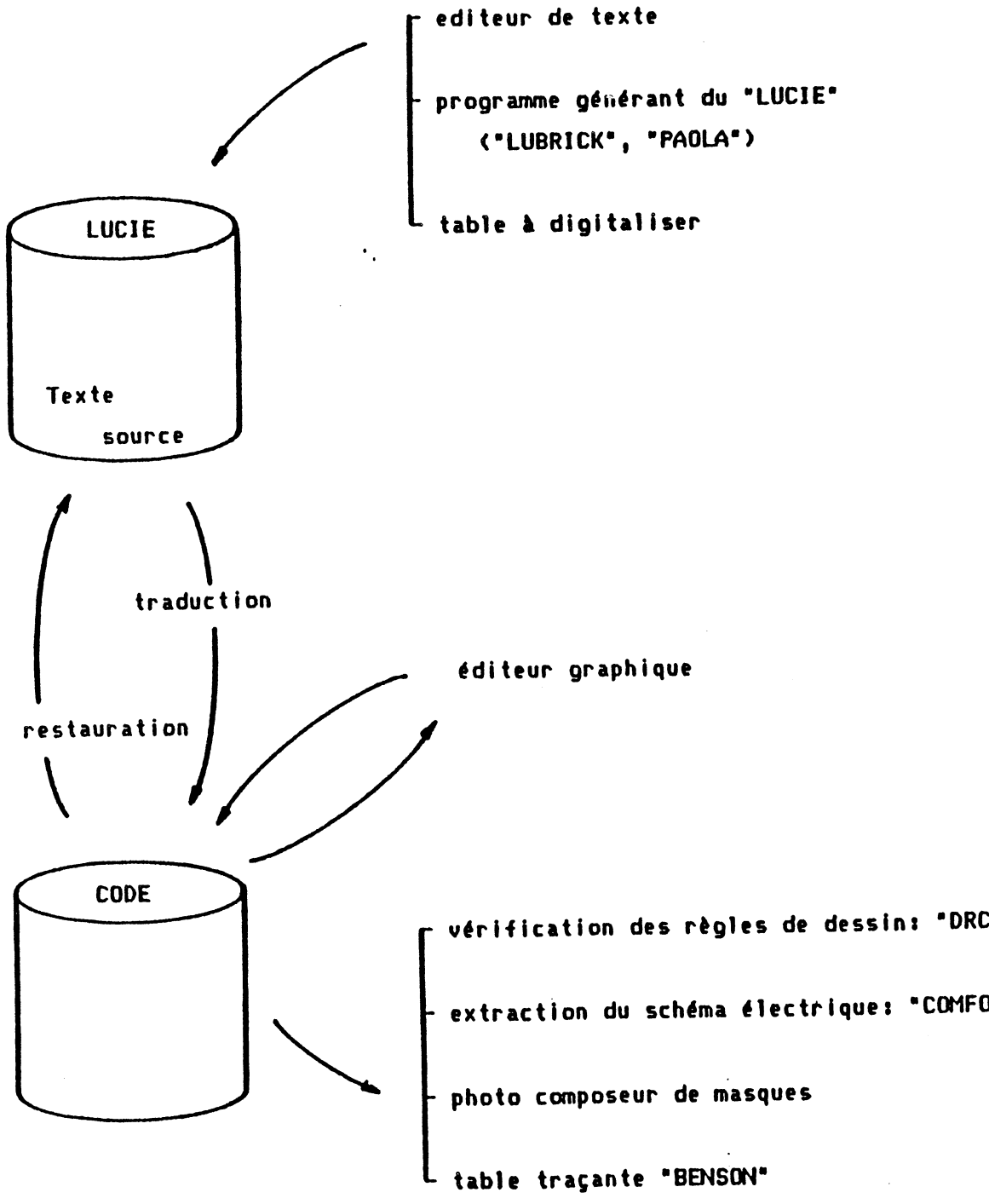
L'éditeur graphique permet de tracer sur un écran graphique tout ou partie d'un dessin décrit - et traduit - en "LUCIE"; un réticule permet alors de pointer et de modifier de façon intéractive la description de cette figure (ajout, suppression, modification de rectangles).

D'autres fonctions associées à la description permettent:

- de visualiser une figure avec un niveau d'imbrication déterminé,
- d'afficher une partie de figure (effet de "zoom"),
- de faire apparaître l'encombrement d'un figure (pour un niveau d'imbrication donné),
- de choisir le niveau de masque souhaité
- .
- .
- etc..

Le concepteur dispose d'une possibilité d'édition graphique sur papier - table de dessin - avec toutes les options habituelles: choix de l'échelle, de la couleur associée au niveau de masque, etc.

fig:A6.3 Environnement de "LUCIE"



ANNEXE 7: VERDICT

(Vérification de règles de dessin)

"VERDICT" permet, sur un texte source "LUCIE" et d'après la description des paramètres technologiques, de vérifier les règles de dessin d'une implantation au micron, à savoir, le respect des dimensions minimales et des règles de garde.

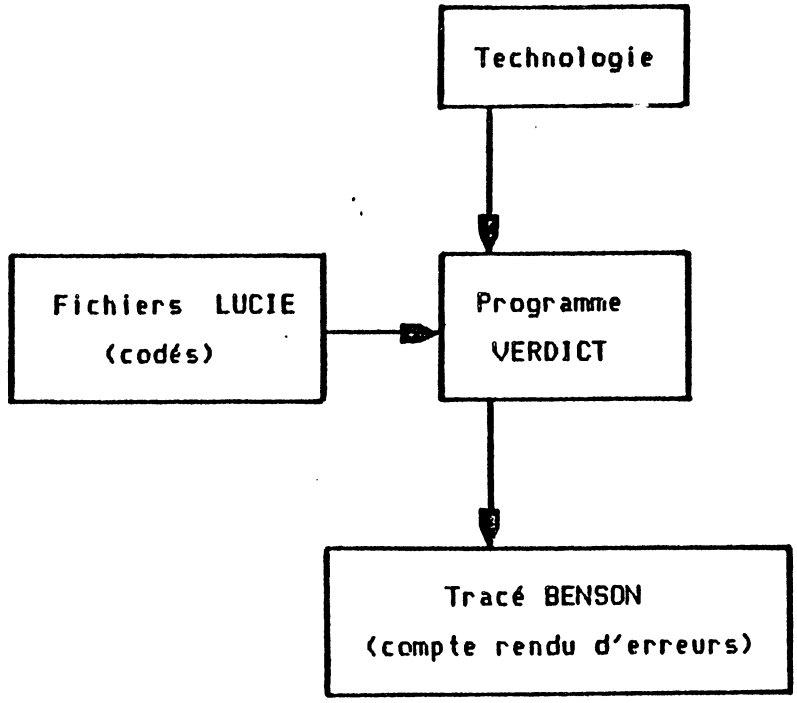


fig:A7 VERDICT (Vérificateur de règles de dessin)

ANNEXE 8: COMFOR

(Extracteur de schéma électrique)

"COMFOR" (JER83) permet d'extraire le schéma électrique d'une description "LUCIE" paramétrée par la technologie; le compte rendu s'effectue sous forme de réseau de transistors interconnectés; il ouvre la voie à des possibilités de simulations électriques ou de dessins logiques.

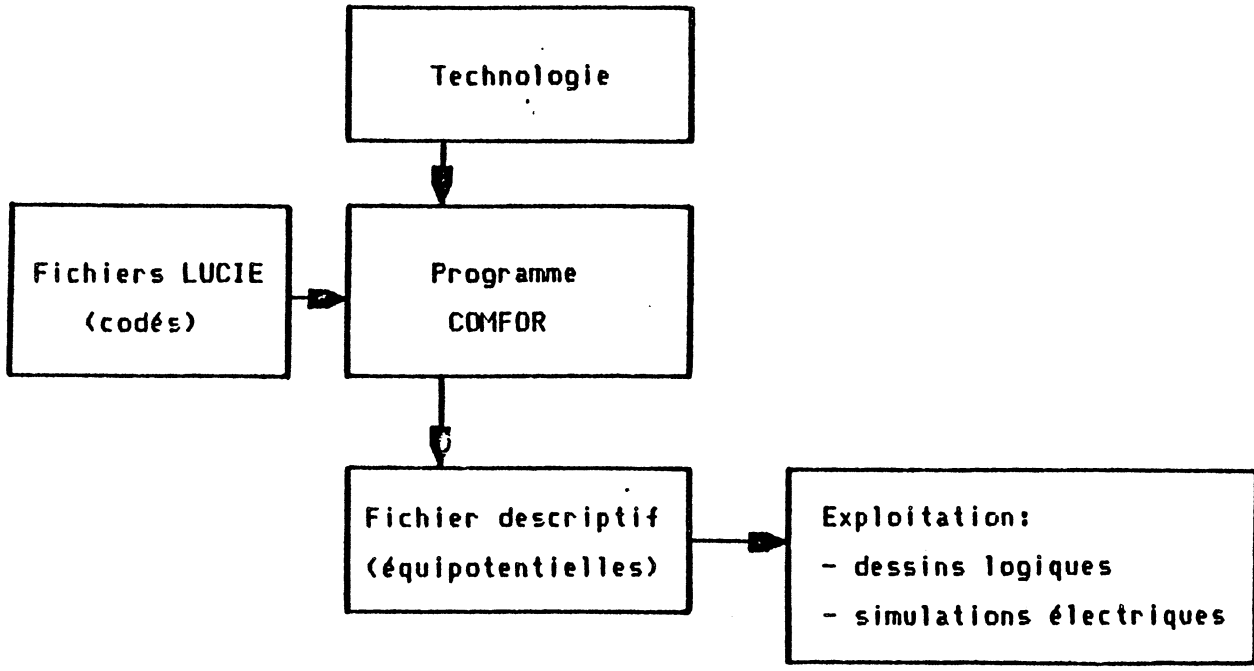


fig:A8 COMFOR (Extracteur de schéma électrique)

ANNEXE 9: LUBRICK

(Assembleur de silicium)

Le système "LUBRICK" permet l'assemblage hiérarchique de cellules "LUCIE" par programme ainsi que le tracé automatique des interconnexions. (SH083)

C'est un programme "PASCAL" dans lequel l'imbrication des fonctions est mise en correspondance avec l'imbrication de la description "LUCIE" des figures.

L'élément de base de l'assembleur est la cellule; il s'agit d'une "boîte virtuelle" englobée par la figure "LUCIE" et dont la description s'explicite en termes de frontières et de connecteurs.

Une cellule est hiérarchisée en ce sens que la description d'une fonction complexe fait appel à des cellules plus élémentaires ou de niveau inférieur.

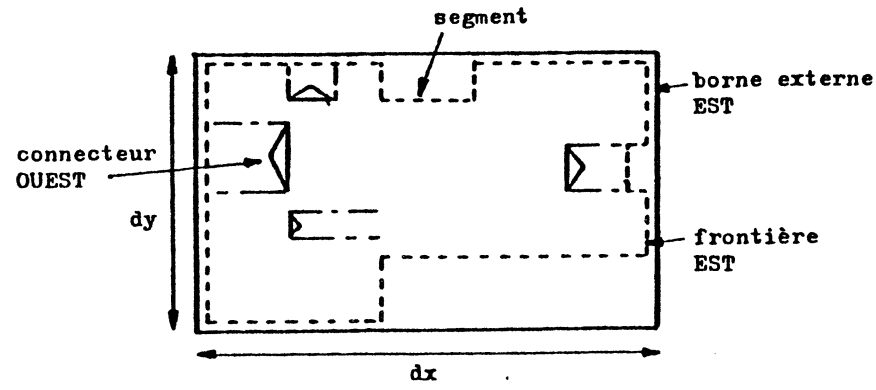


fig:A9.1 Cellule "LUBRICK"

Une description "LUBRICK" ne met pas en oeuvre les coordonnées cartésiennes utilisées par "LUCIE"; le positionnement relatif des cellules est réalisé par rapport aux limites extérieures - NORD, SUD, EST, OUEST - de la figure précédente. L'intérêt d'un tel choix réside dans le fait que la description est paramétrée; la suppression (ou l'ajout) d'une brique ne remet pas en cause les valeurs numériques précédement établies.

1) Notion de frontière

On réalise le placement automatique des figures les unes par rapport aux autres en introduisant la notion de frontière - NORD, SUD, EST, OUEST

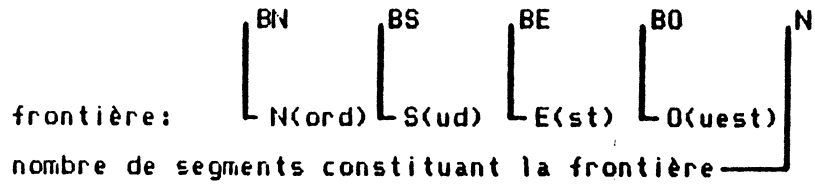
-. Il s'agit d'un périmètre constitué de segments et dont le tracé ne correspond pas forcément aux limites extérieures de la figure "LUCIE". (fig A9.1)

Cette formulation permet d'obtenir une distance inter-cellule minimale tout en respectant les règles de gardes spécifiées par la technologie.(fig A9.4 & A9.5)

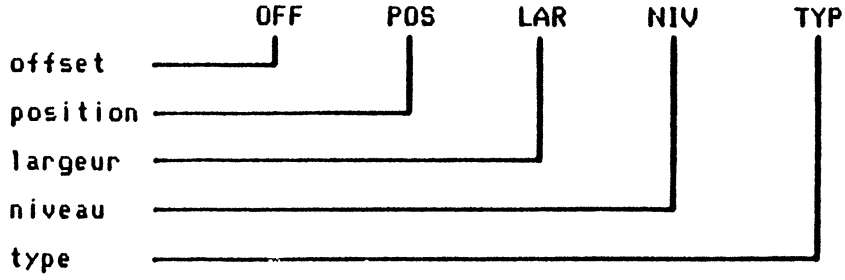
On note que:

- une frontière ne peut dépasser les limites extérieures d'une figure "LUCIE",
- une frontière est déterminée par rapport à un type correspondant au niveau du masque considéré.

syntaxe:



Chaque segment composant une frontière est décrit par un ensemble de cinq paramètres ordonnés qui définissent: sa nature, sa dimension et sa position relative au bornes de la cellule considérée.(fig A9.2)



(pour les connecteurs seulement)

Description:

BN 1

2 3 2 4

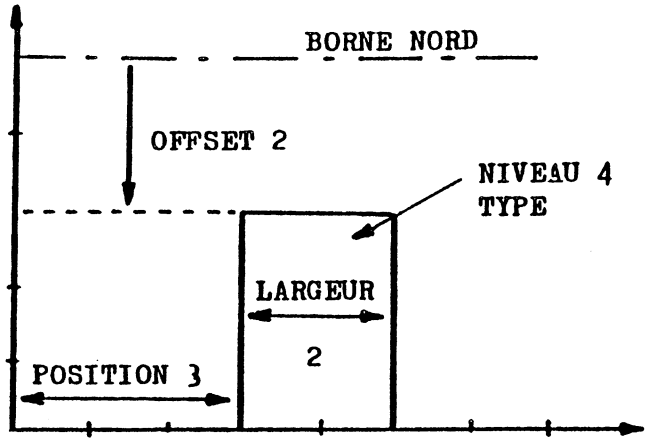


fig:A9.2 Exemple de description d'un segment composant une frontière NORD

2) Notion de connecteurs

Les connecteurs permettent de générer les interconnexions entre cellules; il s'agit de segment(s) orienté(s) par rapport à l'une des quatres frontières de la cellule représentant une connexion éventuelle de la cellule avec l'extérieur.

Une connexion est réalisable lorsque deux connecteurs de même niveau et d'orientations opposées (nord/sud ou est/ouest) ont des positions compatibles.

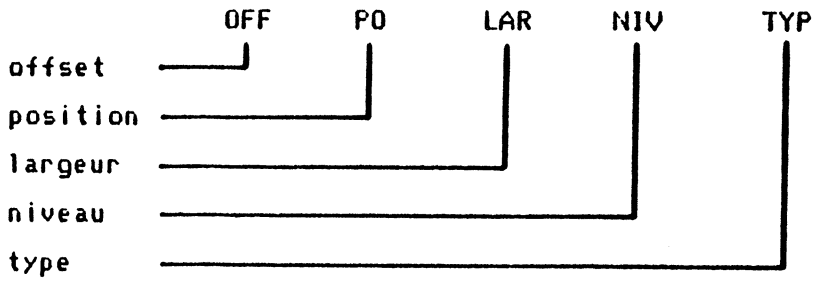
On peut ainsi générer une nouvelle figure "LUBRICK" comportant elle aussi un fichier de description de frontières et de connecteurs; ces frontières sont calculées automatiquement à partir des anciennes frontières et les connecteurs sont ceux non utilisés avec leurs coordonnées définies par rapport aux nouvelles limites de la cellule. Le programme effectue des réductions parmi les connecteurs lorsque des occurrences identiques apparaissent dans une description.

syntaxe:

connecteurs: CN N(ord) CS S(ud) CE E(st) CO O(uest) N

nombre de connecteurs constituant la frontière

On définit un connecteur de cellule "LUBRICK" par une suite de cinq paramètres ordonnés qui définissent sa nature, sa dimension ainsi que sa position relative par rapport aux frontières de la dite cellule.(fig A9.3)



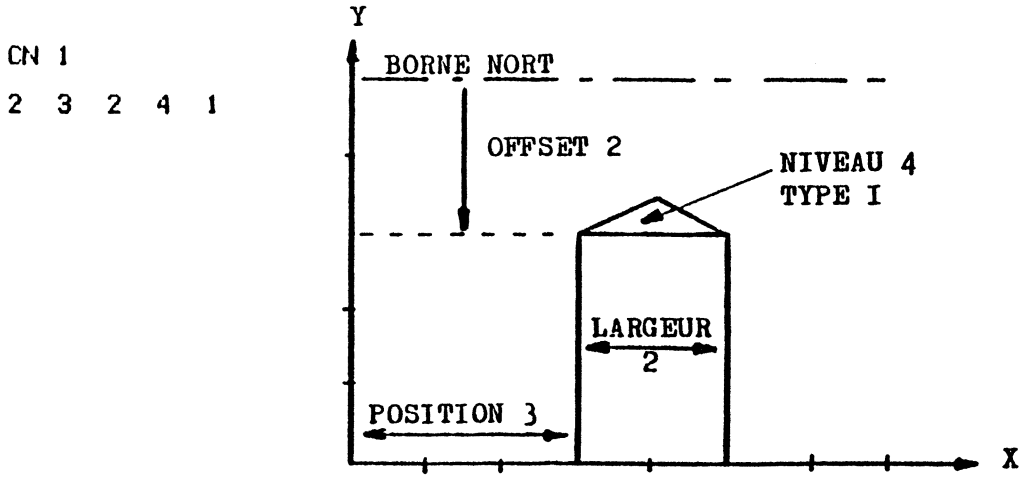


fig:A9.3 Exemple de description de connecteur NORD

3) Description "LUBRICK"

La description d'une figure manipulable par "LUBRICK" comporte:

- ses dimensions "dx" et "dy",

- la définition de ses frontières,
- la liste des connecteurs.

Pour l'utilisateur, une figure comporte:

- un fichier "LUCIE",
- la liste des frontières et des connexions dans un fichier dont la syntaxe est la suivante

```
FIG <nom du fichier>
  dx dy
  <identificateur de frontières>  N
          .
          <offset_position_largeur_niveau_type>
          .
  <identificateur de connexions>
          .
          <offset_position_largeur_niveau_type>
          .
FFIG
```

4) Fonctions et procédure

On dispose des procédures suivantes:

OPENFIG (ouverture de figure)
CLOSEFIG (fermeture de figure)
GETFIG (accès à une figure externe)

Il est important de remarquer qu'une figure n'est manipulable que fermée.

De plus, "LUBRICK" permet de disposer de fonctions d'assemblage (en partant du fait qu'on ne peut ajouter une figure qu'au dessus ou à droite d'une figure déjà existante).

a) la fonction "UP":

syntaxe: UP <cell j, cell K, n fois, offset m>

La fonction a pour effet de placer "n" fois la cellule "K" au dessus de la cellule "j" après une translation de "m".

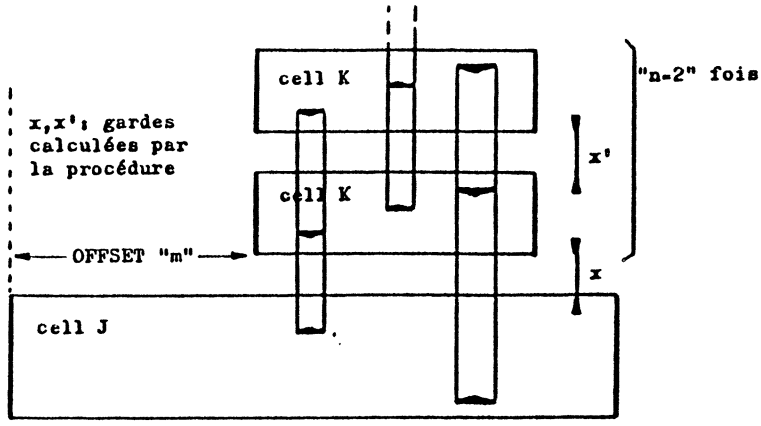


fig:A9.4 Exemple de réalisation de la fonction UP

b) la fonction RIGHT:

syntaxe: RIGHT <cell j, cell K, n fois, offset m>

La fonction a pour effet de placer "n" fois la cellule "K" à droite de la cellule "K" après avoir opéré une translation de "m".

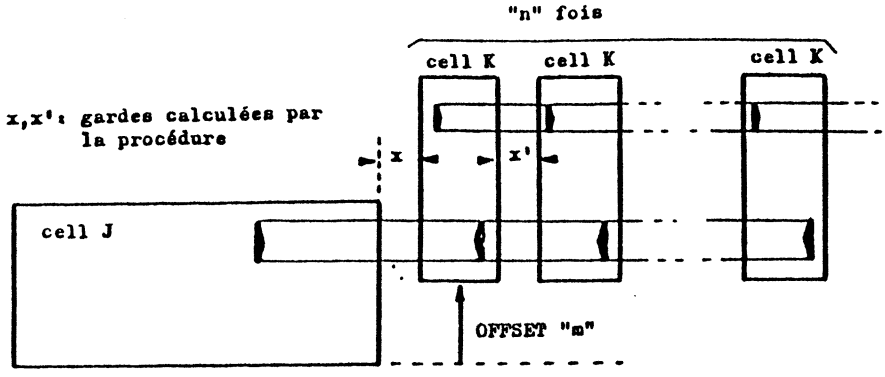


fig:A9.5 Exemple de réalisation de la fonction RIGHT

c) la fonction "CUP":

syntaxe: CUP(cell j, cell k, offset gauche l, offset droit r,
tableau d'interconnexions t)

La procédure "CUP" permet d'interconnecter deux blocs fonctionnels par une fonction de routage, les liaisons pouvant être des lignes brisées.
Un tableau de correspondance précise le type de chaque connexion.

Une procédure analogue "CRUP" réalise une fonction similaire tout en autorisant le croisement des interconnexions ("channel routing").

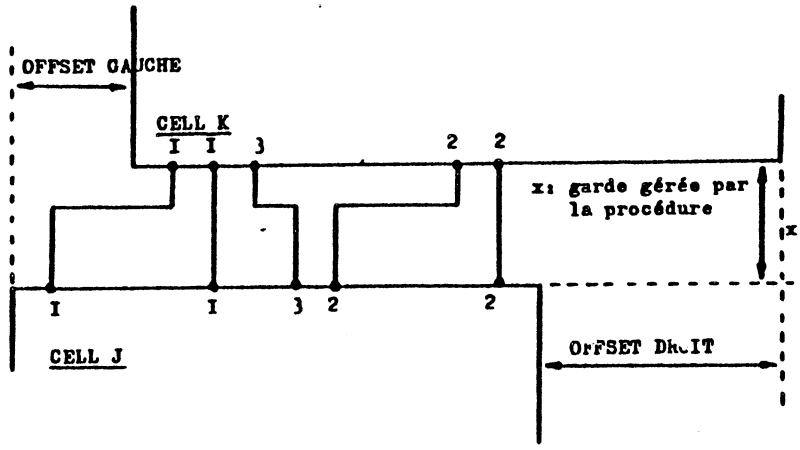


fig:A9.6 Exemple de réalisation de la fonction "CUP"

ANNEXE 10: PAOLA

(Optimisateur de PLAs)

"PAOLA" ("Pla Automatic Optimisation LAyout") est un système destiné à l'optimisation topologique et à la génération automatique de la description des masques de PLAs complexes.(CHU84)

Un PLA ("Programmable Logic Array") est un dispositif mémoire programmé ou programmable qui permet la réalisation de circuits de type combinatoire et séquentiel. La logique combinatoire est réalisée à l'aide de fonctions booléennes en sommes de produits des variables d'entrées, ce qui impose la présence de deux matrices, "ET" pour le produit des entrées, "OU" pour la somme des termes (ou monômes) ainsi réalisés.

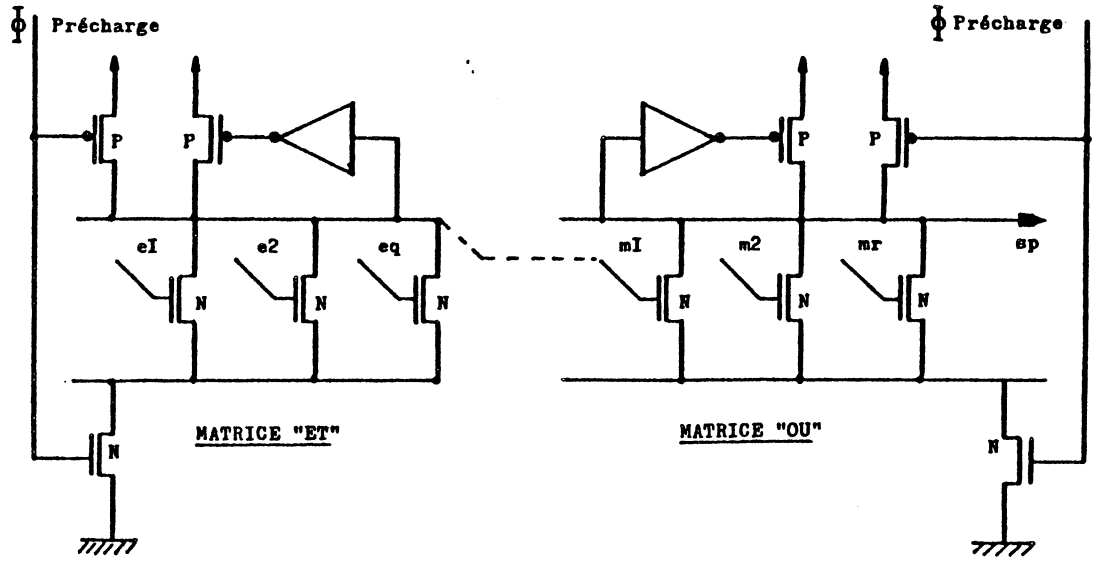


fig:A10.1 Structure classique de PLA

La réduction en surface de PLAs peut se faire par des techniques d'optimisation logique et topologique; "PAOLA" réalise une optimisation topologique par une réorganisation des matrices "ET" et "OU", tout en conservant le même nombre de transistors; les techniques d'optimisation logique agissent sur le nombre de monômes.

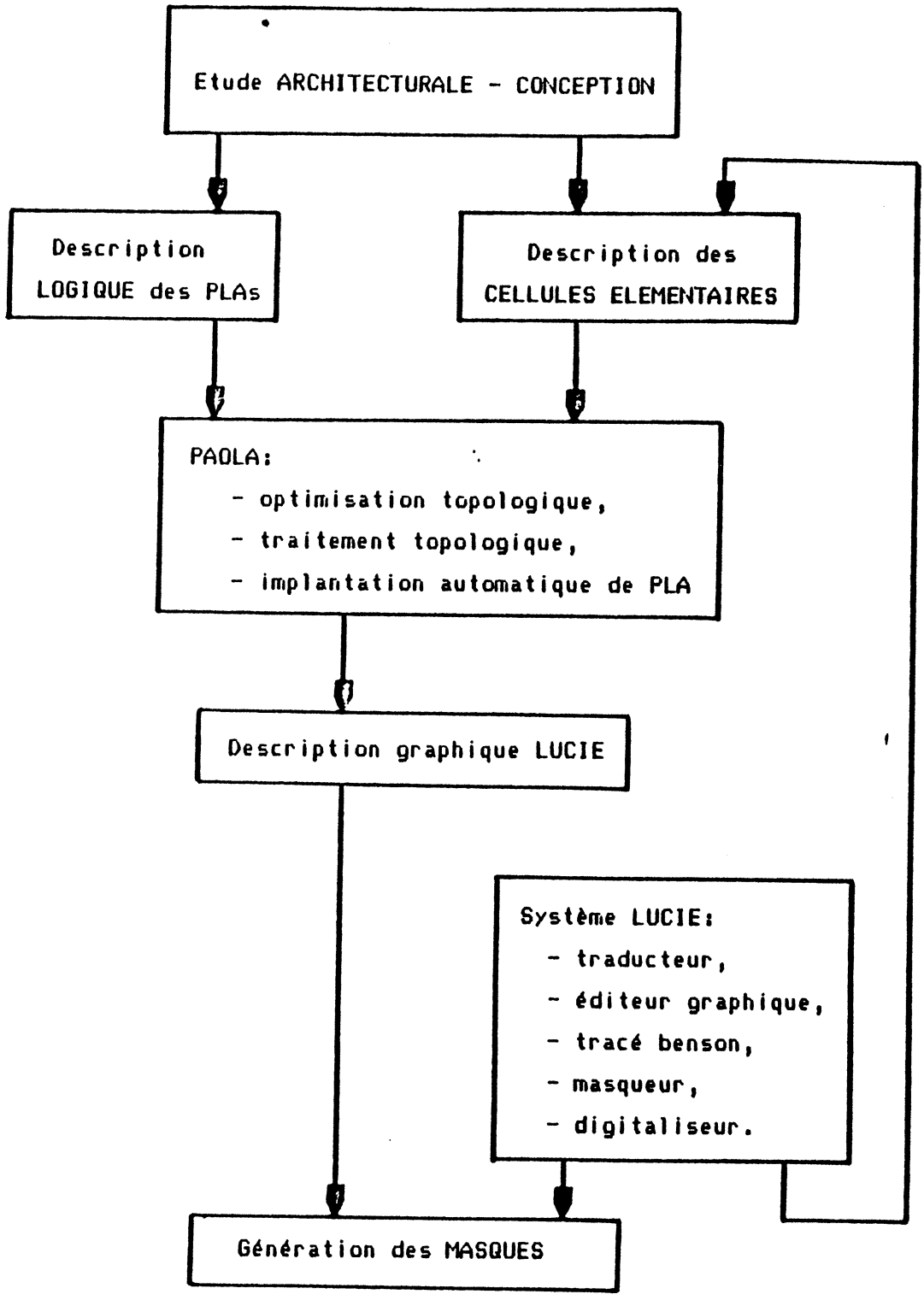


fig:A10.2 Environnement du système "PAOLA"

1) Optimisation topologique

L'implémentation de PLAs classiques (non optimisés) pose les problèmes suivants:

- l'implantation au pas minimal des lignes d'entrées/sorties nécessite l'emploi d'une zone importante d'interconnexions entre la matrice et le bloc contrôlé,
- la matrice "OU" bien que faiblement remplie, occupe une surface relativement importante.

On minimise la surface globale occupée par brisure des lignes d'entrées/sorties, (fig A10.3) - en disposant de plusieurs segments au niveau de chaque colonne -, et par une répartition des connexions sur toute la longueur de la matrice considérée. Cette optimisation topologique fournit une nouvelle structure topologique des matrices "ET" et "OU". Sa réalisation nécessite les trois étapes suivantes:

- réordonnancement des monômes,
- duplication des monômes,
- compactage des matrices.

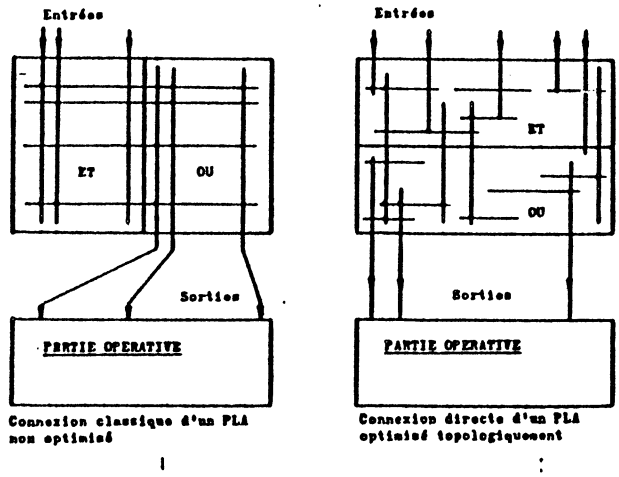


fig:A10.3 Connexion d'un PLA et d'un bloc (CHU84)

a) réordonnancement des monomes

On ordonne les monômes en minimisant la distance entre ceux qui sont

connectés à une même sortie; la matrice tend alors à se diagonaliser, présentant ainsi une configuration optimale.

b) duplication des monomes

La connexion directe d'un PLA au bloc qu'il contrôle peut nécessiter son alignement à la longueur de ce bloc.

Le système "PAOLA" profite de cette opportunuité pour améliorer la diagonalisation de la matrice "OU" par la duplication de certains monômes; on associe à la "partie père" du monôme "ET" chacun des monômes ainsi dédoublés.

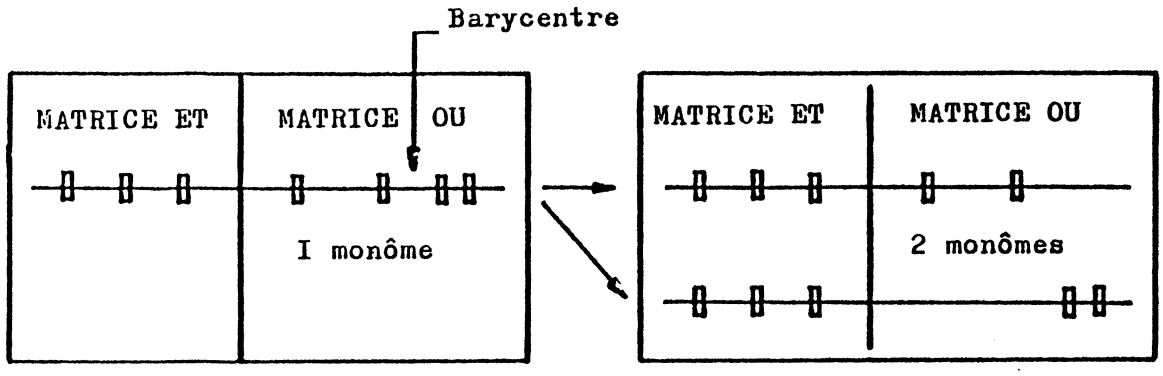


fig:A10.4 Duplication de monômes

Différents critères interviennent dans le choix des monômes à dupliquer, à savoir:

- le nombre de sorties contrôlées,
- la distance entre la première et la dernière sortie,
- la distance entre le barycentre du monôme et chacune de ses sorties.

c) compactage des matrices

On appelle segment d'entrée d'une matrice "ET", le segment de silicium polycristallin de la ligne d'entrée disposé entre le premier et le dernier monôme excité par cette entrée; de même, le segment de sortie

correspond à la ligne de métal située entre le premier et le dernier monôme qu'excite cette sortie.

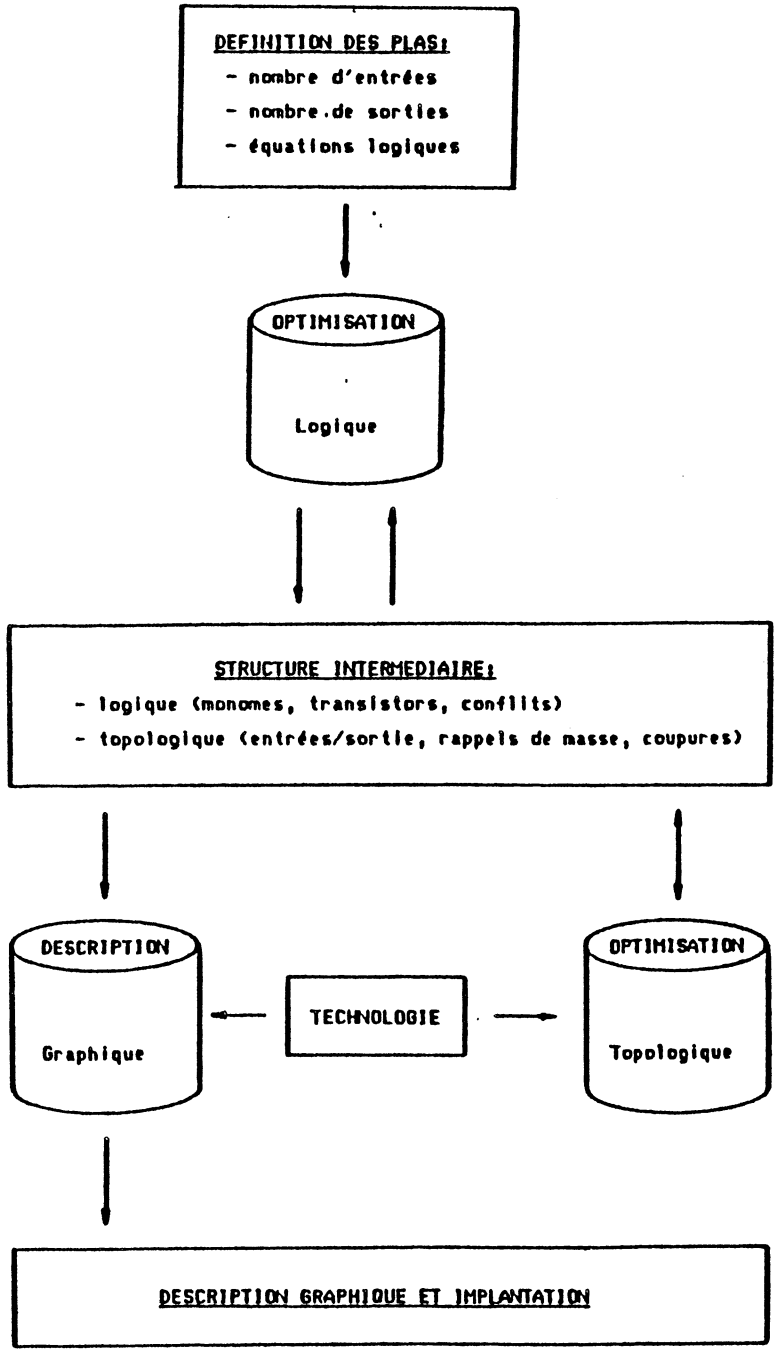
Le compactage d'une matrice consiste à placer plusieurs segments de sortie compatibles dans une même colonne appelée "niveau". Deux segments sont dits compatibles s'ils n'ont aucun recouvrement au sens booléen du terme. On réorganise les niveaux afin de simplifier les connexions des segments avec leurs sorties; l'ordonnancement est réalisé en fonction du nombre de segments de sorties ainsi que du nombre de transistors.

b) génération automatique du dessin des masques

On génère le dessin des masques d'un PLA optimisé par une validation des résultats obtenus par le traitement précédent et comportant l'information d'existence de transistors entre les monômes et les segments de sortie. On distingue à nouveau les trois étapes suivantes:

- la définition de l'organisation physique des matrices (placement des rappels de masse et des canaux d'écartement permettant de résoudre les conflits topologiques).
- le tracé des connexions entre les segments d'entrées/sorties et les bornes disposées sur le pourtour du PLA.
- la génération d'un fichier texte "LUCIE" dans un format reconnu par un système graphique de génération de masques; le programme utilise les paramètres de dessin et dimensionne les rectangles générés.

fig:A10.5 Diagramme d'intéraction des données et du processus



AUTORISATION de SOUTENANCE

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

U le rapport de présentation de Monsieur F. ANCEAU, Ingénieur de recherche

Monsieur BERTRAND François

st autorisé à présenter une thèse en soutenance en vue de l'obtention du titre de OCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 15 juillet 1985

Le Président de l'I.N.P.-G

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

RESUME:

La nécessité d'une conception sûre et descendante de circuits intégrés V.L.S.I. ("Very Large Scale Integration") est largement reconnue.

Nous nous intéressons ici aux premières étapes d'une telle conception à savoir le passage des spécifications initiales à la définition de l'architecture du circuit. La méthode proposée est une méthode de conception par affinements successifs de spécifications; on distingue trois choix fondamentaux dans la conception de circuits:

- le choix des algorithmes,
- le choix du chemin de données associé aux blocs fonctionnels,
- le choix de la structure de la partie contrôle.

Les validations partielles de conception se feront par analyse et simulation, l'optimisation des choix de conception repose en partie sur une évaluation prédictive.

MOTS-CLES:

Algorithme d'interprétation, niveaux d'interprétation, partie opérative, CMOS statique, méthode d'implantation, assemblage de cellules, conception de partie contrôle, PLA et générateur de temps, conception assistée par ordinateur.