Conception de microprocesseurs à haut rendement

Philippe Genestier

THESE

Présentée à

L'Institut National Polytechnique de Grenoble

pour obtenir le grade de

Docteur de l'INPG

spécialité "Microélectronique"

par

Philippe GENESTIER

CONCEPTION DE MICROPROCESSEURS A HAUT RENDEMENT

Thèse soutenue le 9 Juillet 1987 devant la commission d'examen :

Président :
G. MAZARE
Examinateurs :
R. GERBER
M. GLESNER
P. IVEY
G. SAUCIER
J. TRILHE

Thèse préparée au sein du **Laboratoire Circuits et Systèmes**

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH

Année 1987

Vice - Présidents : René CARRE
Jean-Marie PIERRARD

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	ENSEEG
BESSON Jean	ENSEEG
BLIMAN Samuel	ENSERG
BLOCH Daniel	ENSPG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BOUVARD Maurice	ENSHMG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
BUYLE-BODIN Maurice	ENSERG
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHERUY Arlette	ENSIEG
CHIAVERINA Jean	UFR PGP
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COUMES André	ENSERG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPORTES Jacques	ENSPG
DOLMAZON Jean-Mar	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI.Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GREVEN Hélène	IUFA
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOUBERT Pierre	ENSIEG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LOUCHET François	ENSEEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PARIAUD Jean-Charles	ENSEEG
PAUTHENET René	ENSIEG
PERRET René	ENSIEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
POUPOT Christian	ENSERG
SAUCIER Gabrielle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SERMET PIERRE	ENSERG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VEILLON Gérard	ENSIMAG
ZADWORNY François	ENSERG

Professeur Université des Sciences Sociales (Grenoble II)

BOLLIET Louis

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
PLA Fernand
GHIBAUDO Gérard
HAMAR Sylvaine
LADET Pierre
LATOMBE Claudine
LE GORREC Bernard
MADAR Roland
MULLER Jean
NGUYEN TRONG Bernadette
TCHUENTE Maurice
VINCENT Henri

Chercheurs du C.N.R.S

Directeurs de recherche 1ère Classe

CAILLET Marcel
CARRE René
FRUCHART Robert
JORRAND Philippe
LANDAU Ioan
MARTIN

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ANSARA Ibrahim
ARMAND Michel
BINDER Gilbert
BONNET Roland
BORNARD Guy
CALMET Jacques
DAVID René
DRIOLE Jean
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
JOUD Jean-Charles
KAMARINOS Georges
KLEITZ Michel
KOFMAN Walter
LEJEUNE Gérard
MERMET Jean
MUNIER Jacques
SENATEUR Jean-Pierre
SUERY Michel
TEDOSIU
WACK Bernard

Personnalités agrées à titre permanent à diri des travaux de recherche (décision du conseil scientifiqu

E.N.S.E.E.G

BERNARD Claude
CHATILLON Catherine
CHATILLON Christian
COULON Michel
DIARD Jean-Paul
FOSTER Panayotis
HAMMOU Abdelkader
MALMEJAC Yves
MARTIN GARIN Régina
SAINTFORT Paul
SARRAZIN Pierre
SIMON Jean-Paul
TOUZAIN Philippe
URBAIN Georges

E.N.S.E.R.G

BOREL Joseph
CHOVET Alain
DOLMAZON Jean-Marc
HERAULT Jeanny

E.N.S.I.E.G

DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.G

BOIS Daniel
DARVE Félix
MICHEL Jean-Marie
ROWE Alain
VAUCLIN Michel

E.N.S.I.M.A.G

BERT Didier
COURTIN Jacques
COURTOIS Bernard
DELLA DORA Jean
FONLUPT Jean
SIFAKIS Joseph

E.F.P.G

CHARUEL Robert

C.E.N.G

CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIB Maurice
VINCENDON Marc

Laboratoires extérieurs

C.N.E.T

DEMOULIN Eric
DEVINE
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

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 CATEGORIE

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 CATEGORIE

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

DIRECTEUR DE RECHERCHE

LESBATS Pierre	Métallurgie

MAITRE 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

Personalités habilitées à diriger des travaux de recherche

DRIVER Julian	Métallurgie
GUILHOT Bernard	Chimie
THOMAS Gérard	Chimie

Professeurs à l'UER de Sciences de Saint-Etienne

VERGNAUD Jean-Maurice	Chimie des Matériaux et Chimie Industrielle

Je tiens à exprimer toute ma reconnaissance à Madame Gabrièle SAUCIER, Professeur à l'ENSIMAG, pour avoir bien voulu m'accueillir dans son laboratoire de recherche et pour avoir encadré mon travail pendant ces trois dernières années.

Je tiens à remercier :

Monsieur Guy MAZARE, Professeur à l'ENSIMAG pour m'avoir fait l'honneur d'accepter de présider le jury de cette thèse,

Monsieur Roland GERBER Professeur à l'INSA de Rennes et responsable de la division Conception de Circuits Intégrés au CNET / CNS, d'avoir accepté d'être rapporteur de cette thèse,

Monsieur Manfred GLESNER, Professeur à l'Université de Darmstadt, d'avoir bien voulu relire ce travail en français, et accepté d'en être le rapporteur,

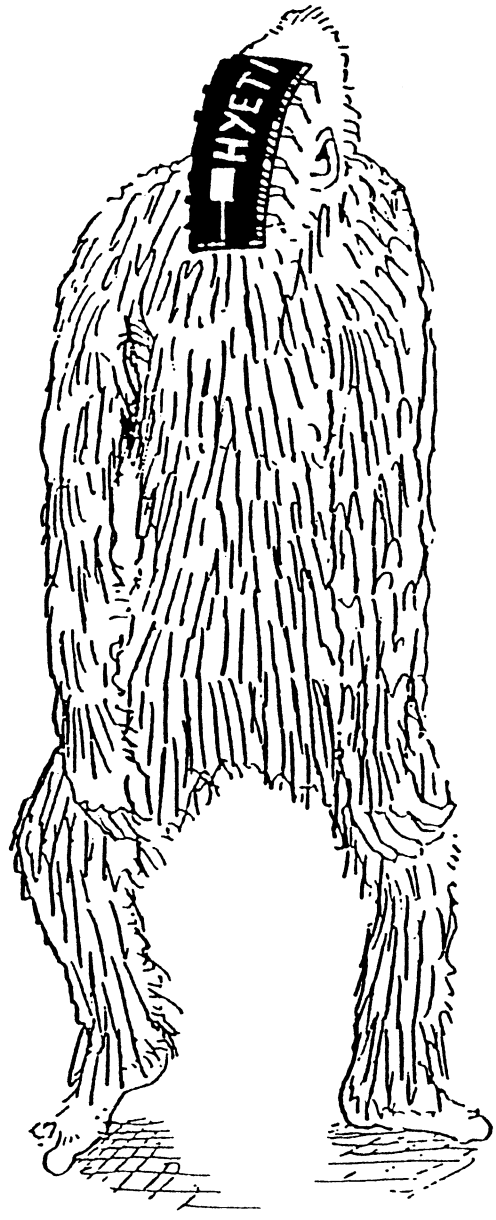
Monsieur Peter IVEY, responsable conception à British Telecom, ainsi que
Monsieur Jacques TRILHE Ingénieur à la Direction Technique de Thomson Semiconducteurs et responsable du projet ESPRIT 824 - WSI,
d'avoir accepté de faire partie de ce jury.

Je remercie également tous mes collègues du laboratoire Circuits et Systèmes pour l'ambiance de travail agréable qu'ils ont su faire régner au sein de l'équipe, et tout particulièrement C. JAY pour sa précieuse collaboration depuis le début de ce travail.

Mein besonderer Dank gibt Herrn Professor Manfred GLESNER für die mir erwiesene Ehre als Berichterstatter meiner These fungiert zu haben.

I am very grateful to Dr Peter IVEY for the honour he did me by accepting to be a member of my thesis jury.

Ce travail a été réalisé dans le cadre du projet 824 (Wafer Scale Integration) du programme ESPRIT financé par la Communauté Economique Européenne.



A mes Parents,

pour avoir su me supporter

durant toutes ces années

Conception de Microprocesseurs à Haut Rendement

Introduction

Les progrès réalisés ces dernières années dans le domaine de la conception et de la fabrication de circuits à très haute intégration (technologies CMOS microniques et sub-microniques, C.A.O (Conception Assistée par Ordinateur) de circuits intégrés puissante) permettent d'envisager la fabrication de circuits comportant plusieurs millions de transistors. Ces possibilités accrues d'intégration ont conduit les recherches vers l'intégration de circuits sur une surface de silicium sans cesse croissante pour en arriver à l'intégration sur tranche entière dite W.S.I (Wafer Scale Integration en Anglais). Ces possibilités ouvrent la porte à l'implantation, sur un seul circuit, de systèmes nécessitant plusieurs cartes imprimées de composants classiques.

Les avantages de ce type de réalisation sont nombreux avec tout d'abord la compacité du système qui est particulièrement importante dans les applications embarquées (aéronautique, spatial, ...) et dont beaucoup d'autres conséquences importantes vont découler. La réduction du nombre de boîtiers entraîne une diminution du nombre d'amplificateurs d'entrée/sortie sur les plots des circuits, ce qui permet une réduction importante de la consommation de courant du système [WSI86]. En diminuant le nombre de connexions entre les boîtiers, on élimine du même coup les risques de mauvaises soudures et de mauvais contacts à tous les niveaux (circuits, connecteurs de bus, ...). Tout ce qui a été énuméré ci-dessus va dans le sens d'une amélioration de la fiabilité des systèmes puisque l'on supprime une bonne partie des sources de pannes. Un autre aspect non négligeable des avantages de l'intégration sur tranche entière concerne la vitesse de fonctionnement des circuits ainsi réalisés : les délais de transmission sont moins importants à l'intérieur d'un même circuit qu'entre plusieurs circuits (pas d'amplificateurs à traverser, connexions plus courtes, capacités liées aux bus et aux circuits imprimés réduites) [FRI84]. Ces différences peuvent aller d'un facteur 2,5 pour l'ECL à un facteur 10 pour le CMOS [NEW85]. Ceci fait que l'intégration sur tranche entière est particulièrement intéressante pour la réalisation de systèmes à hautes performances. Enfin, un dernier avantage et non des moindres est celui du coût du système qui décroît lorsque la densité d'intégration augmente.

La contrepartie de ces avantages est constituée de l'ensemble des problèmes soulevés par la réalisation de tels circuits. Parmi ceux-ci on peut citer la difficulté de réalisation de boîtiers avec un grand nombre de broches et devant permettre la dissipation de la chaleur produite par le circuit en fonctionnement qui sera d'autant plus importante que celui-ci sera rapide [VAL86]. D'autres difficultés de

réalisation proviennent de la taille même du circuit qui implique une distribution d'horloges rapides et de lignes d'alimentation sur de grandes distances (plusieurs centimètres) avec les capacités et les risques de coupures et/ou de courts-circuits que cela comporte. Ce dernier point nous amène au problème crucial pour la réalisation de circuits intégrés de grande taille : les défauts et le rendement de fabrication.

L'objet de cette thèse est l'étude de méthodes permettant la tolérance de défauts de fabrication dans des circuits de type microprocesseur en vue d'en augmenter le rendement de fabrication pour permettre l'intégration de systèmes complets sur un seul circuit.

Dans le chapitre 1 nous étudierons les problèmes posés par les défauts de fabrication, leur incidence sur le rendement de fabrication des circuits intégrés, ainsi que les méthodes possibles de tolérance aux défauts et les différentes architectures adaptées à une réalisation sur circuits de grande taille.

Le deuxième chapitre sera consacré à la présentation des différentes stratégies de reconfiguration pour les parties opératives (PO) de microprocesseurs en donnant des critères de choix.

De la même façon, la conception de parties contrôle (PC) reconfigurables sera abordée dans le chapitre 3.

Enfin, au chapitre 4, on présentera l'application des résultats de cette étude au cas du microprocesseur HYETI réalisé dans le cadre du projet ESPRIT 824 - WSI.

Chapitre 1

Principes généraux de la Reconfiguration

1 - Défauts de fabrication et Rendement des circuits intégrés

1.1 - Défauts de fabrication

Lors de la fabrication de circuits intégrés semi-conducteurs, des millions de dispositifs électroniques de base (transistors, diodes, interconnexions) sont réalisés simultanément par une séquence d'étapes technologiques complexes. Le bon fonctionnement de tous ces dispositifs et de leur assemblage dépend du soin apporté à leur fabrication. La proportion des circuits qui, en fin de fabrication, passent avec succès différentes étapes de tests sévères est appelée le **rendement**. Les défauts de fabrication qui le déterminent peuvent être classés en trois catégories :

* ***Défauts de paramètres :***

Ces défauts résultent de problèmes lors du processus technologique de fabrication et rendent tous les circuits d'une tranche, ou d'une partie importante d'une tranche, inopérants.. Les propriétés des transistors dépendent des durées et des températures utilisées pour les "étapes chaudes" telles que les diffusions, implantation ionique, épitaxie, dépôt d'isolant. C'est au cours de ces étapes que sont formés les sources, les drains, les canaux et les isolants de grille des transistors. Des variations excessives de durée et de température lors de ces étapes vont entraîner une modification des paramètres électriques tels que le gain , la tension de seuil, la résistance ... Des tranches entières de circuits intégrés ne pourront pas fonctionner si ces paramètres n'ont pas les bonnes valeurs. Plus rarement, des parties de tranches seront en panne pour ces raisons. La détection de ce type de défauts se fait par un test des paramètres électriques des circuits à l'aide de motifs implantés à cette fin. La présence de ce type de défauts empêche, de par l'ampleur des dégâts, toute tentative de reconfiguration.

* ***Défauts aléatoires :***

Ces défauts sont ceux qui causent la plupart des pertes de circuits sur une tranche. Il s'agit principalement de défauts de petite taille, défauts d'isolation provoqués par des particules microscopiques déposées sur les tranches au cours des différentes étapes entraînant défauts d'insolation, trous dans l'oxyde ... Ils se traduisent par des motifs manquants ou rajoutés dans les différents conducteurs, diffusion, silicium polycristallin, métallisation entraînant des courts-circuits, des coupures, des disparitions de transistors. C'est pour ce type de défauts que l'on a le plus étudié de modèles de rendement comme nous le verrons au paragraphe 1.2.

* ***Amas de défauts :***

Ils sont dus à des regroupements de particules sur la tranche, à des poussières récoltées lors des manipulations entre les étapes technologiques. Divers paramètres physiques font que ces particules ont tendance à s'agglomérer pour former des défauts de grande taille.

Les trois types de défauts que nous avons présentés jusqu'ici sont ceux qui peuvent empêcher un circuit de fonctionner dès sa fin de fabrication. Toutefois certains défauts peuvent rester cachés et ne pas être détectés tout de suite ; ils seront alors classés parmi les disfonctionnements du circuit détectés durant sa vie et appelés "pannes". Ce processus peut être représenté par la courbe de la figure 1.1 où sont représentées les différentes périodes de la vie du circuit.

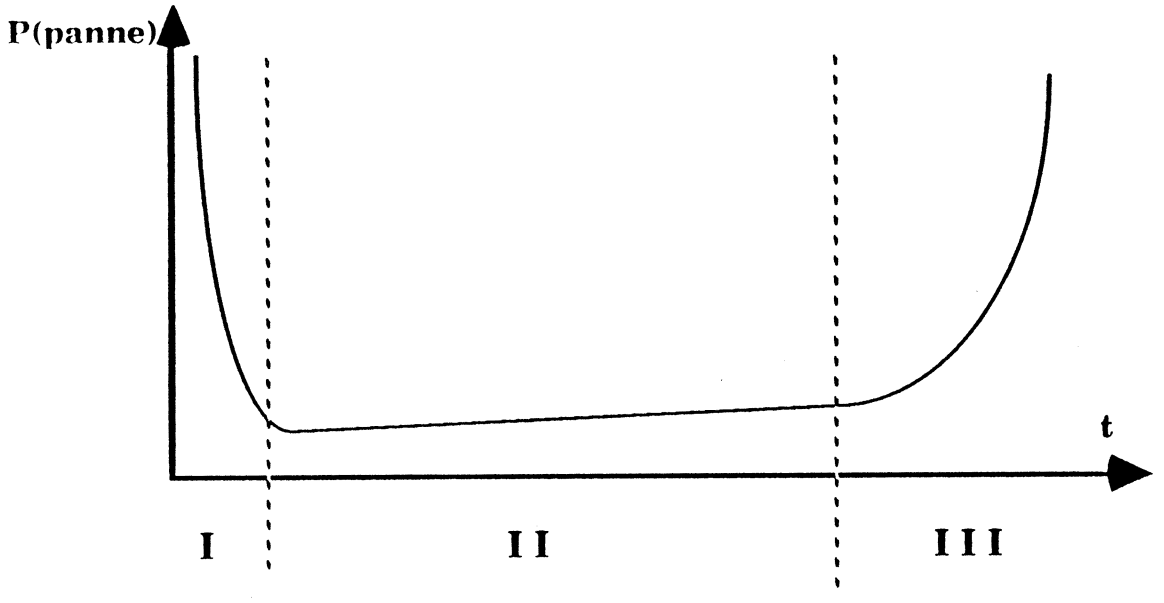


Figure 1.1 : Probabilité de panne d'un circuit au cours de sa vie

On peut distinguer trois périodes principales dans la vie d'un circuit intégré :

En **I**, on se trouve dans une période où le nombre de défauts non détectés par les tests de fin de fabrication peuvent être encore nombreux. Plus le temps passe, plus le circuit fonctionne, et moins on a de risques de trouver des défauts cachés. La première partie de la vie du circuit va donc consister en un vieillissement accéléré de façon à détecter rapidement les circuits défectueux (déverminage). On arrive alors à la période **I I** où le circuit est réputé bon. Durant cette période il peut subsister des défauts de fabrication résiduels ; il apparaît également des

disfonctionnements liés au viellissement du circuit. En technologie MOS ceux-ci sont pour l'instant mal identifiés. On peut imaginer des phénomènes similaires à ceux constatés en technologie bipolaire (corrosion de l'aluminium, électromigration dans l'aluminium) ; ces phénomènes constituent des "pannes" du circuit. De nombreuses études ont été menées pour pouvoir corriger et tolérer ces pannes au cours de la vie du circuit. Dans les applications nécessitant une haute sûreté de fonctionnement, la période **I I I** est celle où l'usure du circuit rend les pannes de plus en plus probables, ce qui conduit au remplacement du circuit.

Notre objectif étant la réalisation de circuits de grande taille, les problèmes de pannes durant la vie du circuit ne nous intéressent pas directement, et nous verrons au paragraphe 2.1 que les méthodes de tolérance aux pannes ne conviennent de toute façon pas dans notre cas. Le problème qui est le nôtre est de pouvoir réaliser avec un rendement acceptable des circuits dont la taille implique qu'ils seront obligatoirement atteints par des défauts de fabrication. Il s'agit donc ici de développer des méthodes de tolérance aux défauts de fabrication. Il ne s'agira pas d'être capable de corriger tous les défauts, certains comme les défauts paramétriques empêchant même tout fonctionnement. Le but va être de pouvoir réparer des circuits atteints par des défauts aléatoires ou, dans une moindre mesure, par des amas de défauts. Ceci est réalisé en implantant du matériel en réserve de façon à pouvoir reconfigurer un circuit défectueux. Les stratégies de reconfiguration dépendent de la répartition et de la taille des défauts qui déterminent celle des blocs reconfigurables. Ces informations sont données par la modélisation des phénomènes d'apparition de défauts.

1.2 - Rendement

Le choix des stratégies de reconfiguration pour un circuit dépendent du nombre d'éléments en panne qui peuvent s'y trouver. On aura par exemple un plus grand nombre de blocs reconfigurables si le nombre de défauts est élevé. D'autre part, le degré de redondance nécessaire pour atteindre un rendement donné dépend du nombre de défauts présents. Comme l'apparition de défauts sur une tranche est un phénomène aléatoire, le nombre exact de circuits en panne n'est pas prévisible. En revanche, une densité de probabilité (probabilité d'avoir un nombre donné de défauts) décrit ce phénomène. Un **modèle de rendement** est un modèle mathématique du processus de fabrication des circuits intégrés reliant la probabilité d'occurrence de défauts à des facteurs tels que les règles de dessin, la densité de défauts, etc... Le seul paramètre directement contrôlé par le

concepteur étant la surface du circuit, c'est en fonction de cette dernière que nous allons décrire un modèle de rendement. Le modèle final recherché dans notre cas étant une fonction **P (X = m ; S)**, donnant la probabilité d'avoir exactement **m** défauts sur une surface **S** de silicium. Cette fonction permettra ensuite de déterminer la taille maximale d'un bloc reconfigurable ainsi que le degré de redondance optimal.

1.2.1 - Modèles de rendement existants

De nombreuses études ont été faites depuis plusieurs années sur le rendement de fabrication des circuits intégrés [HYE86]. Lors des premiers travaux sur le rendement, la densité de défauts a été modélisée par une loi de Poisson [MUR64], ce qui donne l'expression suivante pour le rendement (en supposant que les défauts sont indépendants) :

$$(1.1) \qquad R = e^{-D.S}$$

où **D** est la densité moyenne de défauts sur la tranche, et **S** la surface active totale du circuit. Si D_0 est la densité moyenne de défauts par niveau de masque critique, et si **n** est le nombre de ces niveaux, on a $D = n.D_0$.

Mais la densité de défauts n'est pas uniforme, ce qui amena B. Murphy à proposer une expression plus générale pour le rendement :

$$(1.2) \qquad R = \int_0^{\infty} f(D) \, . \, e^{-D.S} \, . \, dD$$

où **f(D)** est la fonction de distribution de la densité de défauts. La forme exacte de la fonction n'est pas connue et varie d'une chaîne de fabrication à l'autre. Plusieurs modèles ont été proposés qui, utilisés avec les bonnes hypothèses, peuvent être tous employés ; le "bon" modèle étant celui qui s'approche le plus des mesures effectuées sur les produits de la chaîne de fabrication [STA83]. Les modèles d'estimation de rendement les plus couramment utilisés sont les suivants : Poisson (voir ci–dessus), Murphy, Seeds, Murphy–Seeds, Price.

* ***Murphy :***

B. Murphy a proposé plusieurs fonctions pour f(D) pour arriver à un choix. Il a tout d'abord pris une distribution de largeur 0 qui représente le cas d'une loi de poisson (équation 1). Ensuite, une distribution très large, simplifiée en un rectangle et qui donne :

$$(1.3) \qquad R = \frac{1 - e^{-2\ S.D}}{2\ S.D}$$

La dernière solution est en fait un intermédiaire entre les deux autres : la distribution est représentée par un triangle isocèle ayant un sommet pour la densité moyenne D_m et qui donne pour le rendement :

$$(1.4) \qquad R = \left(\frac{1 - e^{-S.D}}{S.D} \right)^2$$

Ces trois versions de f(D) sont illustrées dans la figure 1.2 ci-dessous.

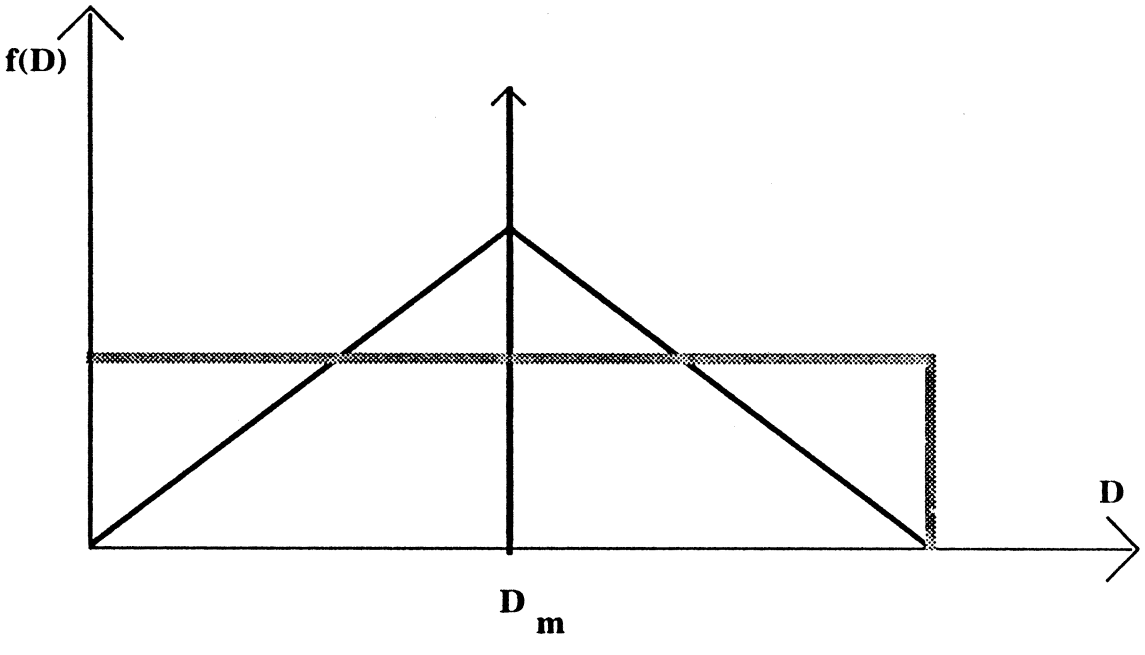


Figure 1.2 : Exemples de fonctions densité de probabilité utilisées par Murphy

* ***Seeds :*** [SEE67]

R.B. Seeds a utilisé une distribution à décroissante exponentielle pour f(D) et arrive à l'expression suivante du rendement :

$$(1.5) \qquad R = e^{-\sqrt{S.D}}$$

* ***Murphy-Seeds :***

Il s'agit ici de prendre comme modèle du rendement la moyenne des expressions de Murphy et de Seeds :

$$(1.6) \qquad R = \frac{\left[\frac{1 - e^{-S.D}}{S.D}\right]^2 + e^{-\sqrt{S.D}}}{2}$$

* ***Price :***

Le modèle développé par J.E. Price à partir des statistiques de Bose-Einstein est basé sur l'hypothèse que le nombre de mécanismes produisant des défauts est proportionnel au nombre d'étapes technologiques. De ces hypothèses est issu le modèle suivant :

$$(1.7) \qquad R = \frac{1}{(1 + S.D_0)^n}$$

Une comparaison de ces différentes formulations du rendement est donnée dans la courbe de la figure 1.3 ci-dessous.

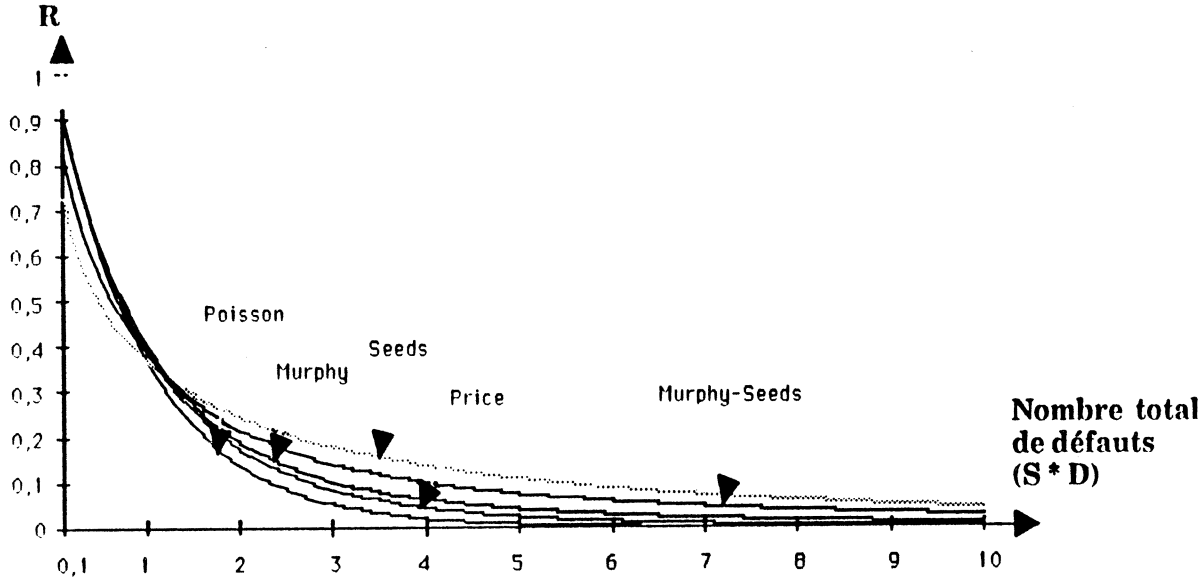


Figure 1.3 : Comparaisons de différentes formulations du rendement

Le tableaux des valeurs de cette courbe ainsi que de ceux des courbes des figures 1.4 et 1.5 sont donnés dans l'annexe 1.

Le choix d'un modèle se fait en fonction des résultats observés pour la technologie utilisée. Dans le cadre du projet ESPRIT 824, cette technologie est la HCMOS3C (1,3 µm de longueur de canal) de Thomson Semiconducteurs, et les

modèles de rendement recommandés sont Murphy–Seeds et Price (aussi appelé Bose–Einstein). Les paramètres utilisés dans ce cas sont : $D_0 = 1$ défaut/cm^2 et nombre de niveaux de masques critiques : $n = 7$.

1.2.2 - Modèle adapté à l'évaluation de la taille des blocs reconfigurables

Le rendement étant la probabilité de ne pas avoir de défauts, on conçoit facilement que les modèles présentés précédemment ne puissent pas être utilisés tels quels dans notre cas. En effet, dans un circuit reconfigurable, un certain nombre de défauts peuvent être présents sans l'empêcher de fonctionner. Nous sommes donc plus intéressés par le nombre de défauts et leur distribution que par le rendement proprement dit.

La probabilité d'avoir exactement **m** défauts sur une surface **S** de circuit est **P (X=m ; S)** où **X** est une variable aléatoire représentant le nombre de défauts. La probabilité pour qu'un circuit de surface S puisse tolérer m' défauts est **P (X ≤ m' ; S)** (fonction de répartition de X).

Soit x_i la variable aléatoire représentant le nombre de défauts introduits par la $i^{ème}$ étape technologique, et **X** le nombre de défauts après toutes les étapes. Dans un processus comportant plusieurs étapes, le nombre total de défauts est la somme des défauts introduits par chaque étape. On aura donc, dans P(X=m ; S), $X=x_1+...+x_k$ pour **k** étapes technologiques. X est la somme de k variables aléatoires indépendantes et de même loi géométrique de paramètre **p** compris entre 0 et 1 [SUB79] :

$$(1.8) \qquad P(x_i = n ; S) = p^n . (1 - p)$$

où $1 - p$ est le rendement par niveau de masque critique, c'est à dire la probabilité de ne pas avoir de défaut sur ce masque dans une surface S de silicium, $1 - p = (1 + S.D_0)^{-1}$ d'après le modèle de Price.Alors X suit une loi de Pascal (loi binômiale négative) de taille k :

$$(1.9) \quad P(X=m ; S) = P(x_1+x_2+ ... +x_k=m ; S)$$

$$= \frac{k . (k+1) . (k+2) ... (k+m-1)}{m!} p^m (1 - p)^k$$

$$= (-1)^m \overset{m}{C}_{-k} p^m (1 - p)^k \qquad k = 0, 1, 2, ...$$

Avec la notation du binôme généralisé :

$$C_x^r = \frac{x\,(x-1)\,(x-2)\ldots(x-r+1)}{r!} \qquad x \in R,\ r \in N$$

A partir de là, on peut établir la courbe de la probabilité d'avoir **m** défauts dans un circuit en fonction de sa surface qui est présentée dans la figure 1.4 pour la technologie HCMOS3C.

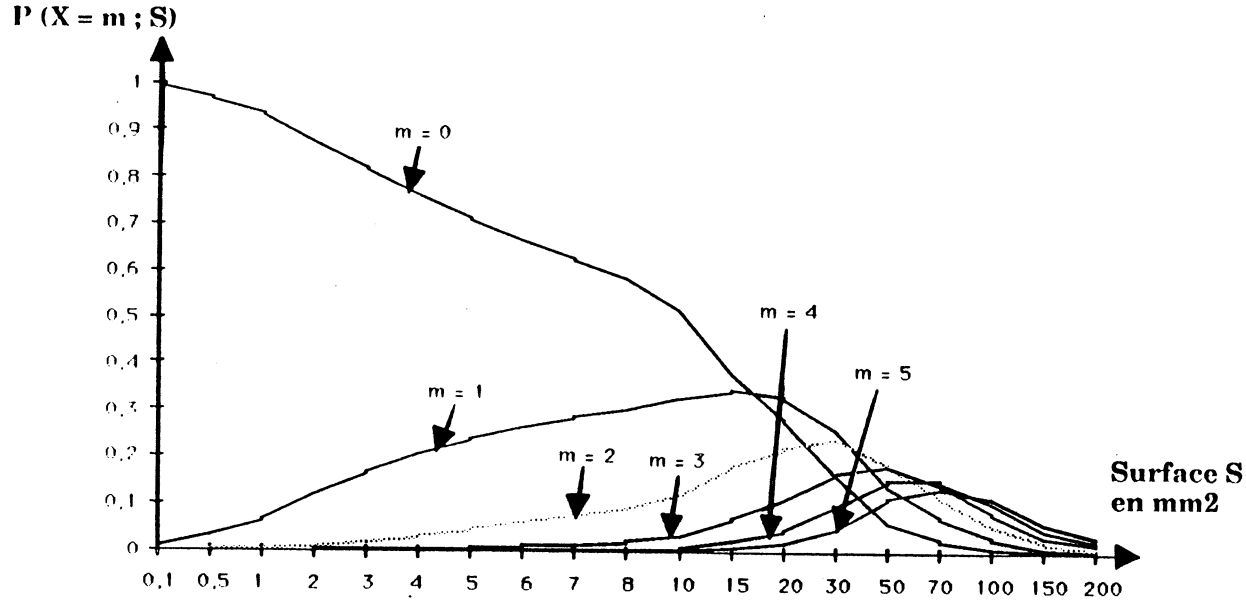


Figure 1.4 : Probabilité d'avoir m défauts dans un circuit

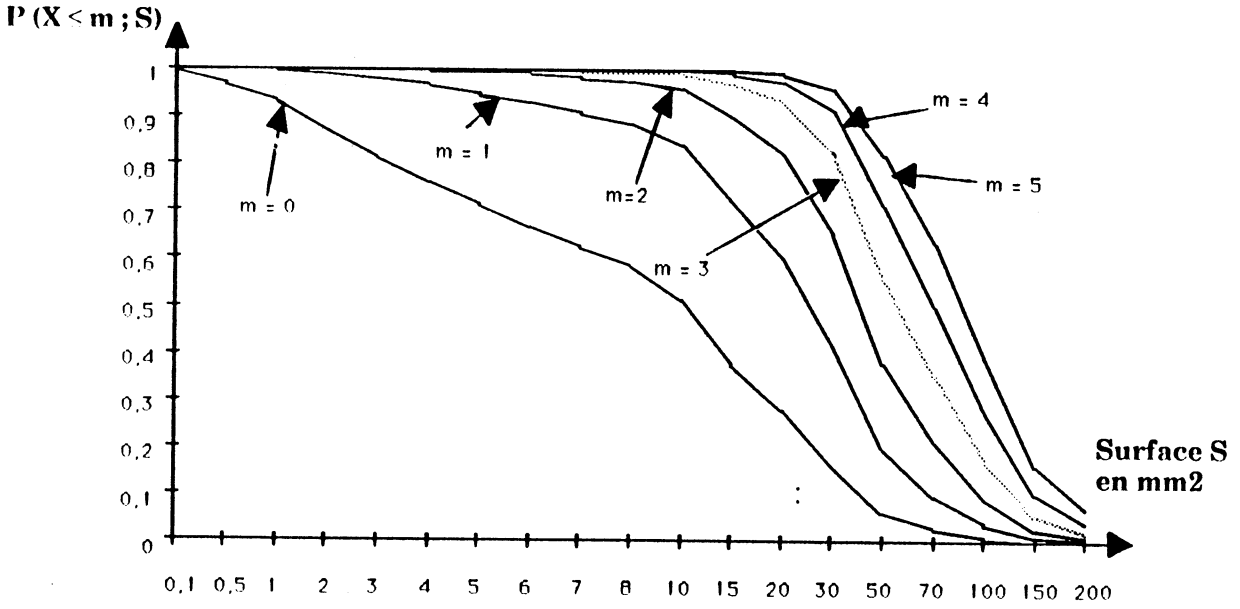


Figure 1.5 : Fonction de répartition de X

Ce résultat permet de déterminer $P(X \leq m ; S)$ qui est décrite dans la figure 1.5. Cette courbe permet de savoir dans quelle surface de circuit on peut avoir au plus m défauts. Il est alors possible d'en déduire la taille d'un bloc reconfigurable en fonction du nombre de défauts que l'on peut y réparer : soit **r** le nombre de défauts réparables dans un bloc (r est déterminé par la stratégie de reconfiguration choisie), soit p_r la probabilité (que l'on désire atteindre) d'obtenir un bloc reconfigurable, la surface **S** du bloc reconfigurable sera obtenue en résolvant l'inéquation :

$$(1.10) \quad p_r \leq P(X \leq r ; S) <=>$$
$$p_r \leq P(X = 0 ; S) + P(X = 1 ; S) + ... + P(X = r ; S)$$

2 - Redondance et Configuration d'un circuit en fin de fabrication

2.1 - Redondance dans les systèmes tolérant les pannes

La tolérance de pannes dans les circuits logiques a déjà été largement étudiée pour les besoins de la transmission de données ou pour des systèmes à haute sûreté de fonctionnement [TRY], [STI77], [HIG76]. Ces recherches ont mis en évidence plusieurs types de redondances permettant la tolérance de pannes survenant pendant la vie du système [CAS84].

* ***Redondance de masquage*** :

Le but est de masquer une panne intervenant au cours du fonctionnement du système. Ce type de stratégie peut être basé sur une redondance massive de matériel (NMR c'est à dire N - Modular Redundancy en Anglais). Une unité est répliquée n fois (n impair), un vote majoritaire sur les sorties permet de masquer celles qui sont en panne. Le schéma le plus connu de ce type de redondance est le TMR où n=3 [WAK76].

Une autre façon de masquer des pannes (dans le cas de mémoires ou de transmission de données) est d'utiliser les codes correcteurs d'erreurs (redondance sélective). Cette redondance est coûteuse dès lors qu'il s'agit de corriger des pannes multiples.

* ***Redondance de réserve (stratégie de détection - remplacement)*** :

Lorsqu'une panne survient, l'unité fautive est remplacée. Le mode de détection introduit des différences importantes dans ce type de stratégie. Si la

détection est faite en utilisant une redondance massive (matérielle ou logicielle), chaque unité subit une détection séparément : lorsqu'une panne survient, l'unité fautive est trouvée et remplacée.

Quand la détection est faite en comparant deux unités complètes ("duplex"), aucun diagnostic n'est possible et les deux doivent être remplacées soit par une paire de réserve ("bi" ou "N-Duplex" [COU76]), soit par une seule unité de réserve ("Duplex réserve"). On peut noter que dans ce dernier cas la sécurité diminue puisque les résultats ne sont plus comparés. Si un diagnostic est possible, on peut éliminer l'unité en panne et former une nouvelle paire avec la réserve ("Duplex dynamique").

* ***Redondance hybride*** **:**

Le but de ce genre de stratégie est de prendre les avantages des deux solutions précédentes : les solutions de détection – remplacement avec leurs dispositifs de matériel en réserve et les solutions de masquage avec leurs dispositifs facilitant la maintenance. On utilisera donc ici les mécanismes de base des stratégies de masquage, codes correcteurs (la redondance sélective) ou TMR (redondance massive). Un dispositif de détection permet à l'unité en panne d'être mise de côté et à une unité de réserve d'être appelée.

On peut faire les remarques suivantes quant à l'utilisation de ces stratégies de redondance :

- Tout d'abord, elles ont été conçues à l'échelle du système et pour lui permettre de survivre à des pannes survenant durant son fonctionnement. Ceci est différent de notre objectif qui est la tolérance aux défauts de fin de fabrication.
- Elles ne permettent, le plus souvent, que de corriger une seule panne.
- L'utilisation d'unités répliquées au moins deux ou trois fois augmente considérablement la taille du système et va donc faire décroître le rendement et la fiabilité du circuit.
- La correction d'erreurs multiples par codage est très complexe à réaliser.

2.2 - Redondance dans les circuits tolérant les défauts

Le principe de base de la tolérance aux défauts de fabrication est de prévoir dès la conception du circuit un certain nombre d'éléments de réserve destinés à

remplacer ceux qui viendraient à être atteints par des défauts. L'utilisation de cette redondance pour configurer un circuit en fin de fabrication implique que l'on puisse détecter et localiser **tous** les défauts. Cela exclut donc toutes les stratégies tendant à masquer les pannes telles celles exposées en 2.1.

Les stratégies de redondance utilisées varient selon l'architecture du circuit (voir paragraphe 3) et selon les contraintes particulières de l'application. On peut néanmoins leur trouver les points communs suivants : dans tous les cas on détermine un ou plusieurs niveaux de reconfiguration. A chaque niveau on détermine la taille du bloc reconfigurable, c'est à dire la taille du bloc où pas plus d'un certain nombre de défauts ne sont susceptibles d'apparaître (probabilité fixée par les calculs vus précédemment), et on implante dans chaque bloc les éléments redondants nécessaires pour corriger ce nombre de défauts.

Ceci suppose l'utilisation de dispositifs de commutation permettant de déconnecter les éléments en panne et de les remplacer par ceux en réserve. De tels dispositifs sont étudiés depuis une vingtaine d'années. Les premières recherches sur l'intégration de circuits sur tranche entière se sont orientées vers le câblage discret des bons circuits sur la tranche. Une fois les cellules testées individuellement à l'aide d'une carte à pointes, un schéma d'interconnexion était généré pour la tranche. Il était réalisé par des niveaux supplémentaires de métal placés sur les cellules. Ce système avait deux inconvénients :

* Un coût excessif dû à l'obligation de faire un ensemble de masques spécifiques pour les motifs d'interconnexion de chaque tranche.
* Les défauts de fabrication qui peuvent affecter aussi bien les niveaux supérieurs de métallisation que les autres niveaux de masquage [AUB78].

Dans [CAL72], D. Calhoun propose une technique permettant de limiter le nombre de masques spécifiques à un seul. Cette technique consiste à avoir un motif commun de connexions dans le niveau supérieur de métal qui relient des plots fixes réalisés dans le premier niveau de métal. Ce niveau inférieur de métal est personnalisé pour chaque tranche de façon à relier les interconnexions des cellules aux plots. Seules les bonnes cellules sont reliées aux plots. Le niveau supérieur de métal réalise une suite standard de connexions entre des points fixés. Les plots ainsi que les connexions entre les cellules et les plots sont adaptés

en fonction de la répartition des défauts sur chaque tranche. Seul le masque définissant le niveau inférieur de métal doit être propre à chaque tranche. Malgré une réduction du coût, cette technique n'a pas permis de produire des systèmes sur tranche fiables et économiques. Les problèmes viennent des suppositions de départ disant que les étapes technologiques supplémentaires n'amèneraient pas de défauts, et que les cellules saines après un test individuel le seraient toujours après les dernières étapes technologiques.

Les travaux suivants [AUB78] se sont orientés vers des solutions ne nécessitant pas d'étapes technologiques supplémentaires. Les méthodes développées nécessitent la présence de dispositifs de commutation intégrés à chaque cellule pour pouvoir ne connecter une cellule qu'à ses voisins sains. Actuellement on peut distinguer trois types de tels dispositifs : ceux programmables par laser, ceux programmables par faisceau d'électrons, et ceux programmés par logique.

2.2.1 - Les dispositifs programmés par LASER

Ils ne sont pas réversibles. Ils sont de deux types : dispositifs de connexion et dispositifs de déconnexion. Parmi les premiers, on compte le dispositif développé au Massachussets Institute of Technology et qui consiste à former des vias à l'aide d'un laser entre deux niveaux d'aluminium séparés par un diélectrique spécial [CHA85]. On étudie également les anti-fusibles dont le principe consiste à relier deux lignes de métal par un procédé de "lift off" par exemple.

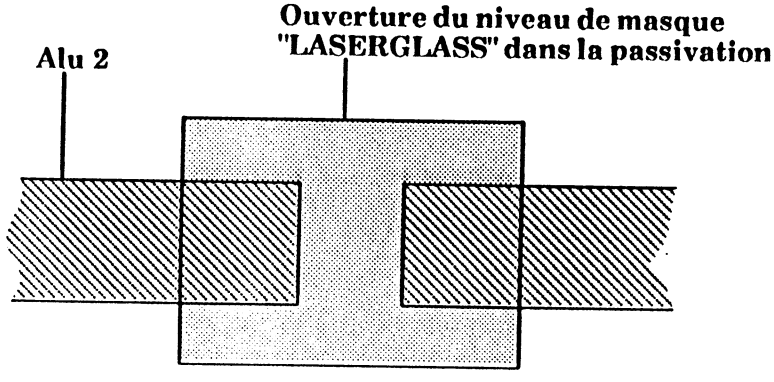


Figure 1.6 : Anti-fusible laser

Les déconnexions sont obtenues à l'aide de fusibles que l'on fait sauter avec un laser. Ces fusibles peuvent être de deux types : soit en silicium polycristallin (figure 5 ci-dessous), soit en aluminium (dans ce cas on parle généralement de coupure de ligne de métal).

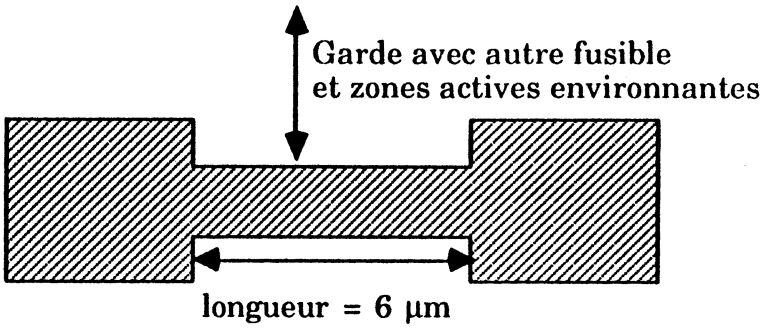


Figure 1.7 : Fusible laser en silicium polycristallin

2.2.2 - Les dispositifs programmés par faisceau d'électrons

Il s'agit de transistors à grille flottante que l'on programme en chargeant (ou en déchargeant) la grille à l'aide d'un faisceau d'électrons [SHA84]. Les principaux avantages de ce système sont que les grilles flottantes peuvent être reprogrammées et que l'information qui y est programmée n'est pas volatile. De plus de tels dispositifs ont un coût peu élevé car ils ne nécessitent pas d'étapes technologiques supplémentaires. L'inconvénient majeur est l'incertitude quant à la rétention de l'information dans le temps ainsi qu'à haute température.

2.2.3 - Les dispositifs programmés par logique

L'emploi de dispositifs de commutation programmés par logique offre une très grande souplesse d'utilisation du fait de leur structure qui comporte essentiellement des registres et des portes de transfert programmées par le contenu des registres en question [CHE85], [KAT86]. De tels commutateurs sont programmés en utilisant le chemin de données. Ils ne nécessitent aucune modification du processus de fabrication des circuits non plus que l'emploi de microscope électronique à balayage (MEB) ou de laser. Ce type de dispositif est indispensable dans le cas où l'on ne peut programmer un réseau de cellules que depuis la périphérie du circuit. Leur inconvénient essentiel est leur encombrement.

3 - Architectures intégrables sur tranche entière

Les méthodes de reconfiguration utilisées pour un circuit de grande taille dépendent de son architecture. On peut actuellement distinguer trois catégories de circuits intégrés (ou intégrables) sur tranche entière :

- Des mémoires de grande capacité
- Des réseaux systoliques
- Des systèmes à base de microprocesseurs

3.1 - Mémoire

Si la réalisation de mémoires mortes ou de mémoires vives dynamiques de grande capacité est possible sur des circuits de taille "normale", il n'en est pas de même pour les mémoires vives statiques. En effet, une cellule de mémoire vive statique est composée de six transistors contre un seul pour la mémoire vive dynamique.

Plusieurs études sont en cours actuellement, et des circuits ont déjà été réalisés.

NTT a réalisé une 4 Méga-bit sur une tranche de 3 pouces en 1980, et une 1,5 Méga-bit SRAM sur une tranche de 4 pouces en 1984. Ces mémoires sont organisées de la même façon, soit , dans le cas de la 1,5 Mbit, en blocs de 256K x 6 bits avec duplication complète de chaque bloc. La sélection des bons blocs est faite par un contrôle de parité. Une telle technique de reconfiguration est très attractive car elle est active à chaque mise sous tension, ce qui permet de corriger non seulement les défauts de fabrication, mais aussi les pannes survenant durant la vie du circuit. Cette mémoire n'est pas très rapide avec un temps d'accès de l'ordre de 520 ns.

Sinclair a réalisé une mémoire à accès série de 0,5 Mbit (en NMOS sur une tranche de 4 pouces) pour remplacer les disques durs dans les micro–ordinateurs portables. Pour une telle application, le temps d'accès annoncé de 10µs est largement suffisant. La stratégie de reconfiguration utilisée est celle proposée par Ivor Catt qui consiste à construire une spirale de cellules saines à chaque mise sous tension.

INOVA travaille sur l'interconnexion de cellules de mémoire vive statique. Cette société américaine a réalisé une mémoire de 1 MBit d'un temps d'accès de 150 ns à l'aide de 144 cellules de 16 KBit sur une tranche de 4 pouces en technologie CMOS 3 µm. Une 24 MBit à base de cellules de 256KBit est annoncée pour fin 1987.

L'intégration d'une mémoire vive statique de grande capacité est actuellement faîte en interconnectant des cellules de capacité plus modeste sur une tranche. On peut noter que la capacité de ces cellules augmente grâce aux progrès de la densité d'intégration et de la taille des puces. L'architecture d'une telle mémoire est la suivante (d'après l'exemple de la mémoire vive statique de 4,5 Mega Bits réalisée dans le cadre du projet ESPRIT 824) :

Organisation en mots de *m* bits (16 bits + 2 bits de codage, ou deux fois 8 bits et un bit de parité dans le cas de notre exemple). Chaque bit est réalisé par un bloc de plusieurs cellules de base (quatre cellules de 64K Bits dans notre cas). L'organisation virtuelle de la mémoire 4,5 Mega Bits est explicitée dans les figures 1.8 et 1.9 ci-dessous où l'on voit qu'il est nécessaire de modifier la périphérie de la cellule de base pour implanter le décodage des deux lignes d'adresse supplémentaires rendues obligatoires par l'utilisation de quatre cellules de base par bit.

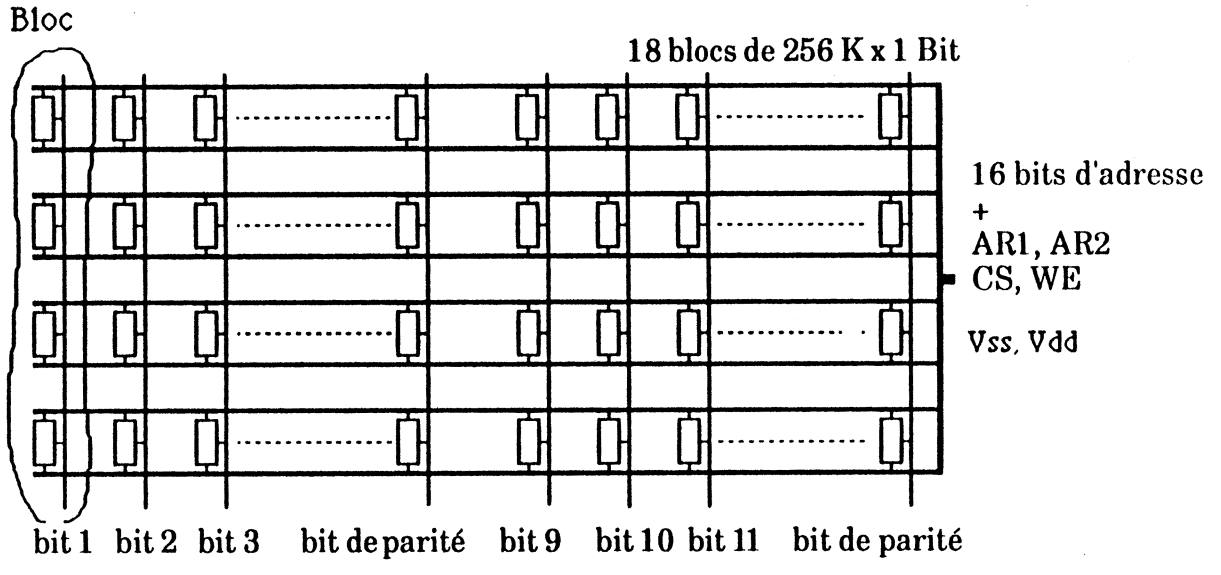


Figure 1.8 : Architecture virtuelle de la mémoire vive statique de 4,5 MBits

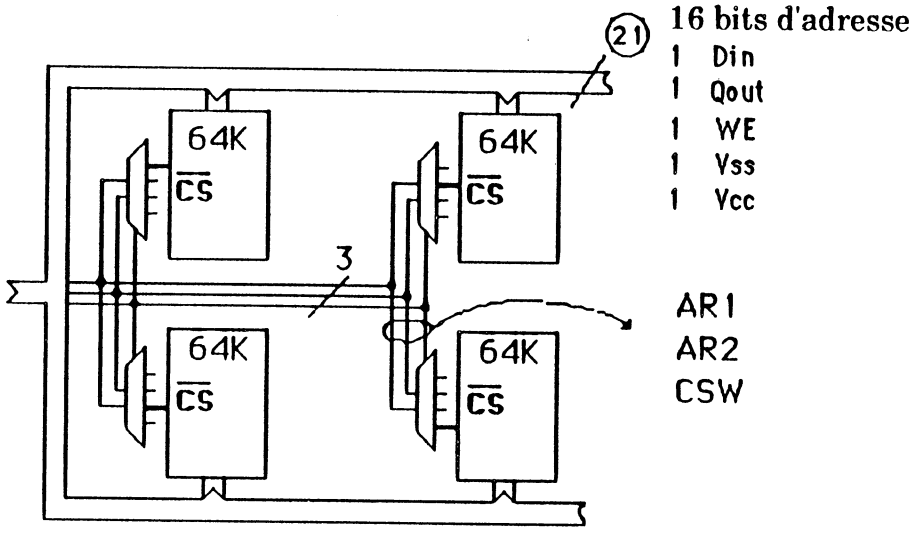


Figure 1.9 : Décodage décentralisé des deux fils d'adresse supplémentaires

Outre les problèmes posés par les lignes d'alimentation et les lignes d'adresse qui courent sur toute la tranche, la principale difficulté est d'intégrer suffisamment de cellules de base pour avoir le nombre requis de cellules saines en fin de fabrication pour réaliser la mémoire. La solution de ce problème, qui est également rencontré dans la production industrielle de mémoires où un faible coût de production implique un rendement élevé, consiste à reconfigurer les cellules de base en panne.

De nombreuses études ont été menées à ce sujet [SAK84], [KAN84], [SMI81], [FLA86]. Les techniques employées consistent à ajouter des lignes et/ou des colonnes de cellules mémoire en réserve. La reconfiguration est faite en modifiant le décodage de l'adresse soit électriquement (fusibles électriques), soit à l'aide de fusibles en silicium polycristallin programmés par laser, cette dernière méthode étant de plus en plus utilisée actuellement. La réalisation de la mémoire à l'échelle de la tranche se fait de la manière suivante :

* Après fabrication, on va tester individuellement chaque cellule de base à l'aide de cartes à pointes classiques (car les cellules de base conservent leurs plots d'entrée/sorties). Les cellules en panne vont ensuite être reconfigurées, après quoi un nouveau test délivrera une cartographie des cellules saines sur la tranche.
* On procède ensuite au test des lignes d'adresse, de données et d'alimentation qui traversent la tranche.

* L'étape suivante consiste à isoler les cellules en panne du réseau d'interconnexion en programmant un transistor à grille flottante qui contrôle les portes de transfert situées entre chaque plot d'entrée/sortie et le réseau d'interconnexion (voir figure 1.10 ci-dessous).

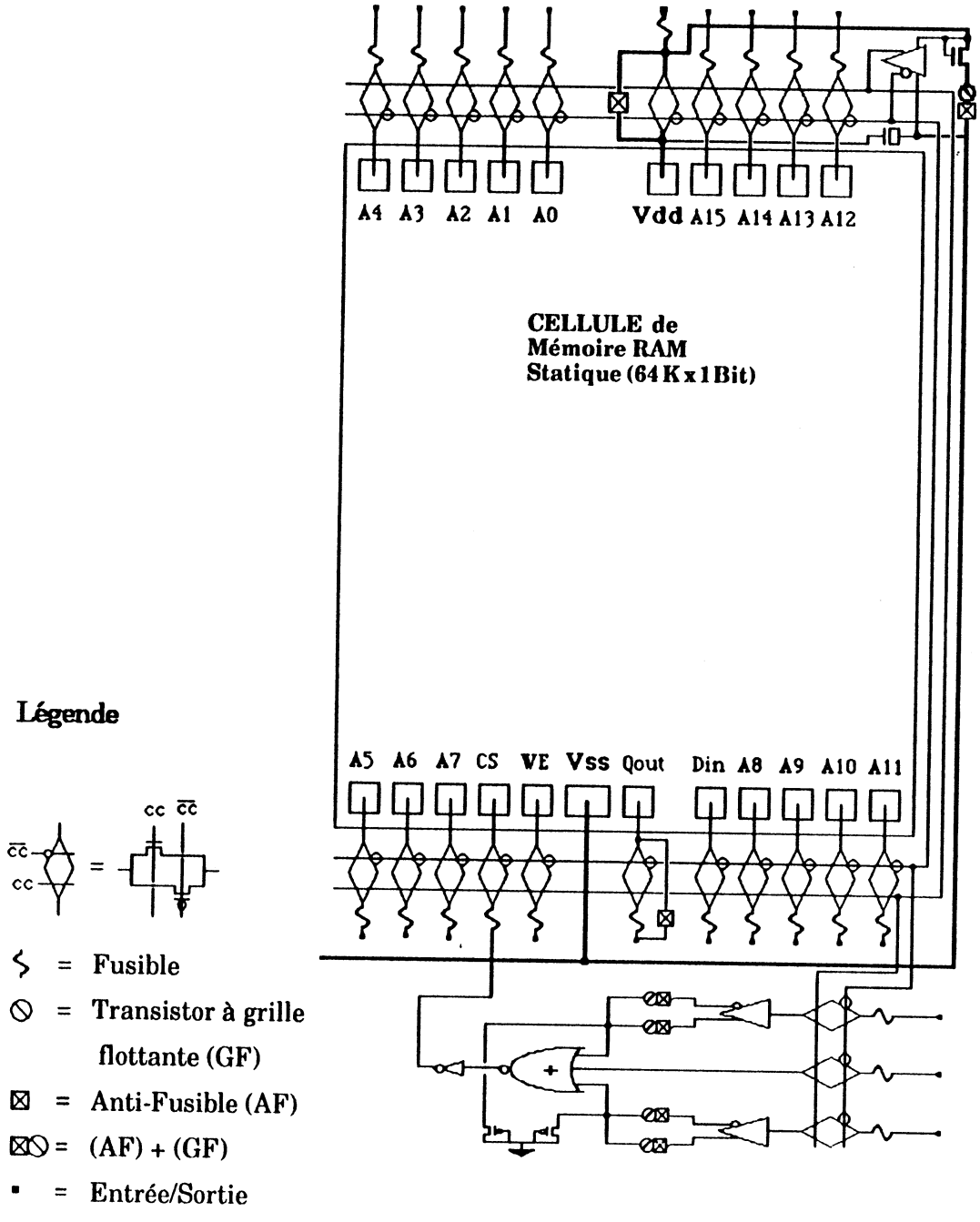


Figure 1.10 : Architecture de la périphérie d'une cellule de 64 K x 1 Bit

On va alors chercher une configuration viable de la mémoire en connectant les cellules saines de la mémoire au réseau d'interconnexion.

* Une fois la tranche configurée et testée, on fige la configuration à l'aide de fusibles et d'anti-fusibles programmés par laser.

3.2 - Réseaux Systoliques

Ce type d'architecture (matrice de cellules identiques ne communiquant qu'entre voisines) est particulièrement bien adapté à une réalisation sur circuits à très haute échelle d'intégration ; on élimine ainsi les coûts et les pertes de performances rencontrées avec les cellules en boîtiers individuels [LEI85]. Ces motivations poussent à augmenter le nombre de cellules par circuit et amènent à l'intégration de réseaux systoliques sur une tranche de silicium entière.

Là encore plusieurs études sont en cours sur ce sujet. Nous citerons pour mémoire les circuits spécialisés pour le traitement d'image tels que le WASP de l'Université de Brunel (Royaume Uni) [LEA86] ou le circuit développé au sein du projet ESPRIT 824 - WSI pour faire de la compaction d'image ou de la transformation géométrique d'image en temps réel, les circuits plus généraux pour le calcul parallèle tels que le CHIP développé à l'Université de Purdue (U.S.A) par K. Hedlung [HED82].

Le principal problème qui se pose pour la réalisation de tels circuits est que sur le grand nombre de cellules intégrées, certaines auront des défauts et ne pourront pas être utilisées. De nombreuses solutions ont été proposées, tant en ce qui concerne les algorithmes de reconfiguration que les dispositifs matériels nécessaires, pour obtenir un réseau fonctionnel à partir des cellules saines sur une tranche. La réalisation d'un réseau de cellules saines suppose que l'on puisse contourner celles qui sont en panne. On a besoin pour cela d'interconnexions programmables.

Les différents types de commutateurs présentés précédemment (2.2) sont utilisables pour intégrer des réseaux systoliques sur tranche entière. Cette intégration de réseaux à une ou deux dimensions requiert l'utilisation d'un algorithme d'immersion des cellules saines dans le réseau d'interconnexion. Il existe beaucoup de tels algorithmes, parmi lesquels ceux présentés dans [LEI86]. On peut distinguer deux types d'algorithmes : ceux qui utilisent toutes les cellules saines et ceux qui n'en utilisent que la plupart. Le principe général consiste (pour les plus performants) à diviser le circuit en sous-circuits de $m X n$ cellules dans lesquels on cherchera à obtenir un réseau de $p X q$ cellules (avec p<m et q <n) soit en éliminant les lignes et /ou les colonnes contenant des cellules en panne, soit en contournant les cellules en panne.

3.3 - Systèmes à base de microprocesseurs

3.3.1 - Architecture

Le but est ici d'intégrer sur un même circuit un système à base de microprocesseur(s) nécessitant plusieurs cartes de circuits imprimés (soit pour réaliser plusieurs systèmes communiquant entre eux sur une même tranche, soit pour intégrer un système complet sur une grosse puce).

Cette approche est déjà proposée au niveau microprocesseur par la société américaine WSI Inc. qui réalise des circuits 16 ou 32 bits personnalisés à base de parties opératives en tranche du type AMD 2901. La configuration de haut niveau étant faite par le micro–programme stocké en EPROM.

La structure de tels circuits comporte un ensemble de blocs prédéfinis : microprocesseurs et périphériques divers, immergés dans une structure d'interconnexion souple. Cette organisation a deux buts : d'une part la souplesse d'interconnexion doit permettre de configurer le système en fin de fabrication en tenant compte des éléments défaillants, et d'autre part elle permet (avec l'utilisation de blocs personnalisables) d'intégrer des systèmes adaptés à une application précise à partir d'un ensemble de blocs prédéfinis, comme montré dans la figure 1.11 ci–après.

Une telle approche s'apparente à la personnalisation de cartes à microprocesseur. Les problèmes rencontrés sont identiques dans les deux cas et concernent le rendement d'un système complet après intégration. En effet, un tel circuit est en partie fait de blocs de grande taille dont le rendement est faible, et il est alors évident que si rien n'est fait pour l'améliorer, le rendement de tels circuits sera nul.

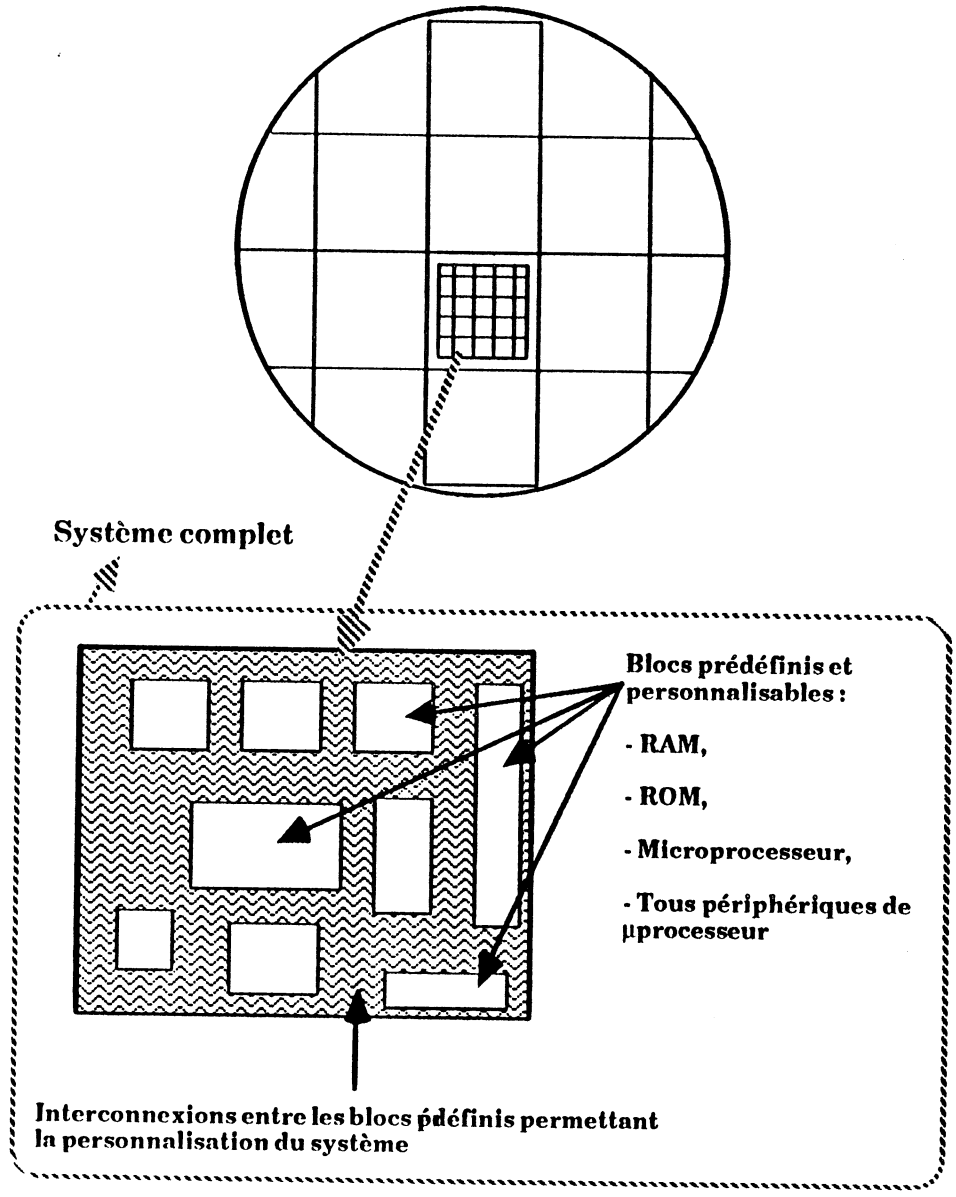


Figure 1.11 : Intégration d'un système complet sur un seul circuit intégré

Le seul moyen d'améliorer le rendement du circuit global est d'augmenter celui des blocs prédéfinis. Ceci est fait à l'aide d'une reconfiguration à deux niveaux :

- Au niveau du système (c.a.d. au niveau du circuit comprenant le microprocesseur et ses périphériques), des blocs de réserve sont intégrés pour remplacer ceux détruits par des défauts. Une telle approche n'est possible qu'avec des blocs de petite taille (périphériques) de façon à ne pas augmenter la taille du circuit dans des proportions qui annuleraient l'effet de la reconfiguration. Il est alors nécessaire de définir des stratégies de remplacement et de contournement des blocs en panne.

◆ Au niveau du bloc. Les grands blocs (Circuits à structure de microprocesseur, mémoires) sont ceux dont le rendement est le plus faible. Vu leur taille, il n'est pas possible de les répliquer entièrement et on doit donc y implanter de la réserve de façon à pouvoir tolérer les petits défauts de fabrication (sachant qu'un gros défaut rendrait un tel bloc irréparable et conduirait au rejet du circuit). Comme ces gros blocs sont ceux où une augmentation de rendement est primordiale, nous les avons étudiés plus particulièrement. Ils ont principalement deux types de structure : mémoire ou circuit de type microprocesseur. En ce qui concerne les mémoires, la reconfiguration a déjà été largement étudiée (voir § 2.1). La reconfiguration de circuits microprocesseurs n'a quant à elle jamais été étudiée. C'est sur ce dernier point que porte cette étude. Ce type de circuits comporte deux parties :

* Une partie opérative (PO) qui réalise les opérations sur les données.
* Une partie contrôle (PC) qui séquence et contrôle les actions de la PO. C'est elle qui réalise l'interprétation du jeu d'instructions du circuit.

Les stratégies de reconfiguration de ces deux parties présentent des aspects différents mais sont basées sur le même principe : on ne peut pas facilement corriger plus d'un défaut à la fois. Ceci amène à partitionner chaque partie en blocs où pas plus d'un défaut n'est susceptible d'apparaître.

3.3.2 - Reconfiguration d'un circuit de type microprocesseur

Il est important de bien définir notre objectif : il ne s'agit en aucun cas de vouloir réparer tous les défauts dans tous les circuits pour arriver à un rendement de 100%. Il est en effet évident que tous les circuits d'une tranche ne pourront être réparés, certains d'entre eux n'étant même pas testables. D'autre part certains défauts pourront empêcher toute reconfiguration. De plus, vouloir prévoir la réparation d'un grand nombre de défauts risque d'amener à augmenter la surface du circuit dans des proportions telles que l'effet de la reconfiguration sera annulé par la diminution du nombre de circuits et par l'augmentation de la probabilité pour ceux–ci d'être atteints par un défaut. Deux facteurs interviennent principalement dans la réparabilité d'un défaut :

- Sa *localisation* : un même défaut n'a pas les mêmes conséquences partout.
- Sa *taille* : ce facteur est très important pour la reconfigurabilité d'un circuit ; en effet on a vu précédemment que deux niveaux de reconfiguration sont nécessaires pour les petits et pour les gros défauts.

Le choix d'une stratégie de reconfiguration fait également intervenir d'autres paramètres tels que le nombre de pannes réparables dans un circuit, l'augmentation de surface autorisée pour la reconfiguration, le rendement que l'on souhaite atteindre ...

3.3.2.1 - Terminologie

Dans la suite de ce mémoire, on entendra par

Circuit : le microprocesseur pour lequel on cherche une stratégie de reconfiguration

Bloc : toute partie du circuit reconfigurable indépendamment des autres

Elément : un élément fonctionnel du circuit (un registre par exemple).

3.3.2.2 - Stratégie générale de reconfiguration

Un certain partitionnement est défini et du matériel de réserve est mis en place pour chaque élément de la partition. La mise en place d'éléments de réserve suppose que l'on puisse localiser précisément les défauts. Ceci est difficilement réalisable si on veut en corriger plusieurs à la fois. On va donc partitionner le circuit en blocs où pas plus d'une (ou deux) défaut(s) n'est susceptible d'apparaître. Ces blocs seront ensuite testés et reconfigurés individuellement.

Le choix de la stratégie de reconfiguration d'un microprocesseur revient alors à choisir le partitionnement du circuit en blocs et la méthode de reconfiguration de chaque bloc. Pour cela le premier paramètre nécessaire est la taille maximale d'un bloc reconfigurable. Les solutions à ces deux problèmes dépendent étroitement de l'architecture du circuit à reconfigurer.

Un circuit de type microprocesseur est constitué de deux parties principales : une partie opérative (P.O.) et une partie contrôle (P.C.), comme indiqué dans le schéma de la figure 1.12 ci–dessous.

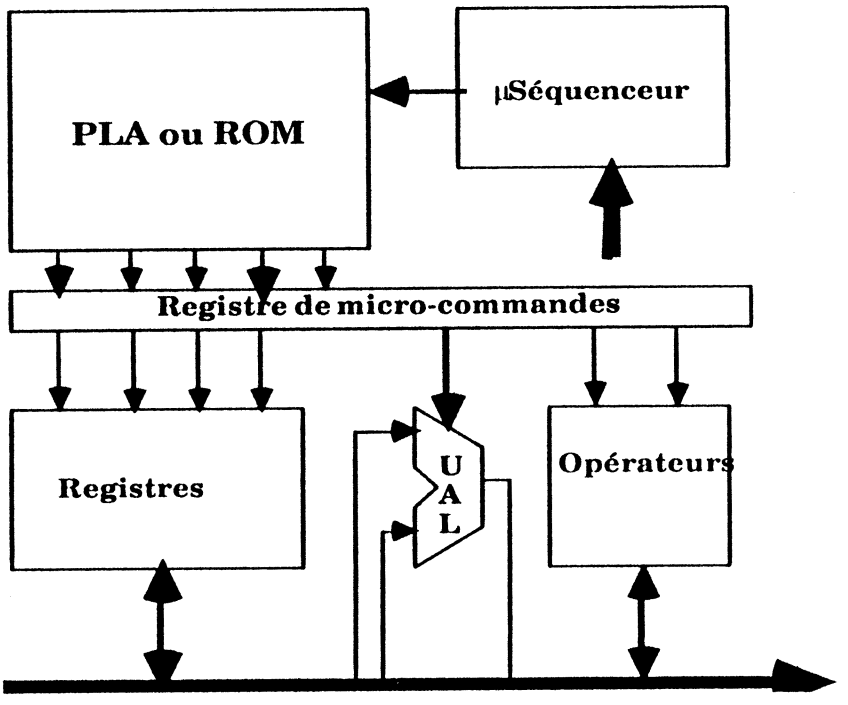


Figure 1.12 : Architecture générale d'un microprocesseur

La partie opérative a pour rôle de faire les traitements de données. Elle a une structure en tranches de bit et comporte différents éléments : registres et éléments de mémorisation divers, opérateurs arithmétiques et logiques (U.A.L, multiplieur,...), parties de logique combinatoire.

La tâche de la partie contrôle est de coordonner l'action des éléments de la P.O pour réaliser l'interprétation du jeu d'instructions du circuit. Cette partie a une structure totalement différente de la celle de la P.O. Il existe un grand nombre d'architectures possibles pour la P.C. [OBR82]. Elles vont des structures les plus irrégulières (organigramme câblé) aux plus régulières (micro–programmation, structures à base de PLAs (réseaux logiques programmables)).

Les problèmes posés par la reconfiguration de ces parties sont très différents. Dans les prochains chapitres, nous allons étudier les stratégies de reconfiguration adaptées à chacune de ces deux parties et fournir des critères de choix d'une solution.

Chapitre 2

Reconfiguration de la Partie Opérative d'un Microprocesseur

1 - Partitionnement

Le partitionnement de la partie opérative en blocs reconfigurables va être fait en tenant compte de sa structure. Comme nous l'avons vu précédemment, une partie opérative a une structure en tranches de un bit, c'est à dire qu'elle est composée de plusieurs tranches identiques de un bit communiquant parfois entre elles (avec des retenues par exemple) (figure 2.1).

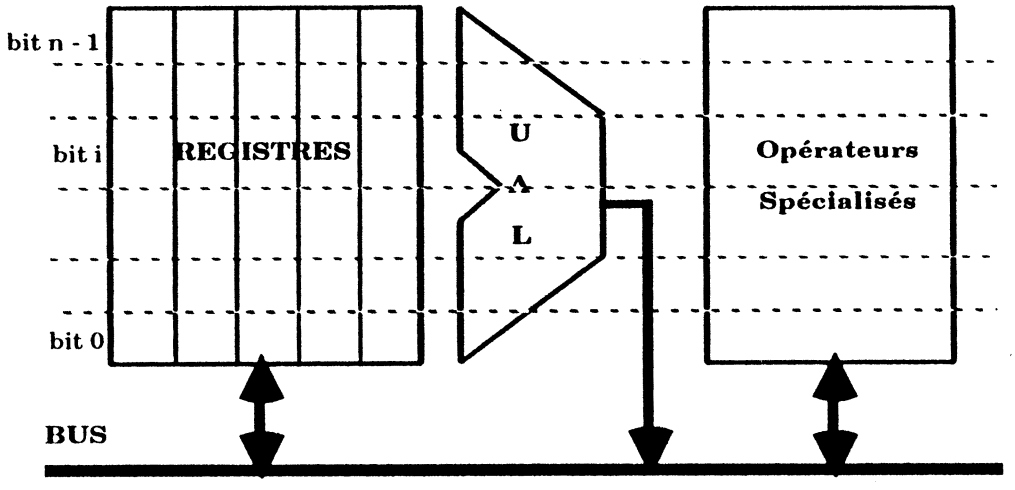


Figure 2.1 : Structure générale d'une partie opérative en tranches

A partir de cette structure de base de nombreuses variantes sont possibles sur le nombre de bus internes, les éléments fonctionnels de la PO, le découpage de la PO. En effet, on trouve de plus en plus de parties opératives pouvant être "découpées" en plusieurs morceaux travaillant indépendamment et se communiquant leurs résultats (traitement de données et calcul d'adresse comme dans le 68000 par exemple). Les éléments fonctionnels que l'on trouve généralement dans une partie opérative peuvent être classés en trois catégories principales :

- Registres, éléments de mémorisation
- Opérateurs (U.A.L, multiplieur,...)
- Blocs de logique aléatoire.

Les éléments des deux premières catégories ont une structure en tranches de bit où les tranches peuvent être interchangeables (c'est à dire sans signification particulière du bit i), ou bien avoir chacune un sens précis (par exemple comme dans un registre d'état). A partir de là on peut envisager deux

types de partitionnement pour la partie opérative :

- partitionnement fonctionnel
- partitionnement en tranches de bit.

1.1 - Partitionnement fonctionnel

Le partitionnement fonctionnel consiste à diviser la partie opérative en blocs reconfigurables en suivant la fonctionnalité des éléments la composant. On aura ainsi, par exemple, un bloc constitué de registres, un autre de un ou plusieurs opérateurs... Ayant été reconfiguré individuellement (par exemple avec des tranches de bit supplémentaires), chaque bloc est alors connecté à ses voisins (figure 2.2 ci–dessous). C'est à ce niveau que se situent les principales difficultés de ce type de partitionnement. On a en effet besoin ici de dispositifs de connexion entre les blocs (et avec les plots d'entrée/sorties), lesquels peuvent être très complexes et surtout d'une surface importante. D'autre part, la tolérance aux défauts par de tels dispositifs est très coûteuse, il est donc nécessaire d'en limiter l'encombrement. C'est dans cette optique qu'est proposée la deuxième manière de partitionner la partie opérative.

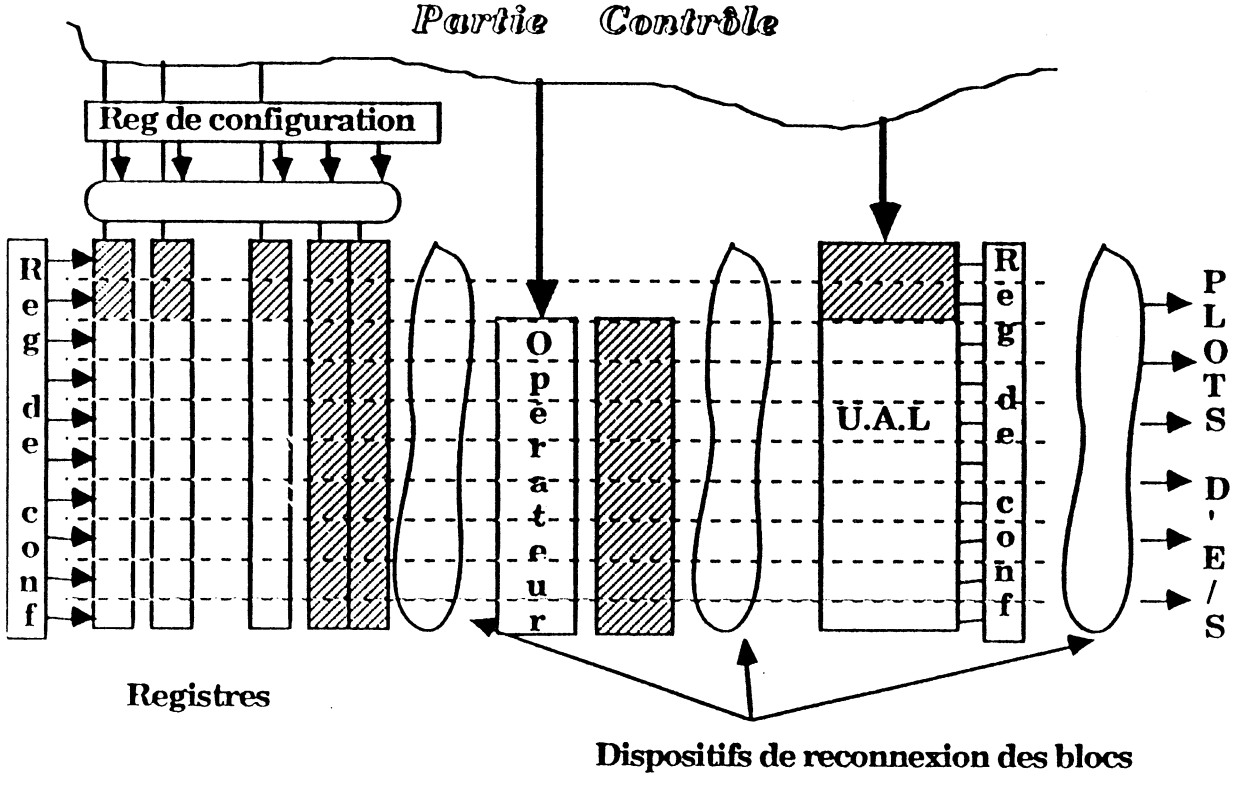


Figure 2.2 : Partitionnement fonctionnel de la P.O.

1.2 - Partitionnement en tranches de bit

Ce deuxième type de partitionnement utilise directement la structure en tranches de bit de la partie opérative. Au lieu de découper les blocs d'après leur fonction, on les forme en regroupant plusieurs tranches entières (ou éventuellement des parties de tranche si la P.O. est trop grande). Les possibilités de reconfiguration sont obtenues en ajoutant des tranches en réserve (figure 2.3a ci–dessous).

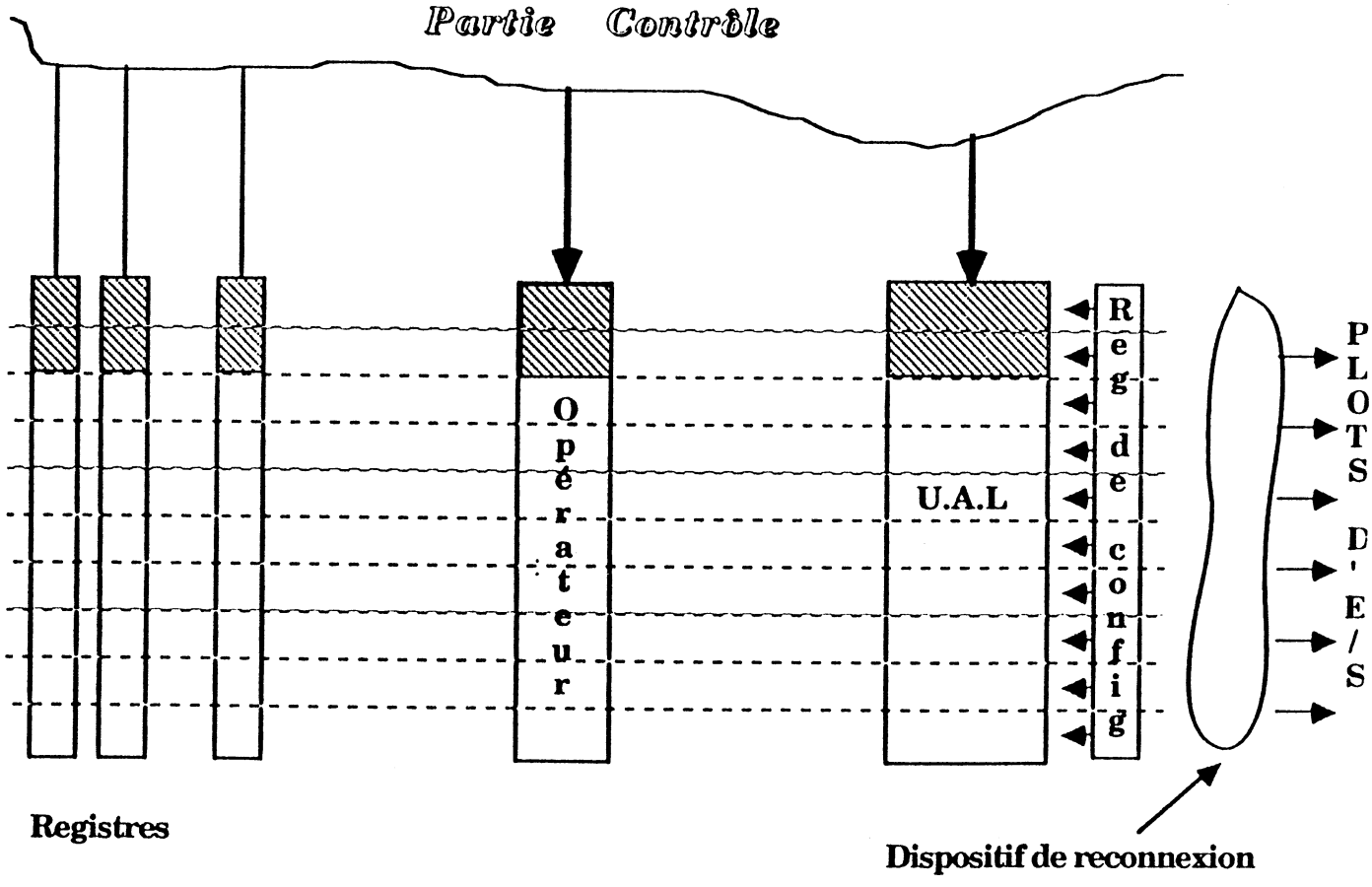


Figure 2.3 a : Partitionnement de la P.O. en tranches de bit

Une panne dans une tranche entraînera sa déconnexion et le décalages de toutes les tranches au dessus d'elle vers le haut (figure 2.3b). Les tranches supplémentaires sont identiques aux autres, et les seules modifications à faire (en dehors des dispositifs de reconnexion aux plots) consistent à prévoir le contournement des tranches inutilisées par les retenues les traversant. Le problème de l'interconnexion des blocs après reconfiguration est grandement simplifié. Le point dur de cette approche est le fait qu'aucune possibilité de reconfiguration n'existe pour les lignes de commande qui doivent atteindre toutes

les tranches y compris celles de réserve. Une des précautions à prendre dans ce cas, est de ne pas les dessiner aux dimensions minimales de la technologie lorsque cela est possible.

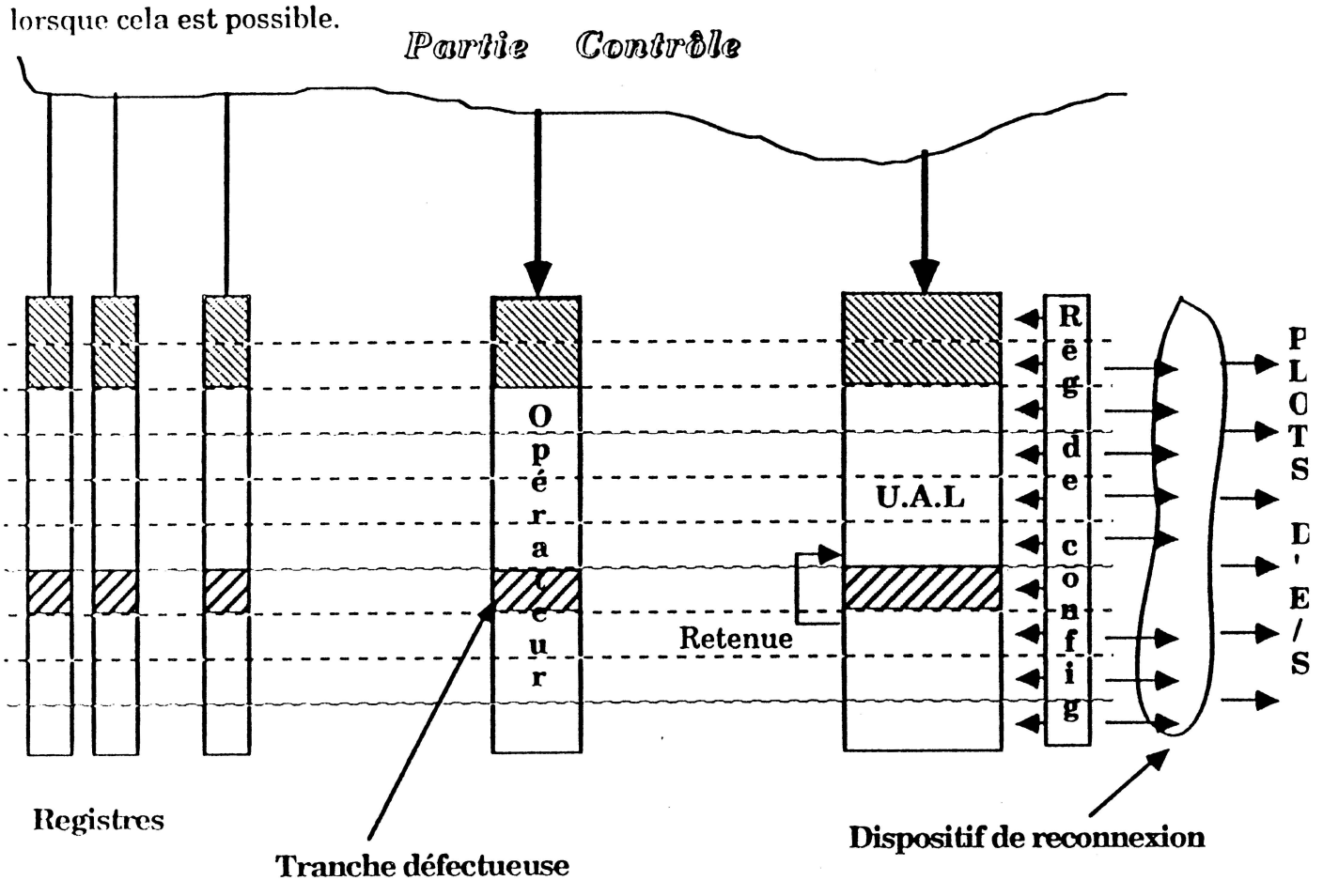


Figure 2.3b : Exemple de l'utilisation d'une tranche de réserve

2 - Choix d'une solution

2.1 - Critères de choix

Le choix de l'une ou l'autre stratégie de partitionnement et de reconfiguration se fait en évaluant la taille du circuit résultant pour chacune d'elles. On cherche à obtenir la solution offrant le plus de possibilités de reconfiguration pour l'augmentation de surface autorisée, c'est à dire que l'on va chercher à minimiser la place prise par les parties non directement "utiles" (commutateurs, dispositifs de reconnexion) dans ce qui est ajouté pour la reconfiguration. Si T_u est la taille des dispositifs directement "utiles" dans ce qui est ajouté, et T_{Aj} celle de tout ce qui a été ajouté pour la reconfiguration, on cherchera à maximiser le rapport : $R = T_u / T_{Aj}$ tout en restant dans les limites de l'augmentation de surface autorisée.

Une fois qu'une solution est choisie, on améliore le rapport taille des dispositifs de réserve sur taille du dispositif de reconfiguration complet en modifiant le contenu des blocs reconfigurables et éventuellement l'architecture de certains éléments de la partie opérative.

L'évaluation est faite à un niveau architectural, au moment de la conception du circuit. Cela implique que les tailles de circuit soient exprimées en nombre de transistors avec des facteurs correctifs venant de l'expérience du dessin de telle ou telle structure particulière, ou de la proportion d'interconnexions par rapport au nombre de transistors dans le circuit.

Le but d'une telle évaluation est de permettre le choix d'une solution en comparant plusieurs possibilités ; le niveau auquel elle est faite implique que ces calculs ne peuvent prétendre donner une valeur exacte de la surface du circuit (car ils sont faits avant la réalisation du circuit). Les résultats qui sont présentés dans ce qui suit concernent les technologies CMOS.

2.2 - Taille des portes de base

On va donner ici la taille des quelques portes classiques qui sont utilisées dans les différents dispositifs de réorientation et de connexion rendus nécessaires par la reconfiguration.

*** Porte de transfert**

Une porte de transfert CMOS comporte deux transistors : **T = 2**

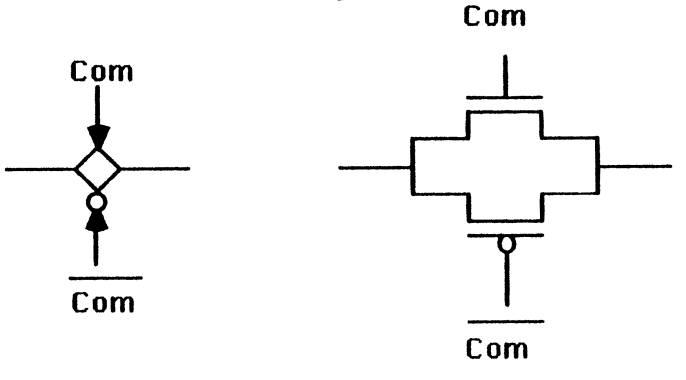


Figure 2.4 : Porte de transfert

*** Inverseur, Portes NAND et NOR**

Un inverseur CMOS comporte deux transistors : **T = 2**

Les portes NAND ou NOR comportent chacune deux transistors par entrée.

Si $\mathbf{n_e}$ est le nombre d'entrées d'une de ces portes, on $\mathbf{T = 2 . n_e}$

*** Eléments de mémorisation**

Un élément de mémorisation (point de registre) est composé de deux inverseurs rebouclés. Il peut être entouré de diverses portes en contrôlant l'accès ; celles–ci seront explicitement spécifiées lorsqu'on les emploiera. Dans certains cas il peut être nécessaire d'utiliser des bascules avec mise à zéro ou à un ; elles seront traitées au cas par cas. La taille d'un élément de mémorisation de base est donc **T = 4.**

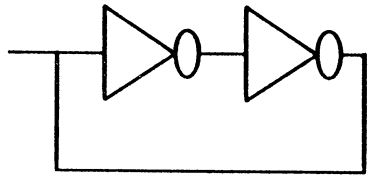


Figure 2.5 : Elément de mémorisation

3 - Reconfiguration des blocs

Nous considérons ici des blocs comprenant une ou plusieurs unités fonctionnelles (selon qu'on utilise un partitionnement fonctionnel ou un partitionnement en tranches). La taille d'un bloc reconfigurable est donc la somme des tailles des blocs le composant : soit T_{Ri} la taille de l'élément **i** avec ses dispositifs de reconfiguration, on a le nombre de transistors du bloc (c'est à dire sa taille) qui est :

$$(2.1) : \quad T_B = \sum_{I \in \{ \text{éléments du bloc} \}} T_{R_I}$$

Avec T_i la taille initiale de l'élément **i** et S_{Ri} la taille des dispositifs de reconfiguration qui lui sont ajoutés, on a :

$$(2.2) \quad T_{Ri} = T_i + S_{Ri}$$

Les méthodes de reconfiguration qui vont être présentées ci-dessous concernent les éléments composant un bloc, et l'évaluation de chaque solution est faite en calculant le T_{Ri} correspondant. On peut distinguer trois façons de rendre un élément reconfigurable :

- par réplication massive

- par ajout de modules de réserve pour les éléments modulaires

- par ajout de tranches de bit supplémentaires.

3.1 - Réplication massive d'un élément

Il s'agit de répliquer tout ou partie d'un élément afin de pouvoir le remplacer s'il venait à être atteint par un défaut. Cette forme de redondance est bien adaptée à de petits éléments critiques dont la réplication totale est négligeable quant à son influence au niveau de l'augmentation de surface du circuit. Dans ce cas , la taille de l'élément après réplication sera :

$$(2.3) \qquad T_{Ri} = (1 + m)\ T_i$$

où **m** est le facteur de réplication : **m < 1** : réplication partielle

m > 1 : réplication totale une ou plusieurs fois.

Cette évaluation n'est toutefois pas complète, en effet, il faut également prendre en compte les dispositifs de connexion rendus nécessaires par la réplication pour isoler l'élément remplacé :

- Si l'élément reçoit des commandes, celles–ci devront pouvoir être déconnectées et orientées vers la réserve.

- Il doit également être possible de réorienter les entrées/sorties de l'élément.

Deux cas sont possibles :

* *élément dupliqué :*

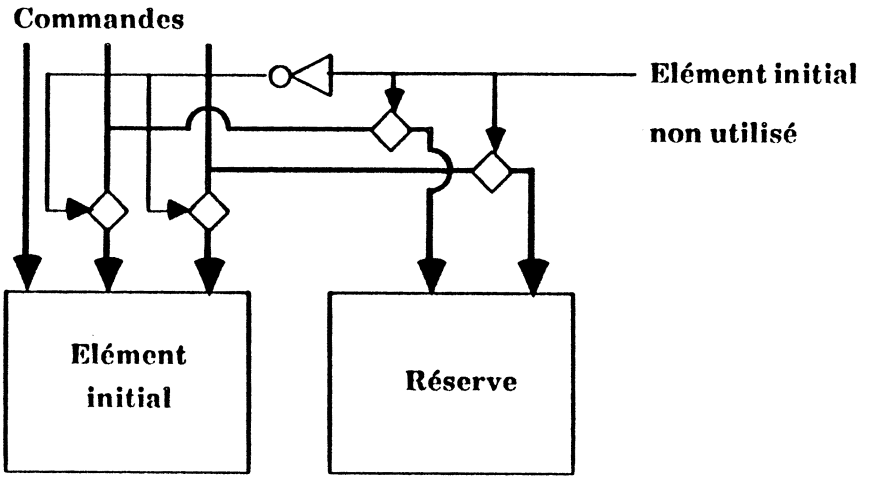


Figure 2.6 : Réorientation des lignes de commande ou d'E/S pour un élément dupliqué

Il faut deux portes de transfert par ligne à réorienter ainsi qu'un inverseur pour le signal *"Elément initial non utilisé"* . Soit L_i le nombre de lignes à réorienter pour l'élément **i**, on a :

$$(2.4) \qquad T_{Ri} = (1 + m) . T_i + 2 (2 L_i + 1)$$

* *élément répliqué plus d'une fois :*

Dans ce cas, la réorientation des lignes est faite comme indiqué dans la figure 2.7.

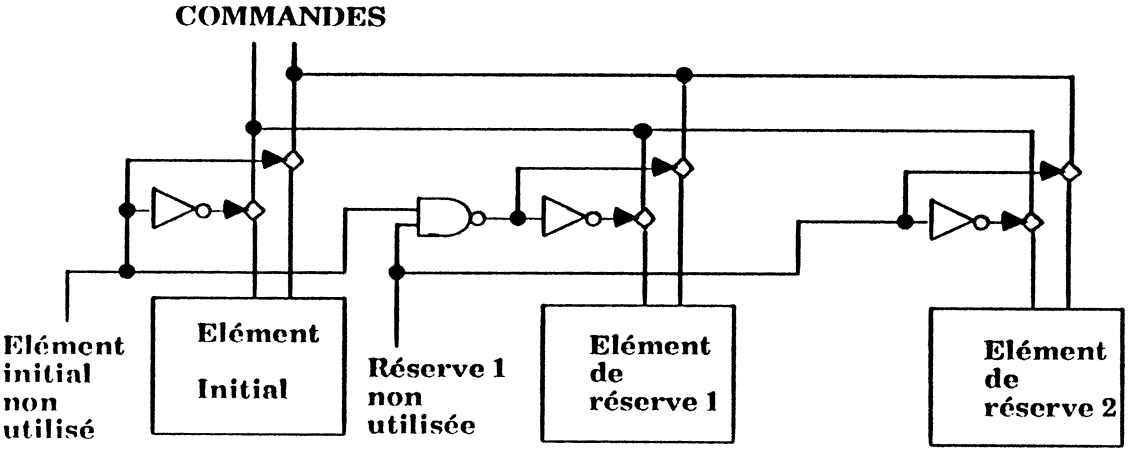


Figure 2.7 : réorientation des lignes de commande ou d'E/S dans le cas d'un élément répliqué plusieurs fois

Soient n_R le nombre de lignes à réorienter,

n_{Rj} le nombre de réorientations possibles pour la ligne **j**,

on a alors :

(2.5) :

$$T_{R_i} = (1+n)T_i + 2(n_R+1) + 6\left[\max_{l \in \{n_R \text{ lignes réorientées}\}} (n_{R_l}) - 1\right] + 2 + \sum_{l \in \{n_R \text{ lignes réorientées}\}} 2.n_{R_l}$$

réplication des blocs	bloc de base	portes NAND et inverseurs	dernier élément de réserve	portes de transfert

Il est également nécessaire de pouvoir mémoriser l'état de l'élément (utilisé ou non). Différentes possibilités existent pour cela qui seront détaillées au paragraphe 4.

3.2 - Ajout de réserve pour les éléments modulaires

On désigne par éléments modulaires ceux qui sont composés de plusieurs modules identiques, comme par exemple les registres. La reconfiguration d'une telle structure peut être obtenue en ajoutant plusieurs modules de réserve vers lesquels les commandes des modules défectueux seront réorientées (figure 2.8 ci–dessous dans le cas de l'adjonction de deux modules supplémentaires).

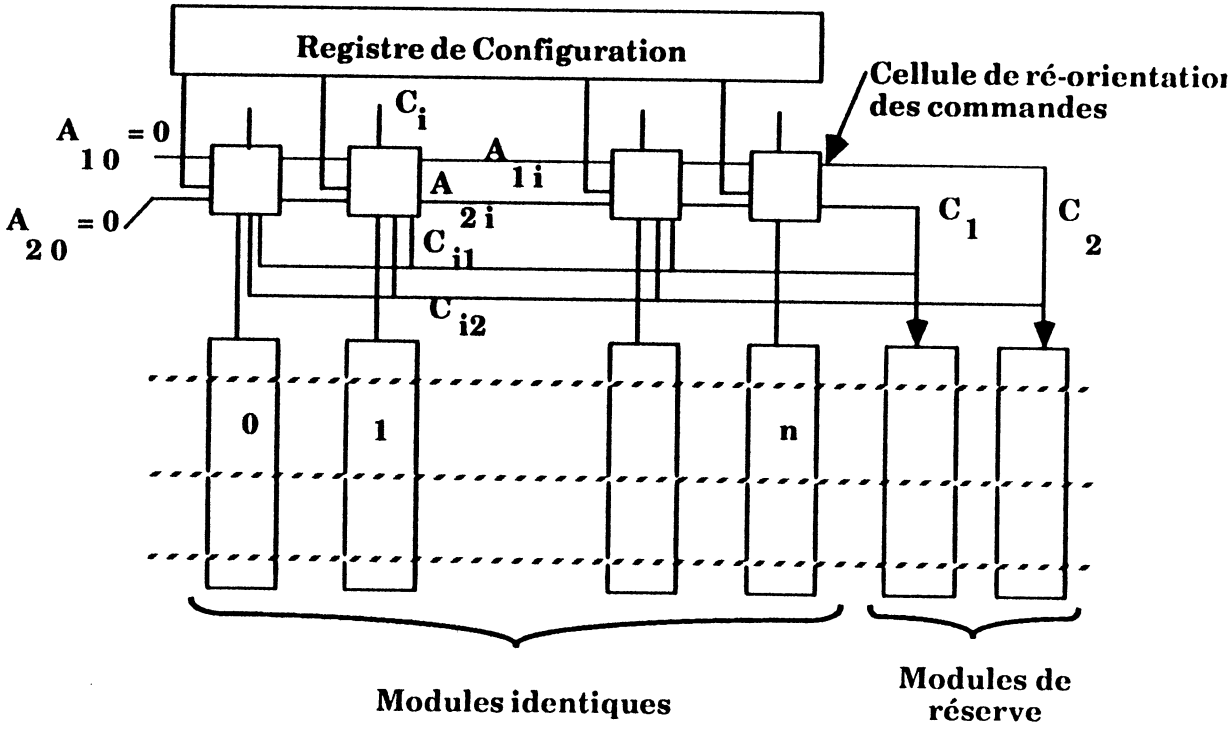


Figure 2.8 : Reconfiguration par ajout de deux modules en réserve

L'évaluation sera faite dans le cas de l'adjonction de deux modules supplémentaires, l'adjonction de plus de modules en réserve augmentant assez fortement la complexité de la cellule de ré–orientation ce qui diminue alors sensiblement l'intérêt de cette méthode de reconfiguration.

* ***Cellule de réorientation des commandes*** **:**

Les commandes d'un module inutilisable sont réorientées vers le premier module de réserve. Si celui-ci est déjà utilisé (information A_{1i} à un), on oriente les commandes vers le deuxième module de réserve. Ceci est réalisé par la cellule présentée dans la figure 2.9 ci-dessous.

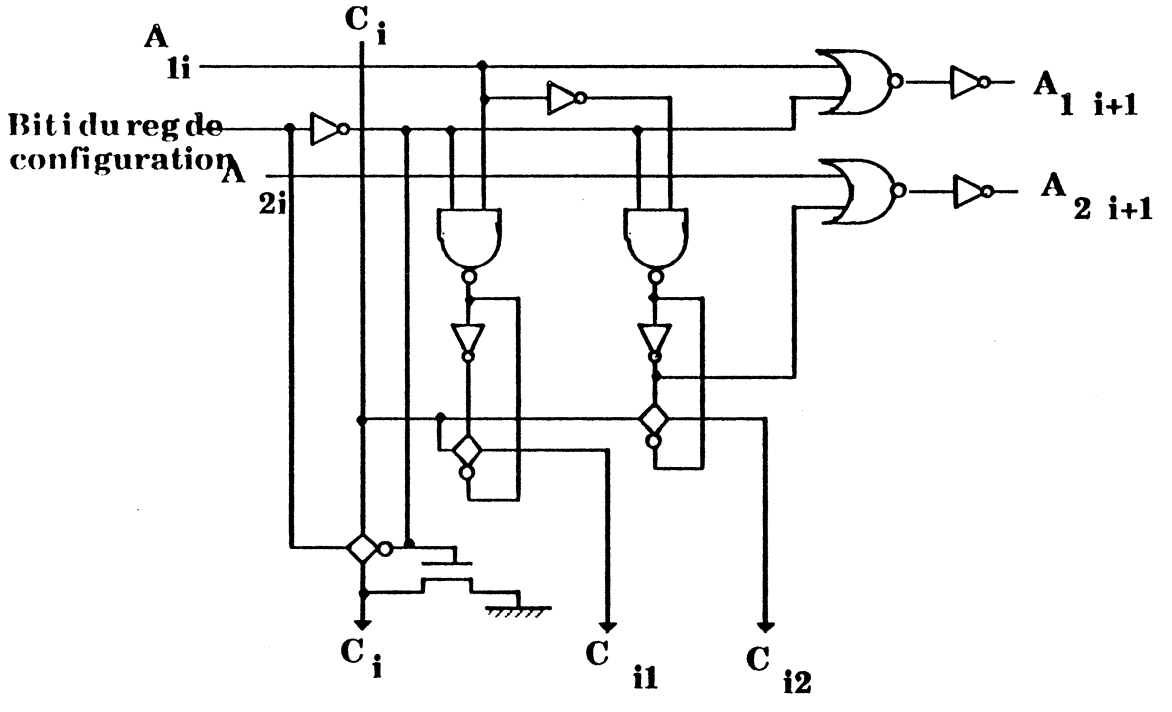


Figure 2.9 : Schéma d'une cellule de réorientation des commandes

Il y a néanmoins un cas particulier pour la cellule **n**, en effet, si tous les modules de réserve ne sont pas utilisés, il faut mettre à zéro les commandes de ceux qui sont inutilisés (de façon à ne pas risquer d'avoir des interférences intempestives en fonctionnement normal). Ceci est réalisé en modifiant la dernières cellule de ré–orientation comme indiqué dans la figure 2.10 ci-après.

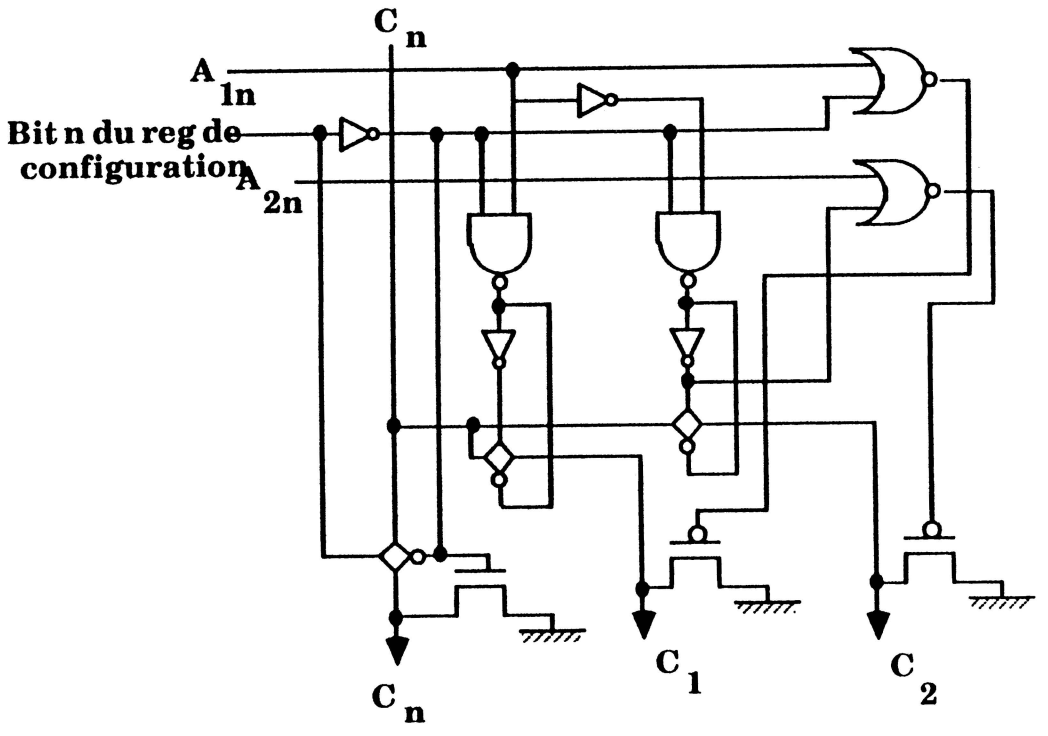


Figure 2.10 : Schéma de la dernière cellule de réorientation des commandes

* ***Evaluation de la taille du dispositif de réorientation des commandes*** :

- Génération des commandes des portes de transfert :

Cellule "normale" : on a deux portes NAND, deux portes NOR à deux entrées (soit quatre transistors par porte), et six inverseurs, ce qui donne un total de **28** transistors

Dernière cellule : on a toujours quatre portes NAND et NOR à deux entrées, mais seulement quatre inverseurs, ce qui donne un total de **24** transistors.

- Portes de transfert et transistors de mise à la masse :

Soient n_{cm} le nombre de commandes à réorienter par module, et n_m le nombre de modules.

Cas des n_m - 1 premiers modules :

Pour chaque commande on a une porte de transfert vers le destinataire normal de la commande accompagnée d'un transistor N de tirage à la masse (en cas de non utilisation du module), et une autre porte de transfert vers chacun des deux modules de réserve, ce qui fait **7** transistors.

Cas du dernier module :

On a la même configuration que ci-dessus avec en plus deux transistors P de tirage à la masse des lignes dirigeant les commandes vers les modules de réserve, ce qui fait 9 transistors.

- Evaluation globale :

On aura donc, pour le dispositif de réorientation des commandes :

$28 (n_m - 1) + 24 + 7 n_{cm} (n_m - 1) + 9 n_{cm}$ transistors.

* ***Evaluation de la taille du bloc après ajout d'éléments de réserve :***

L'ajout de deux modules de réserve va entrer pour $2 T_i / n_m$ dans le résultat final qui est :

$$(2.6) \qquad T_{Ri} = T_i (1 + 2/n_m) + 7 (n_m - 1) (n_{cm} + 4) + 3 (3 n_{cm} + 8)$$

Remarque :

Il n'a pas été tenu compte dans cette évaluation du "registre de configuration". Cet élément indispensable peut être réalisé de plusieurs façons qui seront abordées au paragraphe 4.

3.3 - Ajout de tranches de bit supplémentaires

Cette solution utilise la structure en tranches des éléments de la partie opérative. La principale modification intervenant alors au niveau de l'élément consiste à prévoir la possibilité, pour les données propagées à travers les tranches (par exemple les retenues), de ne pas être modifiées lors de leur passage à travers une tranche inutilisée (c'est à dire quand l'indicateur "tranche non utilisée est à un). Ceci est réalisé de différentes manières selon qu'on a des propagations dans un seul ou bien dans plusieurs sens.

3.3.1 - Propagation de données dans un seul sens

Dans ce cas, le contournement d'une tranche non utilisée se fait comme indiqué dans la figure 2.11.

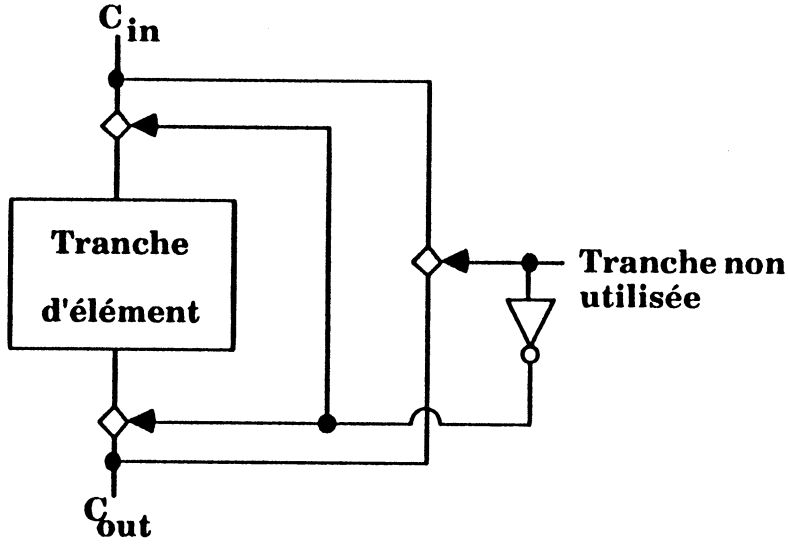


Figure 2.11 : Propagation de données à travers les tranches non utilisées

Ceci nécessite **8** transistors pour la première retenue et **6** pour les autres (car un seul inverseur est nécessaire). Il est également nécessaire de mémoriser l'état de chaque tranche (utilisée ou non). Soient n_b le nombre de bits de la partie opérative (sans reconfiguration), **k** le nombre de tranches de réserve, et n_{re} le nombre de retenues devant contourner les tranches non utilisées, on a :

$T_{Ri} = T_i (1 + k/n_b) + (6 (n_{re} - 1) + 8) (k + n_b)$ <=>

(2.7) $\mathbf{T_{Ri} = T_i (1 + k/n_b) + 2 (3 n_{re} + 1) (k + n_b)}$

3.3.2 - Propagations de données dans plusieurs directions

Ce type de propagation se rencontre dans des opérateurs matriciels tels que les multiplieurs parallèles cellulaires dans lesquels les propagations verticale et horizontale des retenues sont complétées par l'utilisation d'un même bit d'opérande par toutes les cellules situées sur une même diagonale.

Lorsqu'une tranche défectueuse est remplacée (par exemple en décalant vers le bas), le problème des données circulant diagonalement (qui doivent être descendues comme montré par le schéma de la figure 2.12) vient s'ajouter à celui (traité précédemment) des données propagées verticalement. Il faut également prendre en compte le nécessaire contournement des cellules inutilisées du bord (côté sorties) par les résultats et les retenues horizontales.

La reconfiguration d'un tel opérateur a été étudiée à travers l'exemple d'un multiplieur parallèle **i X i -> i** bits réalisant l'opération A*B+C+D dont le schéma général est donné dans la figure 2.12 ci-dessous.

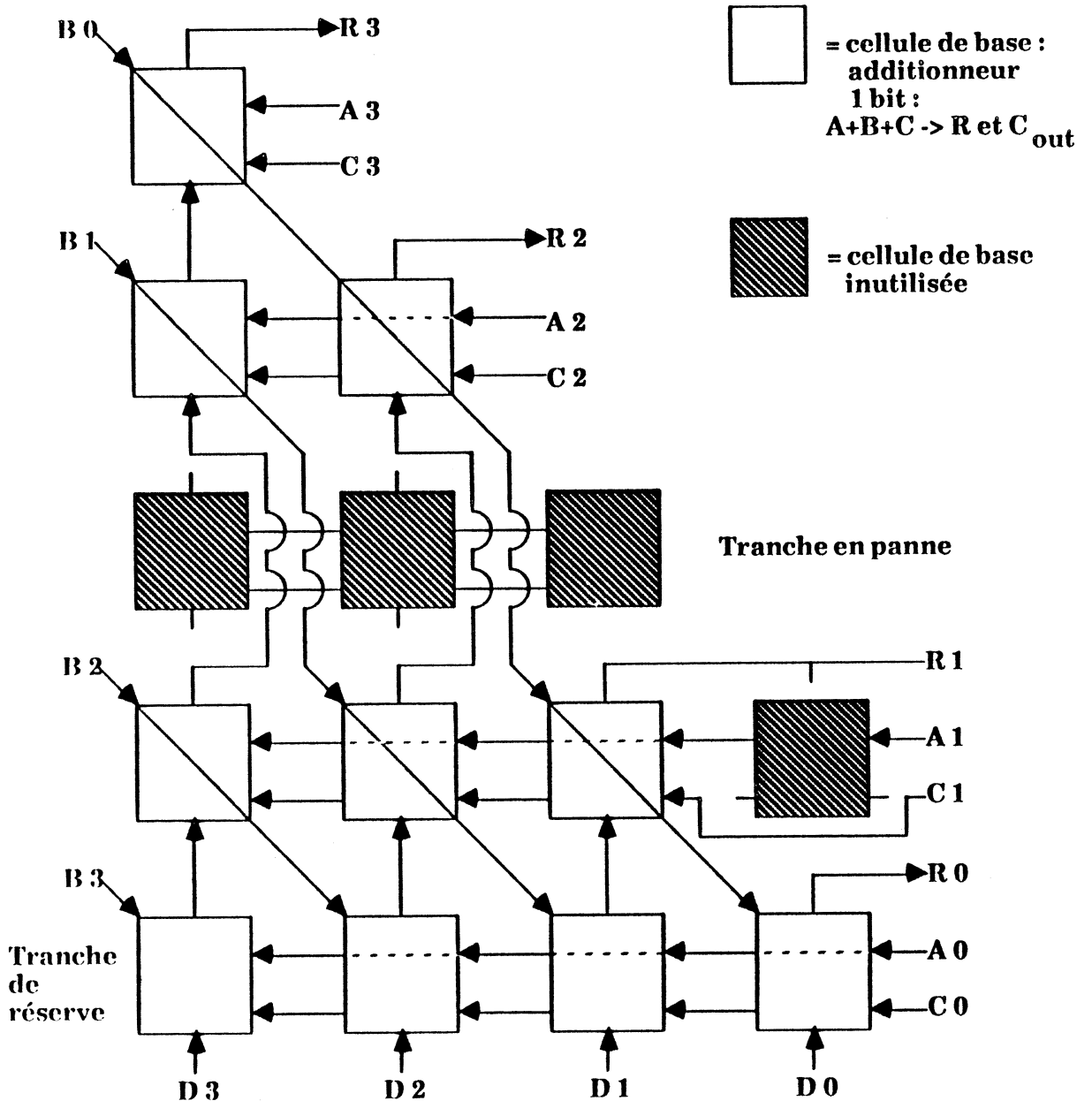


Figure 2.12 : Reconfiguration d'un opérateur matriciel (exemple d'un multiplieur parallèle)

* ***Evaluation des dispositifs de contournement :***

- Propagation des données verticales et horizontales : le contournement par ces données de tranches inutilisées est réalisé de la même manière que pour une propagation de données dans un seul sens (§ 3.3.1). L'évaluation des dispositifs nécessaires se fait alors de la manière suivante :

Données verticales :

Soient n_c le nombre de cellules de l'opérateur, n_v le nombre de données verticales à propager, on a :

(2.8) $T_v = 6\, n_c \,.\, n_v + 2\,(n_b + k)$

portes de transfert — inverseurs des commandes (un par ligne)

Données horizontales :

Ces données n'ont à contourner que les **k** cellules du côté des sorties (ou toutes les cellules pour les **k** premières tranches), d'où, avec n_h le nombre de données horizontales entre deux cellules, on a :

$T_h = n_h \,.\, 6\,(k\,(k+1)/2 + k \,.\, n_b) + 2\,(n_b + k)$ <=>

(2.9) $$T_h = 2\left[k\left(3 \,.\, n_h\left(\frac{k+1}{2} + n_b\right) + 1\right) + n_b\right]$$

- Contournement des cellules par les résultats :

Il est réalisé comme indiqué dans la figure 2.13 pour trois tranches de réserve (donc trois cellules à contourner) ci-après et porte sur les cellules qui reçoivent déjà un dispositif de contournement pour les données horizontales.

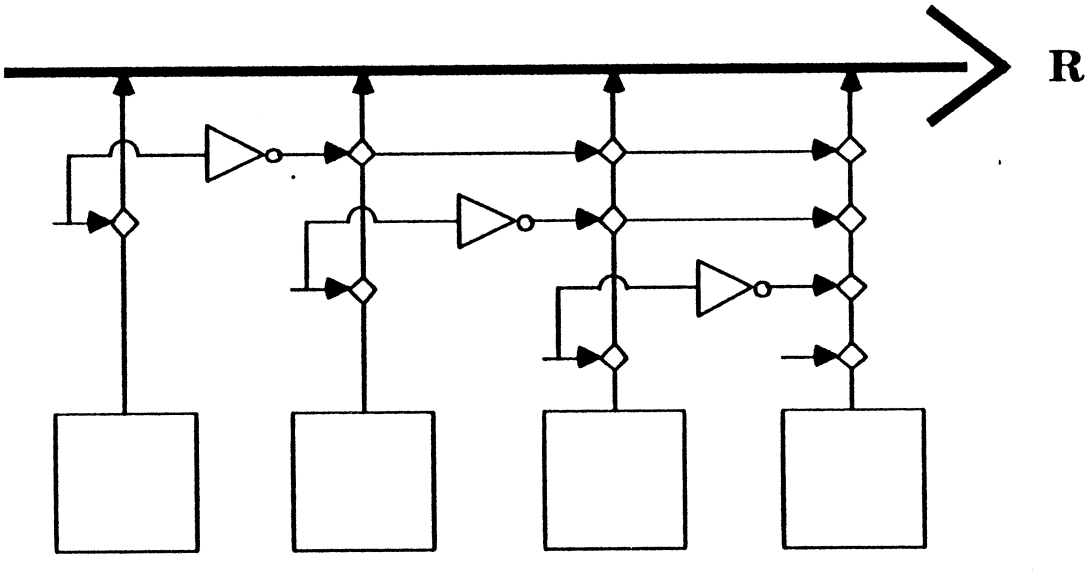


Figure 2.13 : Contournement de cellules pour les résultats

Les commandes étant les mêmes que celles nécessaires pour le contournement par les données circulant horizontalement, il n'y aura pas besoin de prévoir des inverseurs de commandes ici.

Pour les n_b **dernières tranches** il faudra :

$\sum_{j=1}^{k} j$ portes de transfert par tranche

Pour les **k premières tranches** il faudra :

$\sum_{i=1}^{k}\left(\sum_{j=1}^{i} j\right)$ portes de transfert

$= \sum_{i=1}^{k}\left[\frac{i\,(i+1)}{2}\right]$ portes de transfert

D'où on a, si n_s est le nombre de bits de résultat ayant à être propagés sur chaque tranche :

$$(2.10) \qquad T_s = 2\,n_s\left(\sum_{i=1}^{k}\left[\frac{i\,(i+1)}{2}\right] + n_b\,\frac{k\,(k+1)}{2}\right)$$

+ 2 n_b pour les inverseurs des commandes des portes de transfert s'il n'y a pas de données horizontales à propager.

- Contournement de cellules pour les données circulant en diagonale

Ce type de contournement est réalisé comme montré par la figure 2.14 ci–dessous.

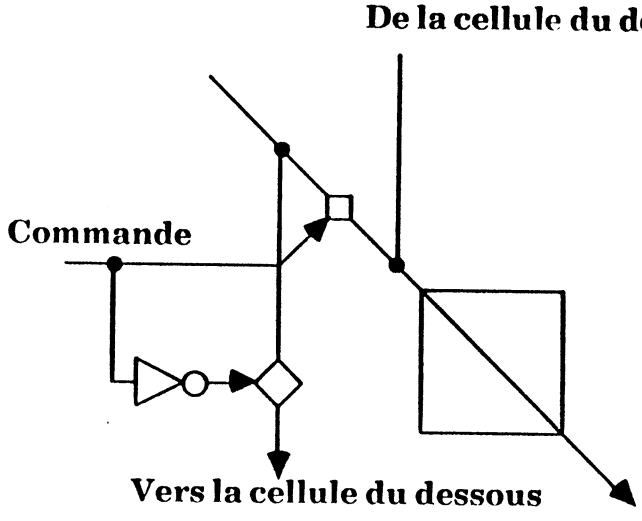


Figure 2.14 : Dispositif de contournement de cellule pour les données circulant en diagonale

Soit n_d le nombre de données circulant en diagonale, on a :

$$(2.11) \qquad T_d = 2\,n_c\,(2\,n_d + 1)$$

Si n_{ch} est le nombre maximal de cellules dans une tranche, comme on ne peut ajouter en réserve que des tranches comportant n_{ch} cellules, on aura pour un opérateur matriciel :

$$(2.12) \quad T_{Ri} = T_i (1 + k . n_{ch} / n_c) + T_v + T_h + T_s + T_d$$

Remarque :

Il faut noter que, vu la complexité des dispositifs nécessaires à la reconfiguration d'éléments matriciels quand tous les types de contournement doivent être employés, l'utilisation de tels éléments ne pourra raisonnablement s'envisager que si la taille des cellules est suffisamment grande devant celle des dits dispositifs. Dans le cas contraire, il parait sage d'envisager une autre architecture pour l'élément concerné. De plus, il ne faut pas négliger la complexité de la génération de toutes les commandes nécessaires au bon contrôle des dispositifs de contournement.

3.4 - Limitations

L 'ajout de tranches de réserve suppose que celles-ci sont interchangeables, or ce n'est pas toujours le cas. En effet, les tranches de certains éléments (registre d'état par exemple) peuvent avoir une signification précise. Dans ce cas l'élément ne peut pas être reconfiguré autrement que par réplication totale (ou bien ne pas être reconfiguré du tout), et dans le cas d'un partitionnement en tranches on doit le reconnecter au reste de la partie opérative (qui aura lui des tranches de bit supplémentaires). Ceci nous amène au dernier problème posé par l'introduction de réserve dans une partie opérative de microprocesseur : la connexion des différents blocs reconfigurables entre eux et avec les plots d'entrée/sortie après configuration.

4 - Connexion des blocs reconfigurables

4.1 - Description du problème

Lors de la reconfiguration d'une partie opérative de microprocesseur, les problèmes de connexion entre ou dans les blocs reconfigurables sont rencontrés dans trois situations :

- Connexion entre un bloc ayant des tranches de bit supplémentaires et un autre n'en ayant pas ou en ayant un nombre différent.

- Connexion entre la partie opérative ayant des tranches de bit supplémentaires et les plots d'entrées/sorties dont le nombre est fixe.
- Connexion entre éléments reconfigurés à l'aide de tranches de bit supplémentaires et éléments non reconfigurables à l'intérieur d'un même bloc.

Ces différentes situations ramènent toutes au même problème, à savoir la connexion de deux blocs de partie opérative n'ayant pas le même nombre de tranches (figure 2.15 ci–dessous).

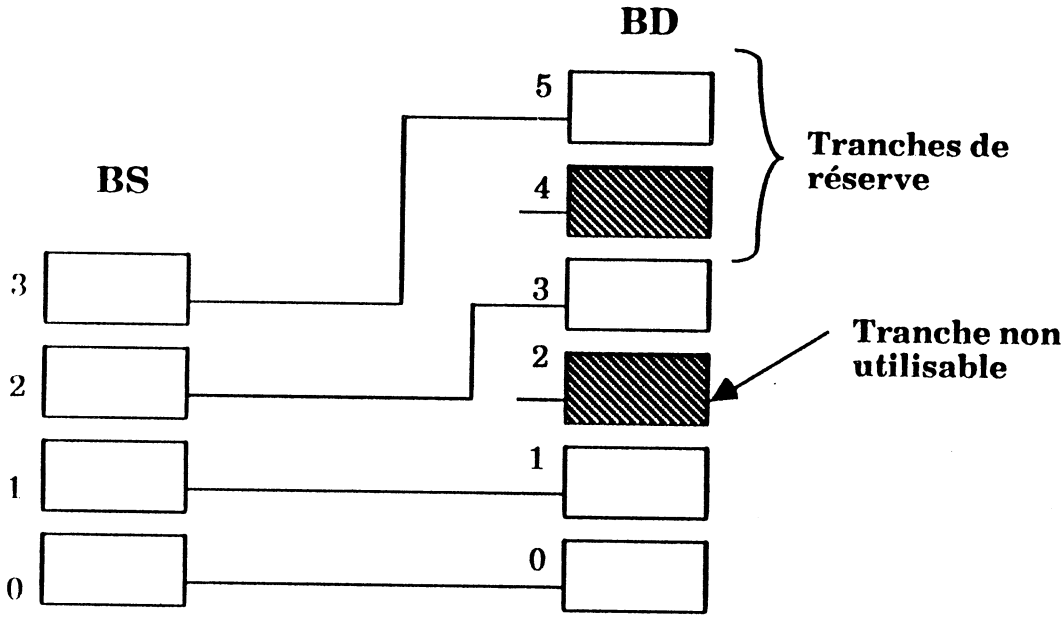


Figure 2.15 : Principe de la connexion de deux blocs n'ayant pas le même nombre de tranches de bit

Deux cas sont alors possibles selon que les deux blocs ont tous les deux des tranches supplémentaires ou non.

* ***Connexion d'un bloc de P.O sans tranches supplémentaires BS (bloc source) à un bloc avec tranches de réserve BD (bloc destination) :***

Si on a **K** tranches de réserve dans BD la connexion se fera comme suit : on essaye de connecter chaque tranche de BS à son vis à vis dans BD, et lorsque celui–ci n'est pas utilisable, on fait la connexion avec la tranche d'indice supérieur (dans la limite des K tranches supplémentaires). Les différentes connexions à prévoir doivent permettre de relier toute tranche **i** de BS à la tranche **i** de BD ainsi qu'aux tranches **i+1** à **i+k** de BD comme indiqué dans la figure 2.16 ci–dessous.

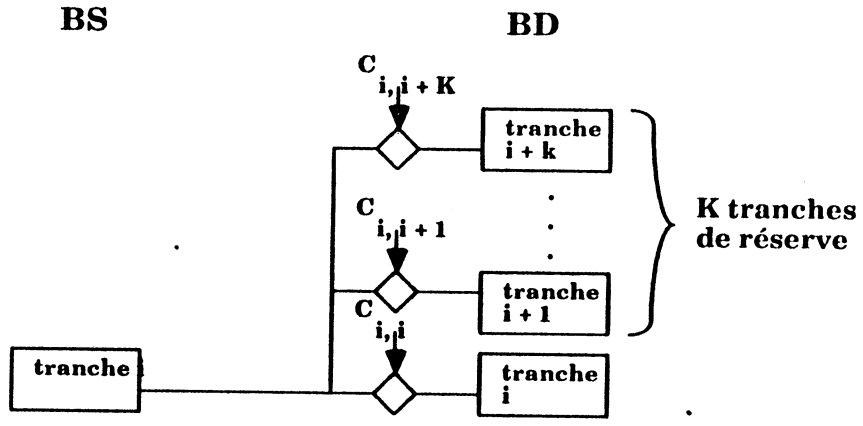


Figure 2.16 : Connexion entre un bloc ayant des tranches supplémentaires et un bloc n'en ayant pas

* ***Connexion de deux blocs de P.O ayant tous deux des tranches de bit supplémentaires :***

Dans ce cas, avec K_s tranches de réserve pour **BS** et K_d pour **BD**, le problème est différent : ici toutes les tranches de BS ne sont pas forcément utilisables, donc si, par exemple, la tranche 1 de BS est en panne, elle sera remplacée par la tranche 2 qui devra alors être connectée à la tranche de BD si celle–ci est saine. On voit que dans ce cas, une tranche **i** de BS devra être connectée non seulement aux tranches **i** à **i+K_d** de BD, mais également aux tranches **i-K_s** à **i-1** (figure 2.17 ci–dessous).

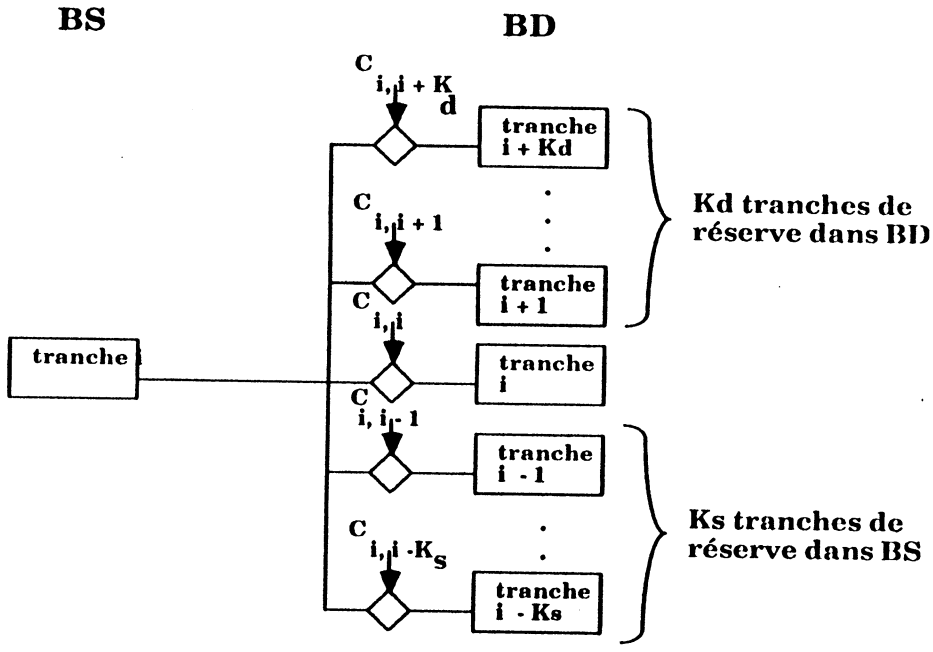


Figure 2.17 : Connexion de deux blocs ayant chacun des tranches supplémentaires

4.2 - Réalisation des différentes possibilités de connexion

Deux choses sont nécessaires à la réalisation du dispositif de connexion :

- les commutateurs permettant la connexion,
- les commandes de contrôle de ces commutateurs.

De plus, dans le cas où on a plus d'une tranche en réserve, ces dispositifs doivent pouvoir être reprogrammés pour pouvoir tester plusieurs configurations. Comme nous l'avons vu au chapitre 1 § 2.2, il existe principalement trois types de commutateurs : les fusibles et anti–fusibles programmés par laser, les transistors à grille flottante programmés par faisceau d'électrons, et les systèmes programmés par logique.

4.2.1 - Utilisation de fusibles et d'anti-fusibles

L'utilisation d'anti-fusibles implique que l'on travaille avec le deuxième niveau de métal. Celui–ci sert généralement à réaliser les lignes d'alimentation et les lignes de bus dans la partie opérative. Ceci impose que la connexion entre BS et BD se fasse au niveau des fils de bus. A part le lien initial, chacune des connexions possibles doit pouvoir être coupée après avoir été créée. L'ouverture de la passivation nécessaire pour pouvoir créer un anti–fusible rend impossible sa destruction ultérieure (car pour couper un fusible, il ne faut aucune ouverture de la passivation au dessus). On devra donc mettre en série un fusible et un anti–fusible pour obtenir un lien qu'il soit possible de créer et ensuite (éventuellement) de détruire (figure 2.18).

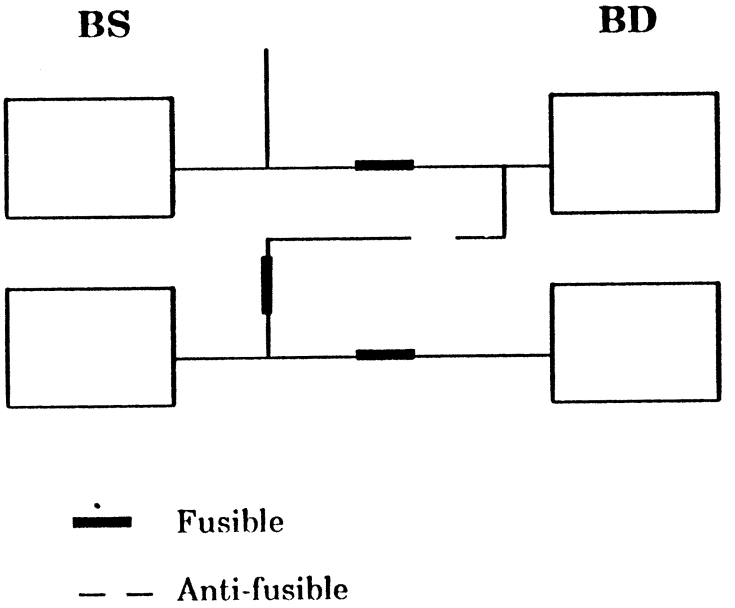


Figure 2.18 : Connexion de blocs à l'aide de fusibles et d'anti–fusibles

Evaluation de cette solution :

Soient T_F la taille (nombre équivalent de transistors d'un point de vue surface) d'un **fusible**, et T_{AF} celle d'un anti–fusible. Soit n_l le nombre de lignes de bus à connecter, il faudra pour chacune d'elles :

- Un fusible pour supprimer la liaison initiale
- Un ensemble fusible/anti-fusible pour chaque lien avec une tranche autre que celle de la liaison initiale.

- **Seul BD a des tranches supplémentaires :**

Soit **K** le nombre de tranches supplémentaires. On aura alors la taille du dispositif de reconfiguration :

$$(2.13) \quad T_{cl} = n_b \cdot n_l \, [K \, (T_{AF} + T_F) + T_F]$$

- **BS et BD ont tous les deux des tranches supplémentaires :**

Dans le cas le plus général, BD et BS ont tous les deux des tranches supplémentaires, et on aura alors :

* *Pour les K_s premières tranches de BS :*

K_s fusibles pour les liaisons initiales.

$K_s \cdot K_d + \sum_{j=0}^{K_s - 1} j$ ensembles fusible/anti-fusible,

soit $(2 K_d + K_s + 1) K_s / 2$ fusibles et $(2 K_d + K_s - 1) K_s / 2$ anti-fusibles.

* *Pour les tranches K_s à n_b - 1 de BS :*

On a par tranche et par ligne de bus :

Un fusible pour la liaison directe

$K_s + K_d$ ensembles fusible/anti-fusible,

soit $(n_b - K_s) (K_s + K_d + 1)$ fusibles et $(n_b - K_s) (K_s + K_d)$ anti-fusibles.

* *Pour les tranches n_b à $n_b + K_s$ - 1 de BS :*

On n'a pas de liaison initiale car ce sont des tranches de réserve, il y a

donc $K_s + \sum_{j=0}^{K_s} j$ ensembles fusible/anti-fusible, soit $K_s (1 + (K_s + 1)/2)$, fusibles et autant d'anti–fusibles.

D'où on déduit la taille de tout le dispositif de connexion de deux blocs de la partie opérative :

(2.14) $T_{cl} =$

$$n_l \frac{K_s}{2}(K_s+3)\left[\left(\frac{K_s}{2}(2K_d+K_s+1)+(n_b-K_s)(K_s+K_d+1)\right)T_F+\left(\frac{K_s}{2}(2K_d+K_s-1)+(n_b-K_s)(K_s+K_d)\right)T_{AF}\right]$$

Remarque : T_F et T_{AF} dépendent de la technologie et du style de dessin utilisés. Ils doivent être évalués pour chaque cas. Les avantages de cette solution sont tout d'abord de performances électriques grâce à l'emploi de métal, et ensuite une grande fiabilité. Les principaux inconvénients sont la surface nécessaire à tous les dispositifs, et surtout les problèmes que poseraient plusieurs passages dans le processus technologique de fabrication des anti–fusibles (ce qui n'a pas été réalisé jusqu'à maintenant). Ce dernier point fait qu'il est préférable (pour le rendement de l'opération) de réserver cette technique aux cas où un seul passage pour les anti–fusibles est suffisant, c'est à dire quand on n'a à connecter qu'une tranche de réserve dans BD (et aucune dans BS). Dans ce cas on n'a plus besoin de pouvoir déconnecter les anti-fusibles après création, donc un seul fusible est nécessaire pour couper la liaison initiale, d'où :

(2.15) $T_{cl} = n_b \cdot n_l (T_{AF} + T_F)$

4.2.2 - Utilisation de transistors à grille flottante

Contrairement aux fusibles et aux anti–fusibles, les transistors à grille flottante ne peuvent pas être utilisés pour connecter directement des fils de bus (à cause des problèmes électriques que cela poserait). Leur action va donc se situer entre les portes de transfert contrôlant l'accès des éléments à connecter au bus et le bus (figure 2.19 ci–dessous).

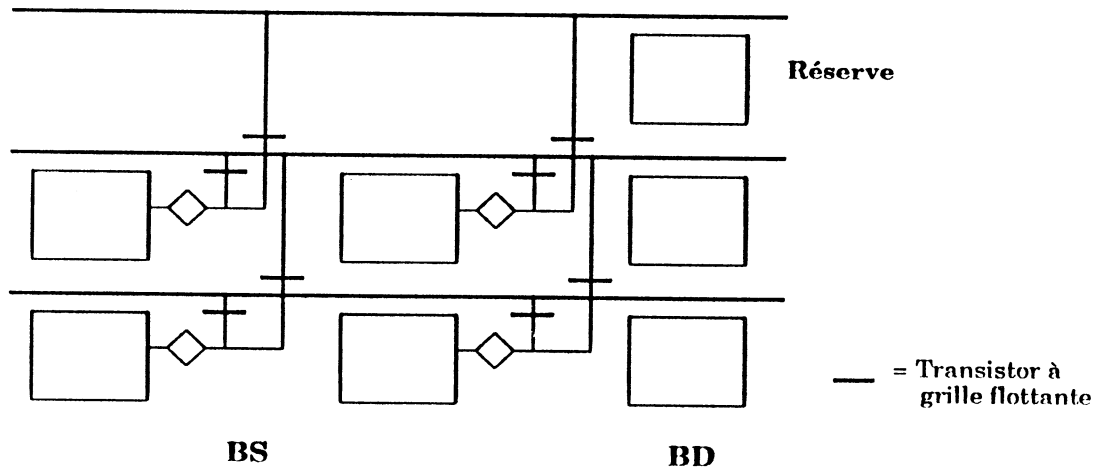


Figure 2.19 : Dispositif de connexion avec des transistors à grille flottante

Evaluation :

Compte tenu des gardes à respecter autour d'un transistor à grille flottante, on peut estimer sa taille comme étant équivalente à celle de deux transistors normaux, d'où l'évaluation du dispositif de connexion pour un élément du bloc :

- Seul BD a des tranches supplémentaires :

Dans ce cas, avec un transistor à grille flottante par lien vers une tranche de réserve, et un pour la liaison initiale, la taille du dispositif de connexion est :

(2.16) $T_{GF} = 2\, n_b \,.\, n_l\, (K + 1)$

- BS et BD ont tous les deux des tranches supplémentaires :

Le nombre de transistors à grille nécessaires est le même que celui des fusibles dans la solution précédente. On a alors :

(2.17) $$T_{GF} = n_l\, K_s\, (K_s + 3)\left[\frac{K_s}{2}\,(2\, K_d + K_s + 1) + (n_b - K_s)\,(K_s + K_d + 1)\right]$$

Les avantages de cette solution sont ceux de l'utilisation des transistors à grille flottante. Le principal inconvénient est l'incertitude actuelle quant à la possibilité de les déprogrammer individuellement.

4.2.3 - Utilisation de dispositifs logiques

Dans certaines circonstances on peut être amené à devoir limiter le nombre des dispositifs relativement longs à programmer que sont fusibles, anti-fusibles et transistors à grille flottante. Pour cela la solution est de contrôler la connexion des différents blocs avec de la logique. On remplace alors les transistors à grille flottante de la solution précédente par des portes de transfert et on y ajoute la logique nécessaire à leur contrôle (figures 2.16 et 2.17).

- Seul BD a des tranches supplémentaires :

Cette situation est représentée dans l'exemple de la figure 2.15 pour $n_b = 4$ et K = 2. La connexion d'une tranche **i** de **BS** à une tranche de **BD** est donnée par les commandes $C_{i,\,j}$ (j = 1 à K) des portes de transfert : si $C_{i,\,j} = 1$, la tranche i de BS est connectée à la tranche j de BD. On remarquera qu'une seule des $C_{i,\,j}$ peut valoir "1" à la fois.

Génération des commandes $C_{i,\,j}$

Pour chaque tranche **i** de **BD**, on dispose d'un indicateur V_i valant "1" si la tranche est utilisée et "0" sinon. Pour l'exemple de la figure 2.15 on aura :

$C_{0,0} = V_0$ (tranche en vis à vis utilisable)

$C_{0,1} = V_1 \cdot \overline{C_{0,0}}$ (on se connecte sur la tranche 1 de BD si la tranche 0 est mauvaise)

$C_{0,2} = V_2 \cdot \overline{C_{0,0}} \cdot \overline{C_{0,1}}$ (on se connecte sur la tranche 2 de BD si les deux premières sont inutilisables)

$C_{1,1} = V_1 \cdot \overline{C_{0,1}}$ (la tranche 2 de BS sera connectée à son vis à vis si celui-ci est utilisable et si la tranche 0 n'y est pas connectée)

D'où l'expression plus générale :

$$(2.18) \quad C_{i,\,i+k} = V_{i+k} \cdot \prod_{i=0,\,i\geq 0}^{k-1} C_{i,\,i+i} \cdot \prod_{j=1,\,i-j\geq 0}^{K-k} C_{i-j,\,i+k}$$

$k \in [0, K]$
$i \in [0, n_b - 1]$ — I — II

Le terme **I** indique que la tranche **i** de **BS** ne peut être connectée à la tranche **i+k** de **BD** que si elle n'est pas déjà connectée à une des tranches **i** à **i+k-1** de **BD**. Le terme **II** indique qu'une tranche **j** de **BD** est ou non déjà utilisée par une tranche d'indice inférieur de BS.

Evaluation du nombre de transistors

* Génération des commandes $C_{i,\,j}$:

Comme en CMOS les portes de transfert nécessitent que la commande soit accompagnée de sa valeur complémentée, on générera $C_{i,j}$ et son complément. Ceci est réalisé en utilisant une porte NAND et un inverseur.

Il y aura K + 1 commandes à générer pour chaque tranche de BS. De l'expression (2.18) on tire que chaque commande sera composée de K + 1 facteurs pour toutes les tranches de BS d'indice supérieur à K. Pour les tranches d'indice 0 à K - 1, il va manquer dans le terme **II** les commandes correspondant aux valeurs négatives de i - j. Le nombre de transistors pour générer les commandes pour ces tranches sera donc :

$$N1 = 2\left[\sum_{i=1,\,i\leq n_b}^{K}\left((K+1)^2 - \frac{(K-i)(K-i+1)}{2}\right)\right]$$

$$<=> (2.19) \quad N1 = 2K(K+1)^2 - \sum_{i=1,\,i\leq n_b}^{K}(K-i)(K-i+1)$$

Pour les tranches **K à n_b - 1**, on aura :

(2.20) $N_2 = 2\,(n_b - K)\,(K+1)^2$

A ces deux valeurs il faut ajouter le nombre de transistors des inverseurs de ces commandes, soit :

(2.21) $N_3 = 2\,n_b\,(K+1)$

D'où la taille totale du dispositif de génération des commandes $C_{i,\,j}$ et de leurs compléments (si $n_b \geq K$) :

$$\textbf{(2.22)} \quad T_c = 2\,n_b\,(K+1)^2\,\frac{K+2}{K+1} - \sum_{i=1,\ i\leq n_b}^{K} (K-i)\,(K-i+1)$$

* Portes de transfert pour les connexions entre BS et BD :

On aura besoin de $n_l\,(K+1)$ portes de transfert soit :

(2.23) $T_t = 2\,n_b\,n_b\,(K+1)$

- BS et BD ont tous les deux des tranches supplémentaires :

Génération des commandes $C_{i,\,j}$

Pour chaque tranche **i** de **BD** et **j** de **BS** on dispose d'un indicateur valant 1 si la tranche est utilisée :

$VS_i = 1$ si la $i^{ème}$ tranche de BS est utilisée

$VD_j = 1$ si la $j^{ème}$ tranche de BD est utilisée.

Soient K_d et K_s les nombres de tranches de réserve dans **BD** et **BS**. Dans l'exemple de la figure 2.20 ci–dessous, on a $n_b = 4$, $K_s = 2$, et $K_d = 3$.

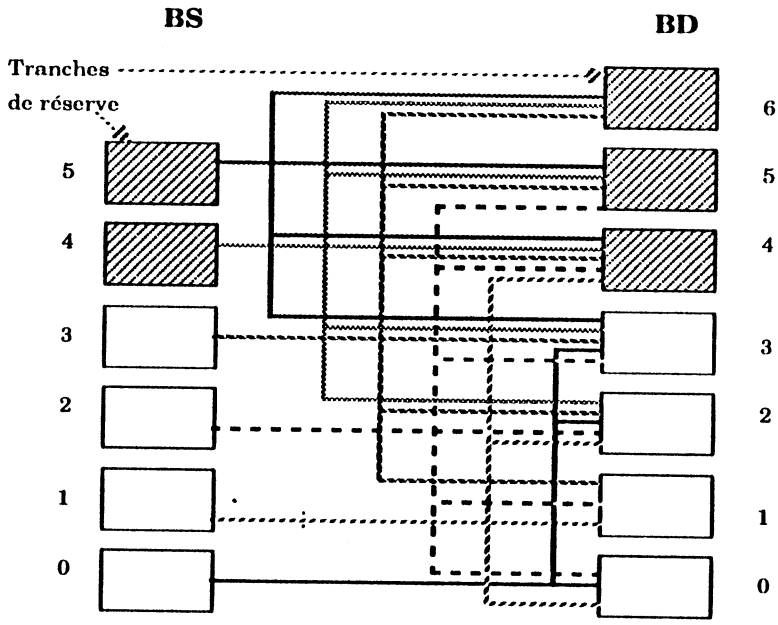


Figure 2.20 : Principe de la connexion de deux blocs ayant tous deux des tranches de bit supplémentaires

Les commandes générées sont :

$$C_{0,0} = VS_0 \cdot VD_0$$

$$C_{0,1} = VS_0 \cdot VD_1 \cdot \overline{C}_{0,0}$$

$$C_{1,0} = VS_1 \cdot VD_0 \cdot \overline{C}_{0,0}$$ (1 si tranches BS1 et BD0 utilisables, et si tranche BS0 non utilisable)

$$C_{3,4} = VS_3 \cdot VD_4 \cdot \overline{C}_{3,1} \cdot \overline{C}_{3,2} \cdot \overline{C}_{3,3} \cdot \overline{C}_{1,4} \cdot \overline{C}_{2,4}$$

indique si la tranche BS3 est déjà connectée

indique si la tranche BD4 est déjà utilisée

L'expression générale des commandes $C_{i,j}$ est la suivante :

$$(2.24) \quad C_{l,\,l+k} = VS_l \cdot VD_{l+k} \cdot \prod_{\substack{l=-K_s,\ l+l\geq 0}}^{k-1} C_{l,\,l+l} \cdot \prod_{\substack{j=1,\ l-j\geq 0}}^{K_d-k} C_{l-j,\,l+k}$$

$k \in [-K_s, +K_d]$
$i \in [0, n_b+K_s-1]$
$i+k \in [0, n_b+K_d-1]$

I II

Le terme **I** représente le fait que pour que la tranche i de BS soit connectée à la tranche i+k de BD, il faut qu'elle ne le soit pas déjà à une des tranches $i-K_s$ à i+k-1 de BD. Le terme **II** indique si la tranche i+k de BD est déjà utilisée par une tranche de BS.

Evaluation du nombre de transistors

* Génération des commandes $C_{i,j}$

De l'équation (2.24) on tire que chaque commande est composée de K_d+K_s+2 facteurs pour toutes les tranches de BS d'indice supérieur à K_d+K_s. Pour les K_d+K_s premières tranches, il va manquer :

dans le terme **I** : les commandes correspondant à des valeurs négatives de i+l.

dans le terme **II** : les commandes correspondant à des valeurs négatives de i-j.

Dans la suite on supposera que $n_b > K_s + K_d$, ce qui est réaliste compte tenu des limitations sur la surface supplémentaire. On aura alors les expressions suivantes pour la génération des $C_{i,j}$:

- **0 à K_s - 1 :**

$$(2.25) \qquad N1 = 2 \sum_{i=0}^{K_s - 1} \left[i\,(2 + i + K_d) + \frac{K_d + 1}{2} (K_d + 2\,i + 4) \right]$$

nb de facteurs constants — partie "triangulaire" des commandes

- **K_s à $K_s + K_d$ - 1 :**

$$(2.26) \qquad N2 = 2 \sum_{i=1}^{K_d} \left[(K_s + K_d + 2)(K_s + K_d + 1) - \frac{(K_d - i)(K_d - i + 1)}{2} \right]$$

nombre facteurs des expressions complètes — nombre de facteurs "manquants"

- **$K_s + K_d$ à n_b -1**

$$(2.27) \qquad N3 = 2\,(n_b - K_s - K_d)(K_s + K_d + 2)(K_s + K_d + 1)$$

nb de tranches de BS concernées — nb. de facteurs par commande — nb. de commandes par tranche

- **n_b à $n_b + K_s$ - 1 :**

$$(2.28) \qquad N4 = 2\,(K_d + K_s + 2) \sum_{i=1}^{K_s} (K_d + i)$$

nb. de facteurs par commande — nb. de commandes par tranche

Le nombre total de transistors nécessaires à la réalisation des $C_{i,j}$ est donc :

$$(2.29) \qquad T = N1 + N2 + N3 + N4$$

Remarque :

On aurait pu envisager une deuxième façon de réaliser les commandes $C_{i,j}$. Celle-ci est basée sur le fait que la plupart des facteurs nécessaires à la génération des commandes d'une tranche de BS se trouvent dans presque toutes ces commandes. Il en est de même pour les commandes allant vers une même

tranche de BD. L'idée est de ne pas réaliser à chaque commande les produits qui sont déjà utilisés par une commande précédente. Cela conduit à la réalisation suivante des $C_{i,j}$:

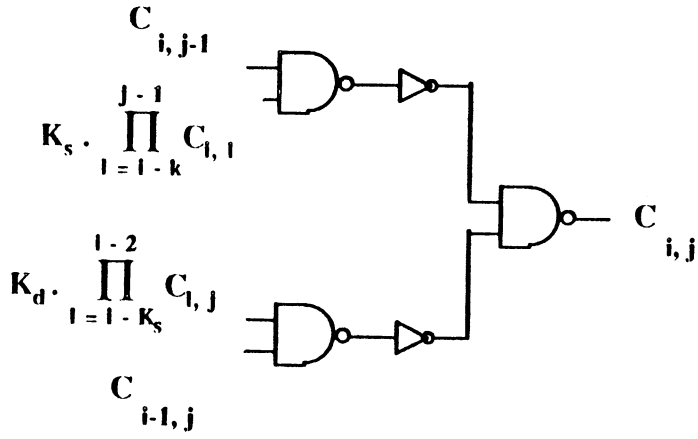


Figure 2.21 : Autre réalisation des $C_{i,j}$

Les expressions du nombre de transistors dans la seconde réalisation des commandes sont les suivantes :

- $i = 0$: $N0 = 2 (2 + 5 K_d)$ (Commandes $C_{0,j}$)
- $i = 1$: $N1 = 12 (K_d + 1)$ (Commandes $C_{1,j}$)
- $i = 2$ à K_s 1 : $N3 = 8 (K_s - 2) (2 K_d + K_s + 1)$ (Commandes $C_{2,j}$ à $C_{Ks-1,j}$)
- $i = K_s$ à $n_b + K_s - 1$: $N4 = 2 (8 n_b (K_s + K_d) - 4 Kd^2 + 9 K_d - 9)$

Comparaison des deux réalisations des commandes $C_{i,j}$:

Lorsque le nombre de termes dans les commandes augmente au delà de 8, c'est à dire quand la redondance est grande, la deuxième solution devient meilleure que la première car le nombre d'entrées à avoir pour réaliser chaque commande reste constant.

* Portes de transfert pour les connexions entre BS et BD :

Pour les K_s premières tranches de BS on a:

$$\sum_{j=0}^{K_s - 1} K_d + 1 + j = \frac{K_s}{2} (2 K_d + K_s + 1)$$

commandes soit, à raison d'un inverseur et de n_l portes de transfert par commande :

(2.30) $T_{ks} = K_d (n_l + 1) (2 K_d + K_s + 1)$

Pour les tranches K_s à n_b-1 on a $K_s + K_d + 1$ commandes par tranche, d'où :

(2.31) $T_{nb} = 2 (n_b - K_s) (n_l + 1) (K_s + K_d + 1)$

Pour les tranches n_b à n_b+K_s-1 on a :

(2.32) $T_{nb+Ks} = 3 K_s (n_l + 1) (K_s + 1)$

4.3 - Mémorisation des configurations

Nous avons vu que les divers dispositifs de contournement d'éléments en panne utilisés pour la reconfiguration utilisent des commandes indiquant si une tranche ou un élément est utilisé. Ces commandes doivent être mémorisées temporairement durant l'étape de test et de reconfiguration du circuit pour être figées lorsqu'une configuration saine a été trouvée. Il existe de nombreuses possibilités pour mémoriser les différentes configurations :

* *Registres :*

On peut utiliser un registre dont l'accès est contrôlé directement depuis l'extérieur du circuit (par exemple avec un accès série). Un tel système, où chaque bit du registre correspond à une tranche ou un bloc, ne peut être utilisé que pour une mémorisation temporaire des configurations et devra être doublé par un système de mémorisation permanente.

* *Transistors à grille flottante :*

L'utilisation de transistors à grille flottante présente l'avantage de permettre à la fois la mémorisation temporaire et la mémorisation permanente (avec toutefois des limitations sur les températures de fonctionnement). Ils facilitent également le test et la reconfiguration en étant programmables directement et sans passer par l'intermédiaire d'une quelquonque circuiterie.

* *Fusibles et anti-fusibles :*

Ils sont plus particulièrement adaptés à une mémorisation permanente de la configuration finale. Il est néanmoins possible de les utiliser durant l'étape de reconfiguration si on est sûr de ne pas avoir à revenir en arrière (c'est à dire réutiliser une tranche ou un bloc après l'avoir déconnecté).

* *Autres :*

On peut également envisager d'autres systèmes de mémorisation permanente et temporaire tels que les PROM (mémoires mortes programmables) effaçables qui au niveau accès se comportent comme des registres, mais dont la réalisation pose des problèmes dans une technologie non prévue initialement pour leur fabrication.

En définitive, on voit que le choix est très ouvert entre les différentes possibilités et qu'il dépendra essentiellement des contraintes d'un cas précis. L'implantation des dispositifs dépend de la technologie utilisée, en effet certaines technologies n'offrent pas la possibilité d'implanter des PROM effaçables. D'autre part, des limitations peuvent venir des équipements de test et de reconfiguration : précision du positionnement, taille du faisceau ...

5 - Conséquences pratiques

Les expressions qui ont été données ici permettent de faire une évaluation de la complexité et de la taille des dispositifs nécessaires aux différentes stratégies de reconfiguration possibles pour tout circuit de type microprocesseur. On peut néanmoins en tirer des règles générales permettant de guider le choix d'une solution en éliminant d'office celles qui ne seront, de toute façon, pas acceptables.

5.1 - Choix du style de partitionnement

En dehors des indications données par une évaluation précise, on peut faire les remarques suivantes quant au type de partitionnement choisi :

Si les éléments de la partie opérative ont une structure en tranche de bit, on a intérêt à utiliser cette caractéristique et à choisir un partitionnement en tranches. Par contre, dans le cas où on a des éléments disparates, de structures très différentes, une telle stratégie est inapplicable. La reconfiguration des éléments se faisant différemment, on est amené à les séparer et donc à choisir un partitionnement fonctionnel.

5.2 - Stratégie de reconfiguration dans les blocs

On peut, en théorie, envisager l'application de toutes les stratégies que nous avons présentées, ce qui amène de nombreuses évaluations dont beaucoup ne

serviront à rien si ce n'est à confirmer l'inadéquation évidente de telle ou telle solution. Le principal point va être de ne pas dépasser (au niveau global) l'augmentation de surface autorisée tout en essayant d'optimiser les possibilités de reconfiguration, ce qui amène les conclusions suivantes pour les différentes méthodes de reconfiguration :

* ***Réplication massive d'un élément :***

Cette méthode entraîne au minimum le doublement de la taille de l'élément considéré, ce qui va au delà des valeurs "efficaces" de l'augmentation de surface. Si l'élément ne peut être reconfiguré que de cette façon (cas d'un bloc de logique combinatoire par exemple), on prendra garde aux points suivants :

- La taille de l'élément doit être suffisamment petite par rapport au reste du circuit pour que l'influence de sa réplication soit négligeable.

- La taille des dispositifs de réorientation éventuels doit également ne pas être trop pénalisante.

Ceci amène à ne conseiller l'utilisation de cette méthode que pour des éléments de petite taille tels que des portions de logique aléatoire ou des petits périphériques.

* ***Ajouts de modules de réserve :***

On a encore ici à veiller à la taille des dispositifs de réorientation par rapport à celle de l'ensemble. Si on prend l'exemple de la reconfiguration d'un ensemble de modules de seize bits par ajout de deux modules de réserve, les courbes de la figure 2.22 indiquent la proportion de matériel rajouté en fonction du nombre de modules et pour plusieurs tailles d'une tranche de module (les valeurs de ces courbes et des suivantes sont données dans l'annexe 1).

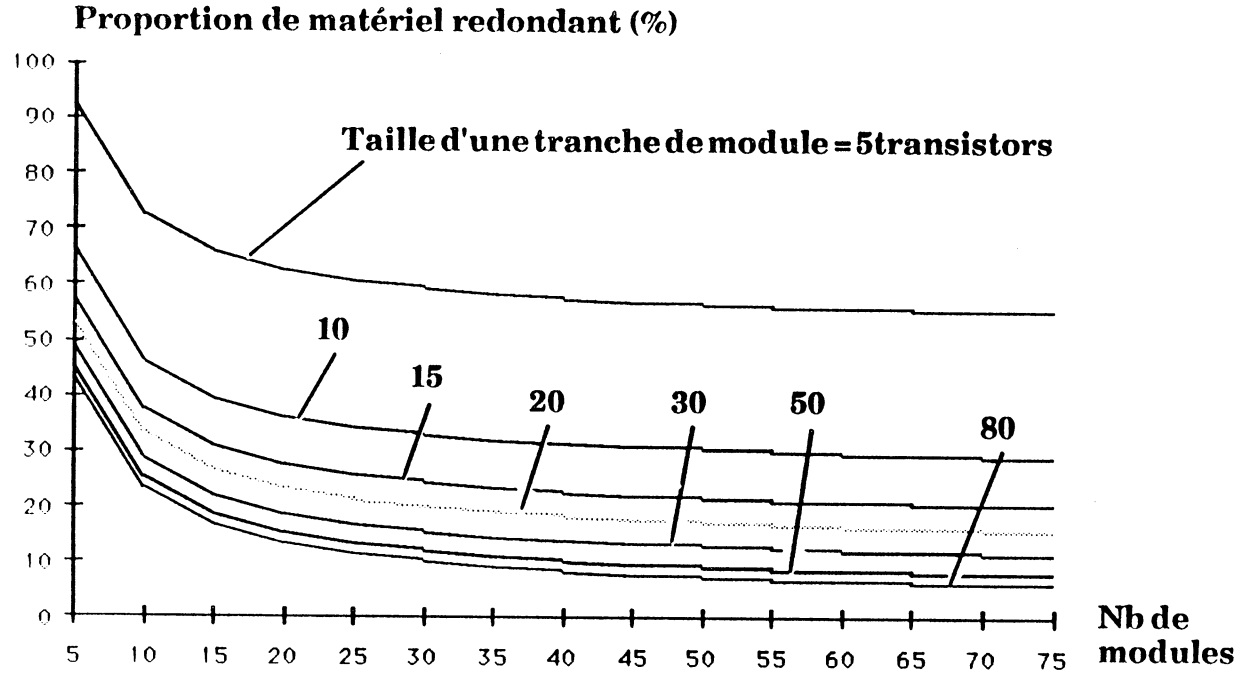


Figure 2.22 : Proportion de matériel rajouté pour une redondance modulaire

On voit par exemple que dans ce cas, ce type de reconfiguration peut être envisagé si on a plus d'une vingtaine de modules dont chaque tranche comporte plus de vingt transistors (si on veut limiter l'augmentation de surface à 30%).

* ***Ajout de tranches de bit supplémentaires*** **:**

Cette stratégie de reconfiguration est la plus couramment utilisable. En effet, la plupart du temps les éléments d'une partie opérative ont une structure en tranche de bit. La recherche des plus grandes possibilités de reconfiguration implique d'essayer de mettre le plus de tranches possible en réserve sans toutefois dépasser la limite de l'augmentation de surface.

Les courbes des figures 2.23 et 2.24 indiquent la proportion de matériel rajouté en fonction du nombre de la taille d'une tranche et pour plusieurs valeurs du nombre de tranches de réserve (opérateurs sur seize bits).

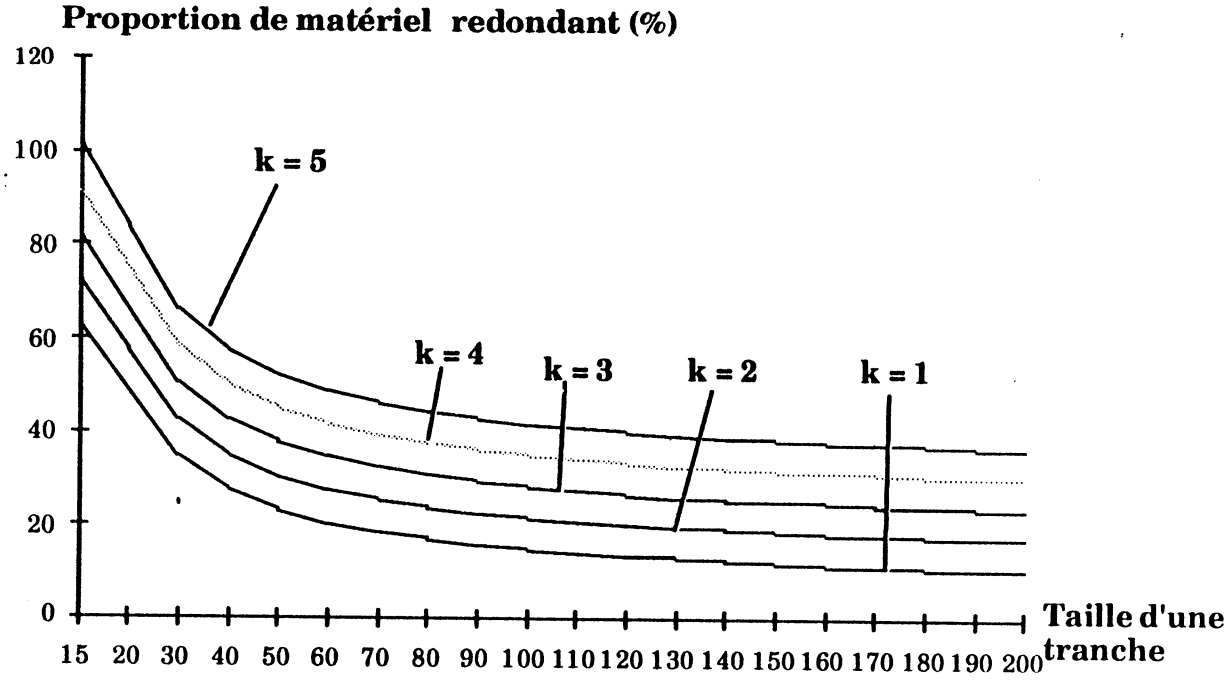


Figure 2.23 : Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec une seule retenue de propagée

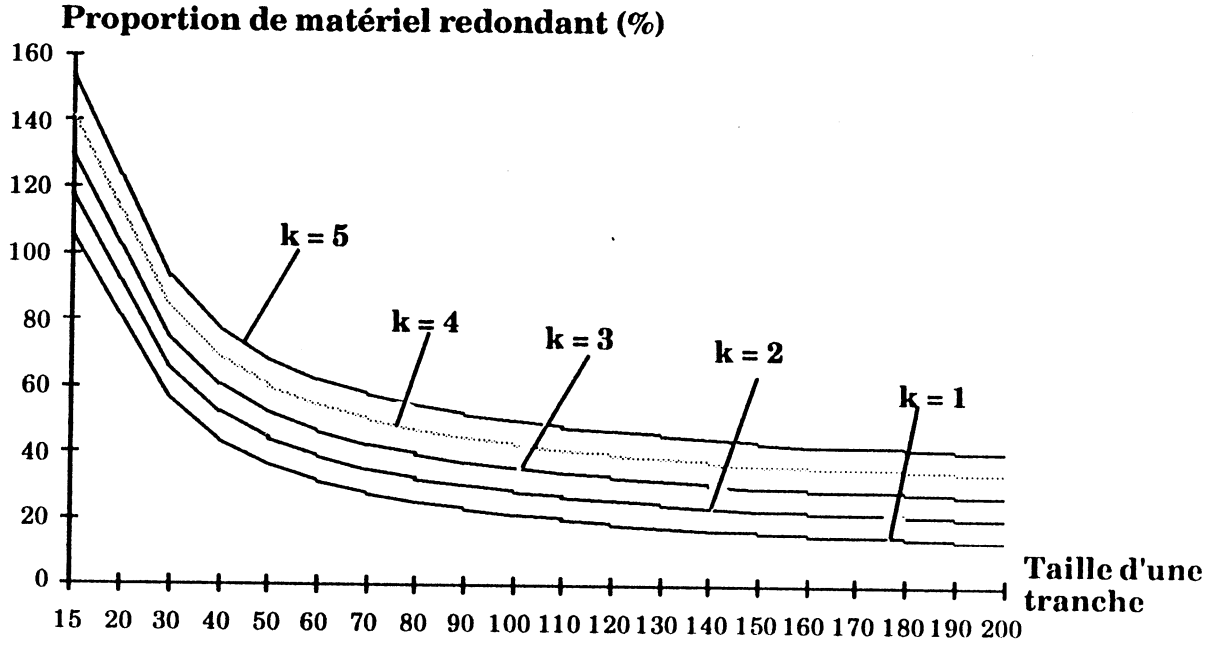


Figure 2.24 : Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec deux retenues de propagées

Ces courbes nous montrent que dans le premier cas (une retenue) une augmentation de surface inférieure à 30% requiert une tranche d'une taille de 60 transistors au moins et pas plus de deux tranches de réserve. Dans le cas ou deux retenues sont propagées, la taille minimale d'une tranche passe à environ 90 transistors pour pas plus de deux tranches de réserve.

* ***Connexion des blocs :***

Il faut tout d'abord choisir le dispositif de connexion qui sera utilisé. Parmi ceux que nous avons vus, on peut pratiquement éliminer les dispositifs programmés par logique à cause de la place énorme qu'ils prennent. De tels dispositifs ne seront utilisés qu'en dernier recours, par exemple si le nombre de dispositifs de mémorisation permanente utilisables est très limité. Les fusibles/anti—fusibles ont pour eux leur bonne fiabilité après programmation, mais en contrepartie, ils ne sont pas reprogrammables et ils occupent une place non négligeable. Les transistors à grille flottante sont le dispositif le plus économe en place, mais leur fiabilité dans le temps et à haute température est encore à prouver.

Lors de l'implantation de ces dispositifs on doit chercher à en limiter le nombre, en regroupant au maximum les éléments non reconfigurables par exemple.

Chapitre 3

Reconfiguration de la Partie Contrôle d'un Microprocesseur

1 - Contraintes architecturales

La reconfiguration d'une partie contrôle de microprocesseur est, du fait de sa structure, un problème totalement différent de la reconfiguration de la partie opérative. Il existe une grande variété d'architectures pour les parties contrôle de microprocesseurs [BEL81], [OBR82]. On peut distinguer les catégories suivantes : organigramme câblé, microprogrammation, structures à base de PLAs (mono-PLA, multi-PLAs avec ou sans hiérarchisation).

La reconfiguration de la plupart de ces architectures se heurte principalement à deux difficultés :

- leur manque de régularité globale
- les difficultés d'accès aux éléments internes.

* ***Manque de régularité :***

Des solutions telles que "l'organigramme câblé" [OBR82], où aucune structure régulière n'est employée et où, de plus, chaque élément a une signification propre (ce qui le rend non interchangeable), ne peuvent de toute évidence pas être rendue reconfigurables. Dans les autres architectures le problème est légèrement différent. En effet, on se trouve ici avec un ensemble d'éléments réguliers, mais c'est l'ensemble qui manque de régularité. On aura alors à déterminer autant de stratégies de reconfiguration qu'il y a d'éléments différents dans la structure choisie, ce qui entraînera alors autant de problèmes de test et de localisation des défauts.

* ***Difficultés d'accès aux éléments internes :***

Dans les architectures classiques de partie contrôle, il existe un grand nombre d'éléments internes tels que registres de micro-adresse, pile câblée pour les adresses de retour de sous-microprogramme, décodeurs internes, multiplexeurs, ... En dehors des problèmes de reconfiguration posés par une telle diversité de blocs, il existe une grande difficulté d'accès à ces éléments internes. Rien n'est généralement prévu pour pouvoir accéder directement aux entrées et aux sorties de ces éléments, ce qui rend une localisation précise des défauts très difficile. La prise en compte de toutes ces difficultés permet de définir ce que doit être une architecture de partie contrôle reconfigurable.

2 - Conception de Parties contrôle reconfigurables

Les deux points principaux sont la régularité et la simplicité. La régularité va se traduire par l'emploi de structures de type PLA, et la simplicité par la limitation du nombre d'éléments différents aux PLAs et aux registres. Quel que soit le type de structure utilisé pour la partie contrôle, il n'est pas possible de remplacer un élément défectueux sans entièrement le répliquer. Une telle approche est inacceptable à cause de l'augmentation de surface qu'elle entraînerait. La solution consistera, donc à rajouter de la redondance à l'intérieur des éléments de la P.C.

La conception d'une telle partie contrôle se fait à l'aide d'un outil de synthèse automatique de contrôleurs tel que le système ASYL (Aide à la SYnthèse Logique) développé au Laboratoire Circuits et Systèmes, et comporte les étapes suivantes [SAU87] :

- Spécifications initiales : Il s'agit du graphe d'états de l'automate de contrôle (organigramme de contrôle) où à chaque noeud sont associées les opérations effectuées par la P.O durant l'état correspondant. Des prédicats sont attachés aux arcs pour contrôler le séquencement. Un assistant de validation permet de vérifier sur ce graphe l'abscence de blocages dans un noeud et l'abscence de parallélisme entre les arcs.

- Choix d'un style d'architecture et utilisation d'un assistant de synchronisation qui à partir de l'organigramme initial et des caractéristiques temporelles de l'architecture retenue va générer un graphe de contrôle temporisé. En ce qui nous concerne (contrôleurs reconfigurables) les architectures retenues sont à base de PLAs.

- Codage des états de l'automate de contrôle en utilisant un assistant fonctionnant à base de règles [SAU87].

- Génération des équations des sorties et des variables internes.

- Partitionnement du PLA virtuel unique ainsi obtenu en plusieurs petits PLAs. Ceci est fait en respectant les principes de hiérarchisation de la PC et en prenant en compte les contraintes topologiques issues de la partie opérative pour la localisation des commandes.

- Minimisation logique de chaque PLA à l'aide du minimiseur booléen à base de règles d'ASYL. Ces deux dernières étapes étant répétées jusqu'à l'obtention d'une solution satisfaisante.

Une fois le contenu logique du contrôleur obtenu, on passe à sa réalisation avec des outils de génération de PLAs et on implante les différents dispositifs rendus nécessaires pour la reconfiguration.

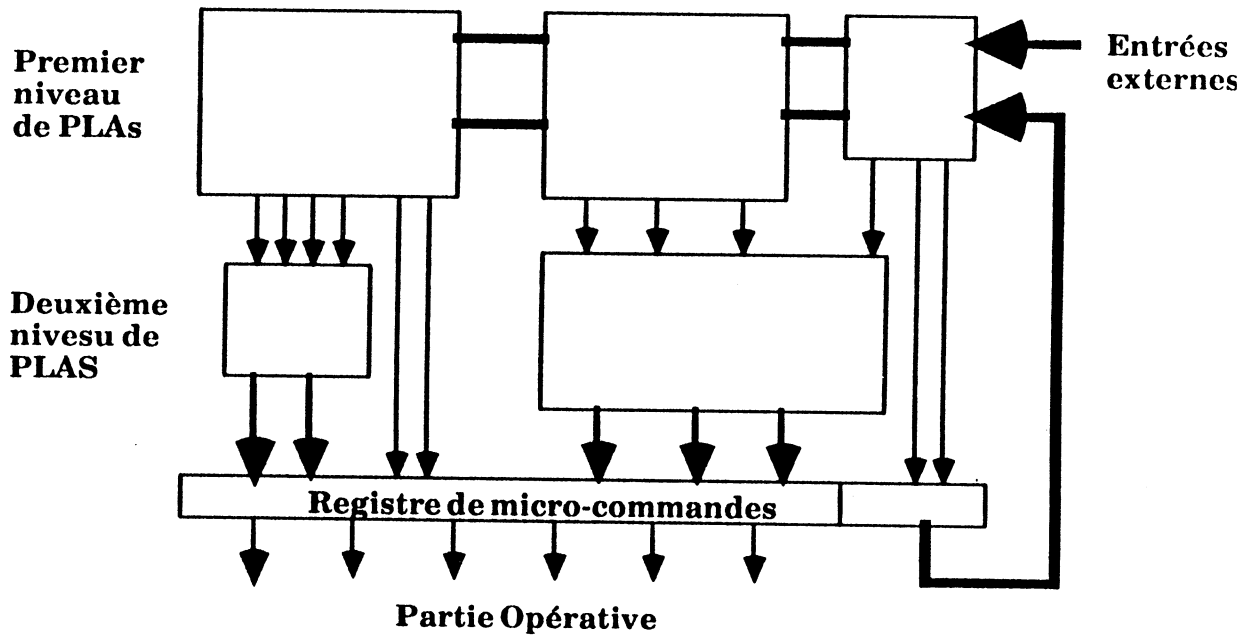


Figure 3.1 : Structure d'une partie contrôle multi-PLAs

3 - Reconfiguration des éléments de la partie contrôle

3.1 - Registres et interconnexions

* ***Registres :***

Une partie contrôle de microprocesseur comprend un certain nombre de registres de taille variable : registre de micro-commandes, registre de numéro d'état suivant, ainsi que divers registres internes permettant de contrôler les entrées et d'observer les sorties de tous les PLAs en vue de permettre une détection et une localisation des défauts. Ces registres peuvent être simples ou maître/esclave. Les registres implantés à des fins de test et de localisation de défauts sont accessibles directement de l'extérieur par des plots de test. Comme

ceux du registre d'état de la partie opérative, les bits de ces registres ont chacun une signification particulière. Tout ceci détermine les contraintes de base à prendre en compte pour leur reconfiguration

On peut remarquer que ces registres sont, pour la plupart, de petite taille. La probabilité d'y avoir un défaut sera supposée négligeable, et ils n'ont donc de ce fait pas besoin d'être reconfigurables. Lorsqu'un registre atteint une taille telle qu'il puisse être atteint par des défauts (s'il est très long, comme un registre de micro-commandes au dessus de la partie opérative par exemple), deux solutions s'offrent pour le rendre reconfigurable : la réplication et l'ajout de cellules de bit en réserve.

- La **réplication** permet de préserver la spécificité de chaque cellule de bit. Elle suppose une réorientation des commandes et des entrées/sorties vers le (ou les) registre(s) de réserve. La taille d'une telle solution est évaluée de la même manière que pour la réplication d'éléments de la partie opérative (voir chapitre 2, §3.1).

- L'ajout de **cellules de bit** de réserve permet de limiter le degré de redondance au strict nécessaire, mais, comme les cellules ne sont pas interchangeables, il faut prévoir un dispositif permettant de rendre le remplacement d'une cellule défectueuse transparent pour le reste du contrôleur (figure 3.2 ci-dessous).

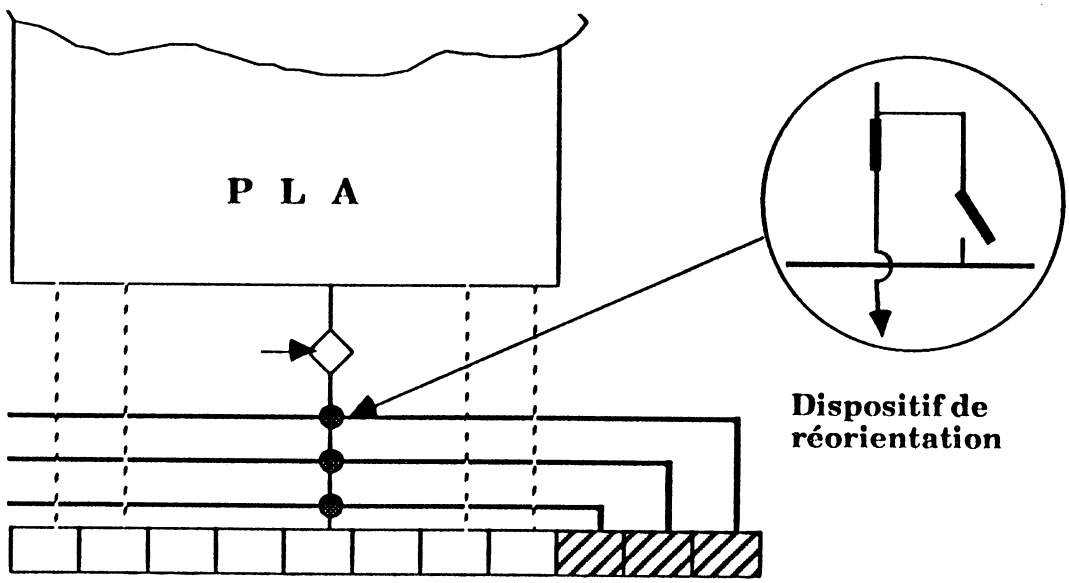


Figure 3.2:Reconfiguration d'un registre par ajout de cellules supplémentaires

Pour un registre simple, un seul dispositif de réorientation est nécessaire pour chaque cellule, mais pour un registre maître/esclave, il en faut deux. Le dispositif de réorientation peut être réalisé de trois manières comme le système de connexion de deux blocs de la partie opérative : avec des fusibles (et des anti-fusibles), avec des transistors à grille flottante, avec des portes de transfert (dispositif logique). Comme indiqué dans la figure 3.2, initialement le chemin "normal" doit être programmé et les commutateurs sont là pour permettre sa déprogrammation et sa réorientation vers une des cellules de réserve. L'utilisation de portes de transfert classiques nécessite la génération de commandes fixes dès la fabrication du circuit (pour le lien initial), lesquelles commandes ne peuvent être fournies qu'à l'aide des deux autres dispositifs (fusibles (et anti-fusibles), transistors à grille flottante). Ceci implique que la surface requise pour implanter cette solution sera de toute façon plus importante que celle requise pour les deux autres. Si on désire employer des fusibles/anti-fusibles, il faut mettre un fusible pour le commutateur de la liaison initiale, et un anti-fusible pour chaque lien possible avec une cellule de réserve. De même avec les transistors à grille flottante, il en faut un qui soit passant et un qui ne le soit pas. Vu les tailles respectives de ces dispositifs, il apparaît que la solution à base de transistors à grille flottante est la plus économe en surface supplémentaire. Soit T_c la taille d'une cellule de bit du registre, n_{cr} le nombre de cellules de réserve, et n_{br} le nombre de bits du registre, on aura la taille d'un dispositif de reconfiguration de registre :

$$(3.1) \qquad T_r = n_{cr} \cdot T_c + 2\, n_{br} (n_{cr} + 1)$$

* ***Interconnexions*** :

Les connexions reliant les différents éléments d'un contrôleur sont très nombreuses et de dimensions variables. Pour des raisons évidentes de place et de performances, on cherche toujours à les réduire au maximum. Lorsqu'il est impossible d'éviter l'emploi d'une grande "nappe" de fils, on peut se poser le problème de leur sensibilité aux défauts de fabrication. Une première solution consiste à ne pas dessiner ces lignes de métal aux dimensions minimales de la technologie. On peut également envisager d'ajouter des lignes de réserve. Le problème posé est alors de même nature que pour les registres (en effet, chaque ligne a une signification particulière), et le seul moyen pour y répondre est d'ajouter des lignes de réserve munies à chaque extrémité de dispositifs leur permettant d'être connectées aux mêmes points que les lignes qu'elles remplacent. Contrairement au cas précédent, les caractéristiques électriques que

ces lignes doivent conserver font qu'il est préférable d'utiliser des fusibles/anti-fusibles plutôt que des transistors à grille flottante dont la résistance est trop grande et qui dégradent les niveaux logiques.

3.2 - Réseaux logiques programmables (PLAs)

Les PLAs sont les plus grands éléments de la partie contrôle d'un microprocesseur. C'est donc pour eux que le problème de la sensibilité aux défauts de fabrication va se poser avec le plus d'acuité. Le problème de la conception de PLAs reconfigurables a été étudié en détail dans [DAN86] et nous en rappellerons ici les grandes lignes.

Un PLA peut être divisé en trois ensembles : les entrées, les monômes et les sorties(figure 3.3). Dans un PLA reconfigurable, ces ensembles sont divisés en sous-ensembles où des lignes de réserve sont implantées pour remplacer celles qui viendraient à être défectueuses. Il est important de noter que cette stratégie de reconfiguration ne change pas l'interface du PLA avec ses voisins (l'utilisation de réserve est transparente pour eux).

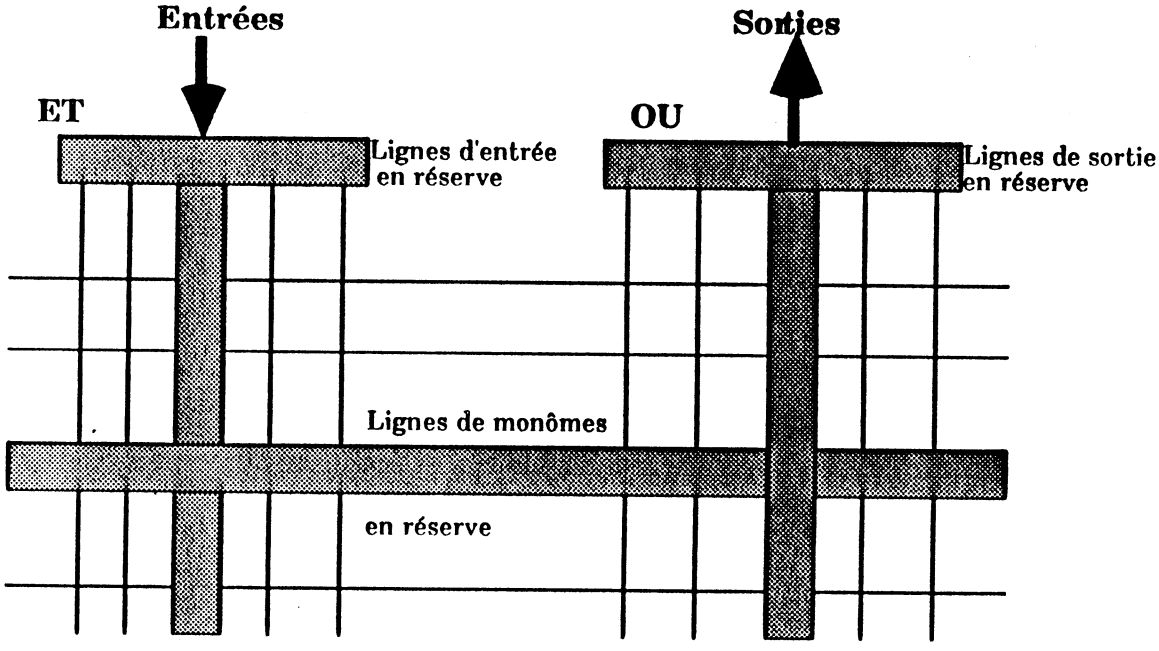


Figure 3.3 : Structure de PLA reconfigurable

3.2.1 - Reconfiguration des entrées

Les lignes d'entrées sont partitionnées en sous-ensembles, et des lignes de réserve sont implantées dans chaque sous-ensemble de manière à pouvoir en remplacer n'importe quelle ligne. Ceci implique l'implantation dans les lignes de réserve des transistors correspondant à toutes les connexions possibles des lignes

du sous–ensemble aux lignes de monômes. Durant l'étape de reconfiguration, les lignes de réserve peuvent être programmées pour remplacer n'importe quelle ligne du sous–ensemble en ne connectant que les lignes de monômes reliées à l'entrée en panne. Il existe également un commutateur permettant d'isoler la ligne défectueuse et de connecter son lien externe à la ligne de réserve (figures 3.4 et 3.5 ci-après).

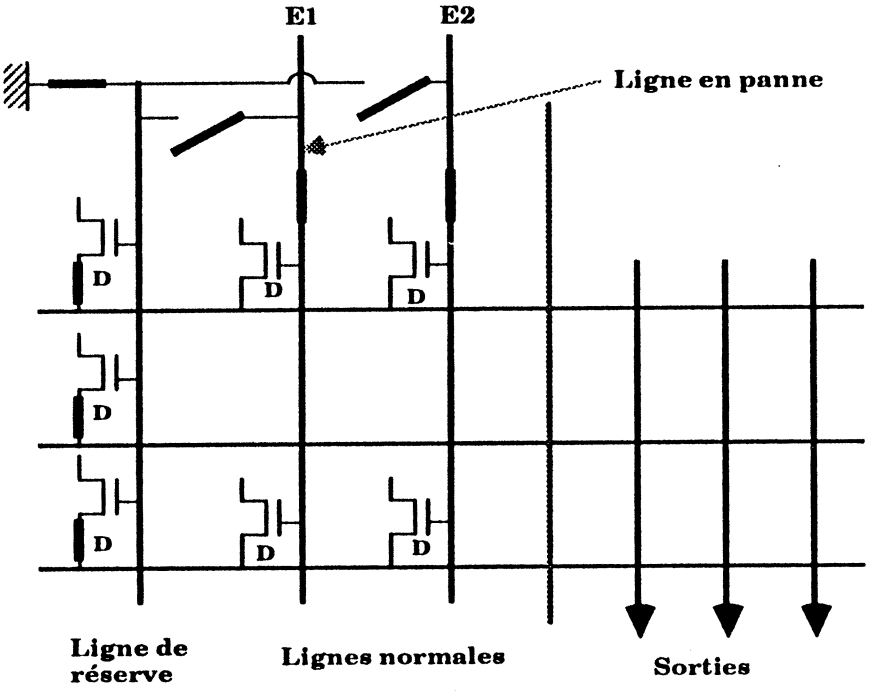


Figure 3.4 : Avant reconfiguration des entrées

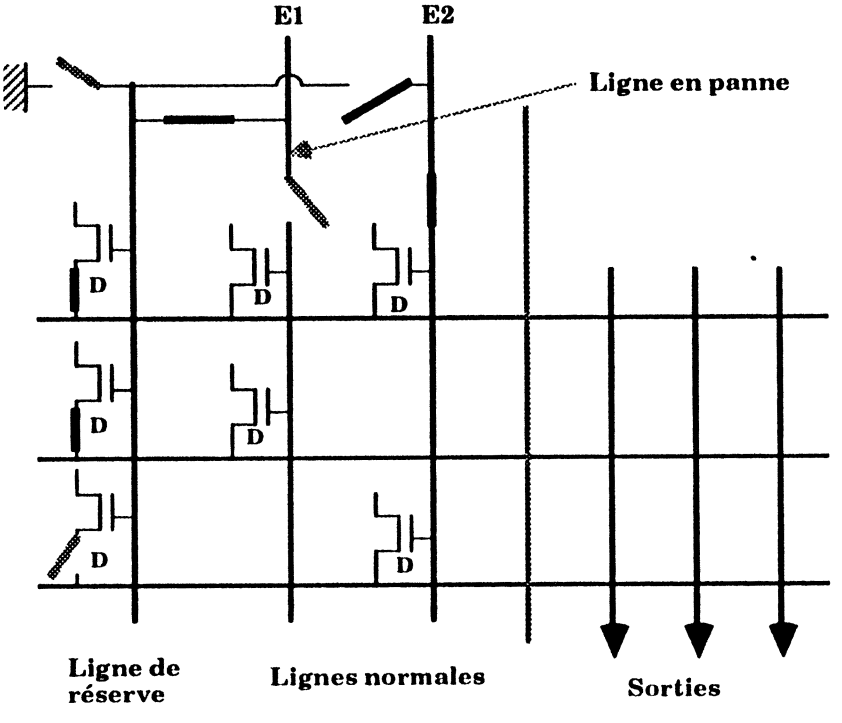


Figure 3.5 : Après reconfiguration des entrées

3.2.2 - Reconfiguration des monômes

Chaque ligne de monôme (qui est le plus souvent faite d'aluminium dans la matrice ET et de silicium polycristallin dans la matrice OU) comporte des drains dans la matrice ET et des grilles dans la matrice OU. Il y a également un étage d'amplification entre les deux matrices.

Comme pour les entrées, les monômes sont partitionnés en sous-ensembles où des lignes de réserve programmables sont implantées. Lors de l'étape de reconfiguration, les lignes de réserve peuvent être activées en ouvrant les transistors ne correspondant pas à des transistors des monômes qu'elles remplacent.

On peut remarquer que les lignes de réserve sont neutres pendant le test car toutes les entrées y sont présentes avec leur valeur complémentée, ce qui les amène au niveau logique zéro, les empêchant par là même de décharger les lignes de sortie (figures 3.6 et 3.7 ci-dessous).

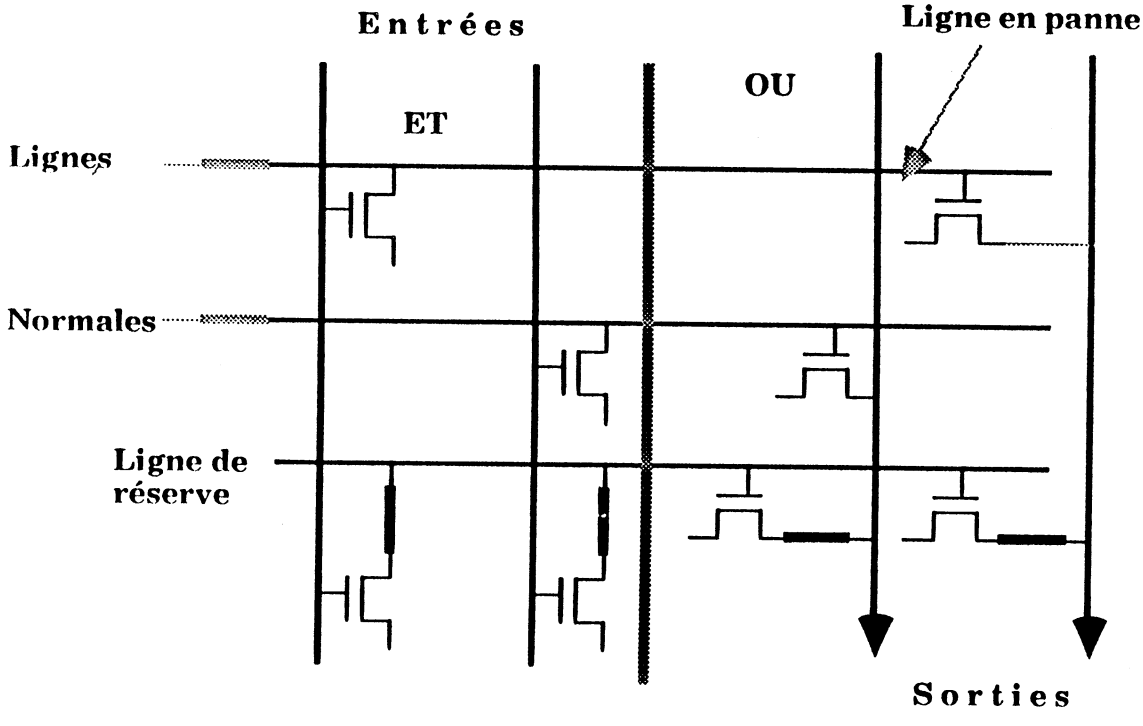


Figure 3.6 : Avant reconfiguration des monômes

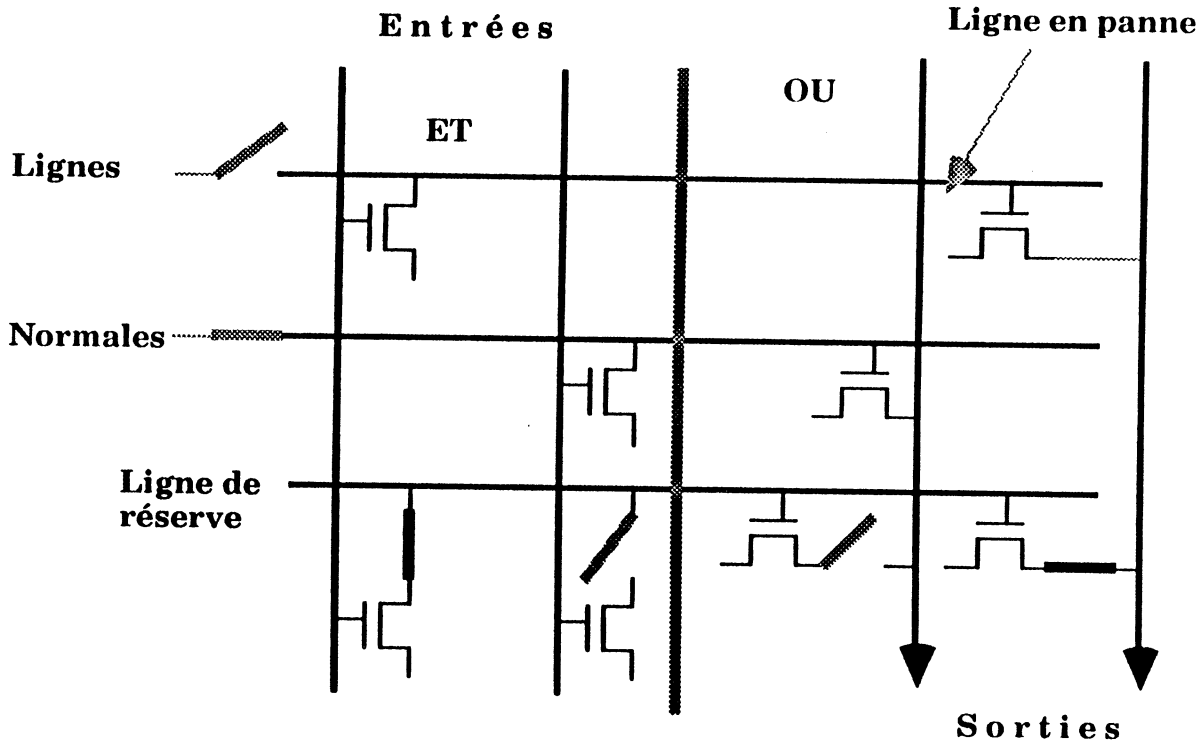


Figure 3.7 : Après reconfiguration des monômes

3.2.3 - Reconfiguration des sorties

On se trouve ici dans le même cas que pour les entrées. Les sorties sont partitionnées en sous-ensembles où des lignes non programmées couvrant toutes les connexions des lignes du sous-ensemble sont implantées. On a aussi les commutateurs permettant de connecter une ligne de réserve au lien externe de la ligne défectueuse qu'elle remplace (figures 3.8 et 3.9 ci-dessous).

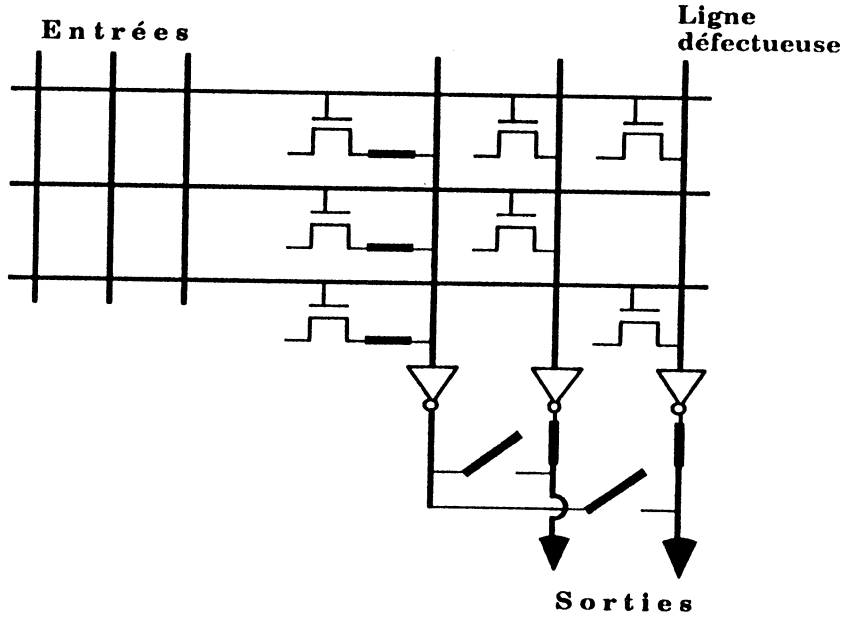


Figure 3.8 : Avant reconfiguration des sorties

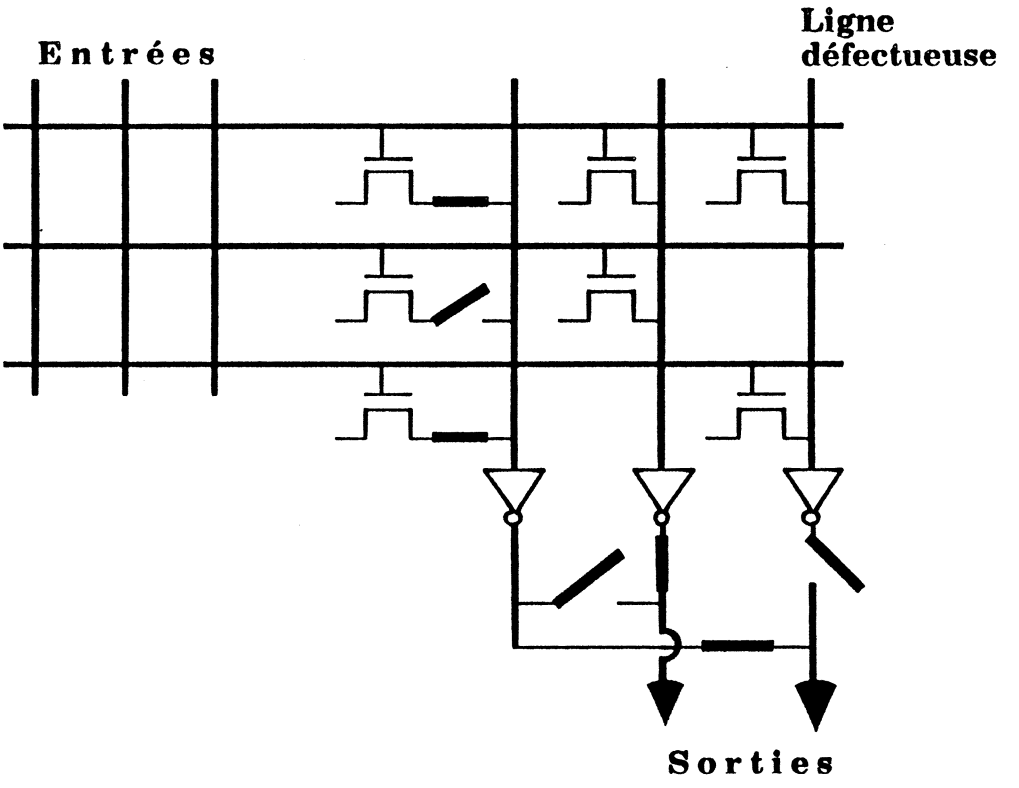


Figure 3.9 : Après reconfiguration des sorties

L'observation des défauts de fabrication montre [SCH78], [CAS76], [MAN84] que l'on a plus de défauts affectant les transistors que de défauts affectant les interconnexions. Puisque les transistors défectueux dans la matrice ET sont réparés par la reconfiguration des lignes de monômes, et que les défauts atteignant des transistors dans la matrice OU sont couverts par la reconfiguration des lignes de sorties, on peut supprimer la reconfiguration des entrées.

3.2.4 - Réalisation des commutateurs

Les différents commutateurs doivent pouvoir être programmés de façon permanente. Cela exclut l'emploi de transistors classiques. On a le choix entre les fusibles/anti-fusibles et les transistors à grille flottante. Ces derniers ont pour eux leur faible encombrement qui permet de les utiliser à l'intérieur des matrices de PLA sans nuire à la compacité de l'ensemble. Les fusibles ne sont, quant à eux, utilisables qu'aux endroits où ils n'entraînent pas une augmentation du pas entre lignes (au niveau des amplificateurs entre les deux matrices par exemple).

Le nombre de lignes de réserve implantées est déterminé en fonction de la taille du PLA, de l'augmentation de surface autorisée pour le circuit, ainsi que du

nombre de défauts que l'on souhaite pouvoir corriger. Pour que cette correction soit possible, il faut rendre la localisation des défauts possible. Pour cela, on peut, par exemple, implanter des registres à décalage aux entrées et aux sorties des PLAs, de façon à pouvoir les contrôler directement. Il est également envisageable de rajouter des entrées permettant d'avoir des configurations particulières pour le test.

Evaluation de la taille d'un PLA

La grande diversité des méthodes d'implantation de PLAs fait qu'il n'est pas possible de donner une expression précise pour en évaluer la taille. Nous allons donner ici une forme simplifiée d'évaluation où il faudra fixer certains paramètres en fonction des particularités de chaque implantation.

Soient **E** le nombre d'entrées du PLA, **S** le nombre de ses sorties, et **M** le nombre de ses monômes, on aura alors pour un PLA sans reconfiguration :

$$(3.2) \qquad T_{PLA} = (2E + S + T_{KM})M + T_{KES}(2E + S)$$

où T_{KM} est la taille des dispositifs présents une fois dans chaque monôme (précharge,...), et T_{KES} est la taille des dispositifs présents aux extrémités des lignes d'entrées et de sorties (précharge, inverseurs d'entrée, amplificateurs de sortie, ...). Ces deux paramètres dépendent du type de PLA réalisé, ainsi que de la taille du PLA. Pour ce qui est des dispositifs de reconfiguration, on peut évaluer la taille d'une ligne programmable (entrée, sortie ou monôme) à **quatre** fois celle d'une ligne normale quand la programmation est réalisée avec des transistors à grille flottante.

4 - Conséquences pratiques

Comme pour la partie opérative, l'éventail des solutions de reconfiguration à évaluer est important. Il donc intéressant de pouvoir faire un premier tri à l'avance. Ce tri consiste à déterminer quels sont les éléments de la PC pour lesquels on implantera de la redondance. Il est évident que les premiers à recevoir de la réserve sont las PLAs. En ce qui concerne les registres et les interconnexions, on peut dire deux choses :

* Les registres sont relativement peu nombreux et petits par rapport au reste de la PC. Leur reconfiguration présente donc assez peu d'intérêt.

* Les interconnexions peuvent être un facteur très pénalisant, en effet, si le partitionnement et le plan de masse font apparaître un grand nombre d'interconnexions, celles-ci vont être sensibles aux défauts. Les stratégies de reconfiguration pour les interconnexions ne sont pas faciles à mettre en oeuvre. L'objectif devra donc être de limiter le plus possible les interconnexions dans la PC, par un partitionnement et une implantation adéquates.

Chapitre 4

Application au Microprocesseur HYETI

1 - Présentation générale

1.1 - Caractéristiques principales

A l'initiative d'industriels français notamment la S.N.C.F., un microprocesseur à jeu d'instructions réduit "HSURF" a été conçu pour les systèmes de contrôle à haute sûreté de fonctionnement [GEN84], [JAY84], [JAY86]. Ce circuit a été retenu comme démonstrateur pour les techniques de reconfiguration de microprocesseurs développées dans le cadre du projet ESPRIT 824 - WSI. L'architecture de la machine HSURF a été étudiée pour les besoins des applications de contrôle en environnement haute sécurité. La plupart des dispositifs rendus nécessaires par cette classe d'applications ont été conservés dans sa version reconfigurable, HYETI, qui a les caractéristiques suivantes :

* Les instructions sont celles utilisées le plus couramment dans le domaine ; elles rendent l'écriture des logiciels d'application extrêmement simple.

* Des instructions spécialisées permettent d'implanter un excellent test en ligne à travers le programme d'application.

* Les techniques de compaction de l'information permettant de "signer" l'état du microprocesseur et le rendent d'un usage aisé pour les architectures redondantes.

* Sa structure interne autorise la mise en œuvre d'un test hors ligne rapide et efficace.

* ***Caractéristiques matérielles :***

* Architecture 16 Bits :

  - Format des données : 16 Bits.
  - Espace d'adressage : 16 pages de 64 K-Mots.

* Dispositif de compaction programmable de l'information, autorisant la signature de toutes les informations circulant sur les bus internes du microprocesseur.

* Opérateurs internes statiques entièrement testables.

* Indicateur d'alarme.

* Tension d'alimentation unique de 5 volts.

* ***Caractéristiques logicielles*** **:**

* Jeu d'instruction réduit adapté au contrôle des automatismes logiques (son contenu est détaillé en annexe 2).

* 14 Modes d'adressages

* Instructions de lecture/modification/écriture.

* Instructions de modification/comparaison pouvant porter sur des mots, des parties de mots ou des bits isolés.

* Modes d'adressage spécifiques pour le traitement de matrices bidimensionnelles.

* Instructions spécifiques de recherche dans une matrice et de comparaison de deux matrices de même caractéristiques.

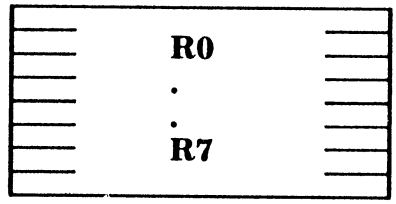
* Instructions spécifiques pour le test en ligne (autotest en cours de fonctionnement à travers le programme d'application).

* Instructions spécifiques pour le test hors-ligne (commandabilité et observabilité permettant le test exhaustif des blocs matériels clefs du circuit).

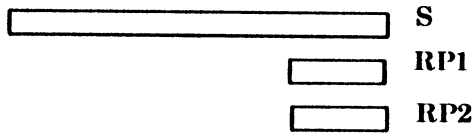
* ***ARCHITECTURE***

Organisation des registres internes :

Registres utilisateur :



Registres Adresse



Registres système

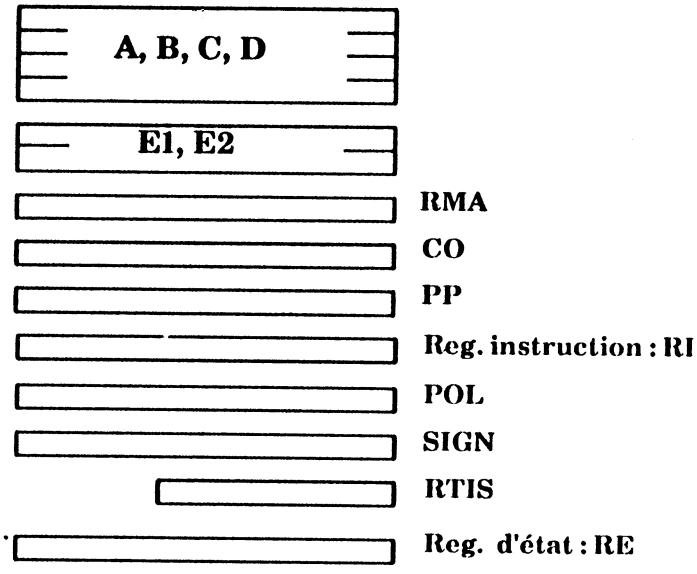


Figure 4.1 : Organisation des registres

Organisation de la mémoire :

HYETI peut adresser directement 1 Méga-Mots de 16 Bits organisés en 16 pages de 64 K-Mots. L'organisation de la mémoire est la suivante :

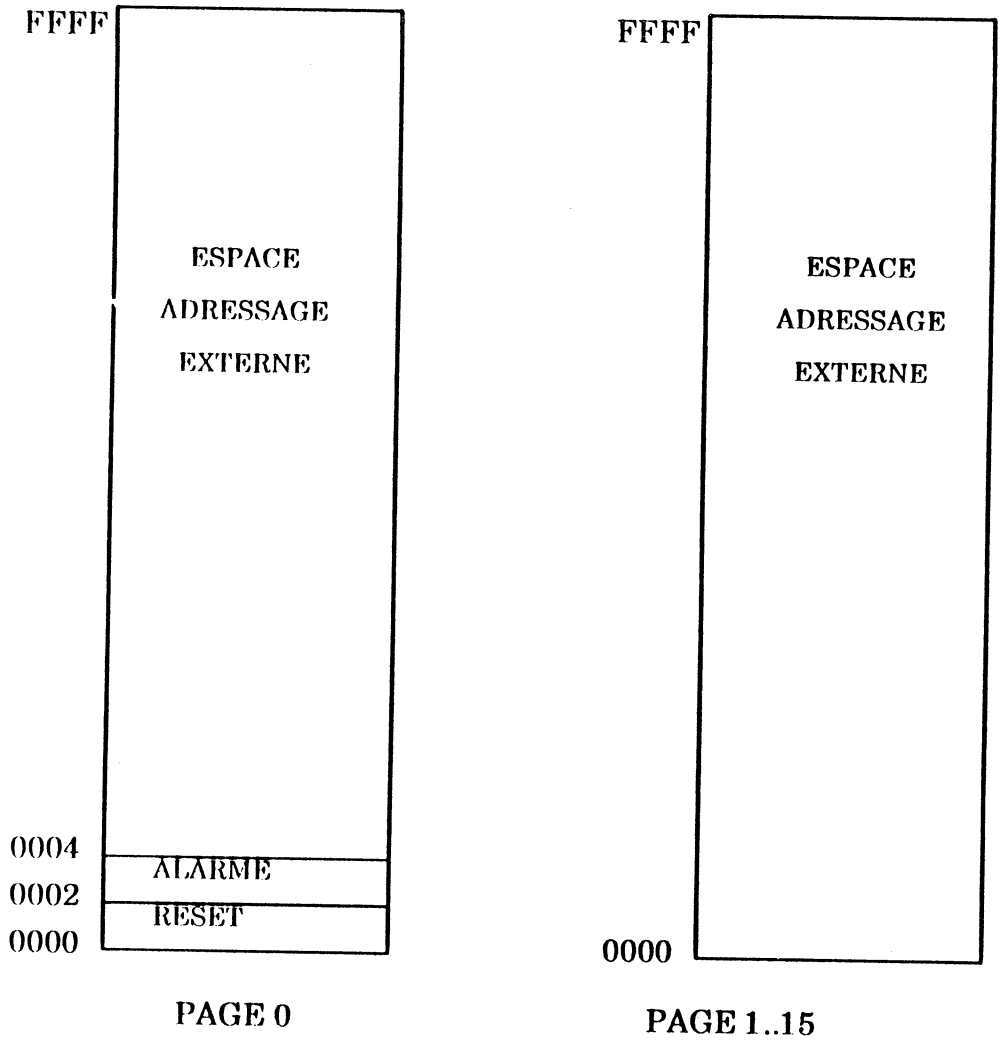


Figure 4.2 : Espace d'adressage mémoire

Modes d'adressage :

HYETI permet de réaliser des opérations à 2 opérandes s'effectuant suivant le format suivant :

DEST <--- DEST **op** OPERANDE

où DEST est un registre interne et où OPERANDE peut être :

- Un autre registre (REGISTRE).
- Une constante immédiate (IMMEDIAT).
- Un mot mémoire (DIRECT ou INDIRECT/REGISTRE).
- Un élément de matrice (MATRICE).

Il existe plusieurs modes d'adressage pour chacun des types d'opérandes précédents, soit :

- 4 modes d'adressage normaux : IMMEDIAT, REGISTRE, INDIRECT par REGISTRE et DIRECT.
- 8 modes d'adressage matrice dépendant du mode de passages des paramètres.
- les modes d'adressage INHERENT et RELATIF (Instructions de branchement).

Jeu d'instructions :

Six classes d'instructions peuvent être considérées :

- Instructions **arithmétiques, logiques** et de **transfert :** ADD, ADDC, SUB, SUBC, INC, DEC, NOT, AND, OR, NAND, NOR, EOR, LSL, LSR, ASL, ASR, COMP, LOAD, STORE, EXCH.

- Instructions de **rupture de séquence :** BRA, Branchement conditionnel pour chacun des bits du registre d'état avec test si "1" ou si "0", RSR, JSR, JMP.

- Instructions avec opérande **inhérent :** CLC, SEC, NOP, PUSH, PULL.

- Instructions avec **masquage :** Toutes les instructions logiques, de comparaison et de transfert peuvent être réalisées avec masquage (seuls les bits correspondant à des bits à "1" dans le registre masque sont affectés par l'opération).

- Instruction de **comparaison de matrices** et de **recherche dans une matrice :** COMPARMAT.

COMPARMAT permet de comparer de façon automatique deux matrices bidimensionnelles de même caractéristiques :

COMPARMAT Adr1 Adr2 1 1

Adr1	0	1	2
0	T	O	N
1	B	O	D
2	Z	V	K
3	Z	A	N

Adr2	0	1	2
0	C	I	A
1	B	T	D
2	Z	V	K
3	Z	U	N

Eléments testés

Premier élément différent : MAT(3, 1)

L'exécution de cette instruction a pour effet de positionner un bit du registre d'état (E) à 0 si les deux matrices sont identiques et à 1 si une différence à été détectée ; dans ce cas, deux registres contiennent les coordonnées des éléments différents.

COMPARMAT réalise également la recherche du contenu d'un registre dans une matrice bidimensionnelle.

- Instructions de **Test :** Le microprocesseur HYETI offre une observabilité et une accessibilité à l'ensemble de ses ressources internes. Un dispositif programmable de compaction de l'information intégré directement au sein du circuit permet de contrôler avec précision l'exécution des programmes d'application. Afin de faciliter la mise en œuvre de ces dispositifs, des instructions spécifiques de test ont été développées :

  - LRS, SRS permettant d'accéder en lecture et en écriture tous les points de mémorisation internes au microprocesseur.
  - RAZ, AJS, AJST permettant d'influer sur le contenu du registre de signature afin de mettre en œuvre les stratégies de test en ligne.

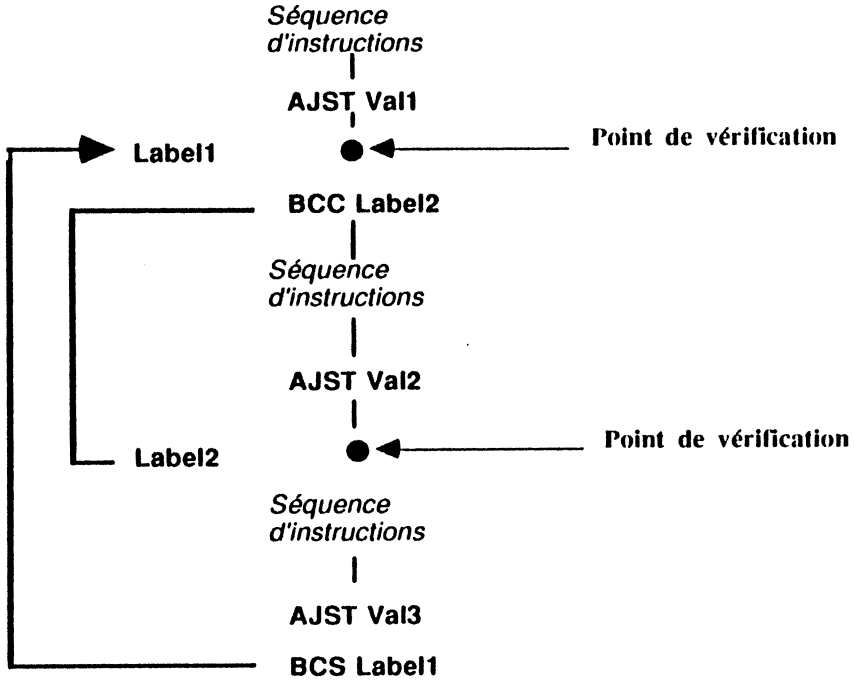
HYETI intègre les concepts de :

* **Test en ligne** à travers le programme d'application avec ajustement de signature : le programme utilisateur est transformé automatiquement de telle sorte que la suite des codes opérations compactés par signature est invariante pour tout chemin entre deux points du programme d'application.

* **Test hors-ligne (maintenance)** : Un automate de test intégré au circuit permet de vérifier l'ensemble des ressources internes au microprocesseur par accès direct à tous les points critiques. Des séquences de test minimales avec opérandes pré–stockés permet-tent de réaliser un test exhaustif des opérateurs de la partie opérative.

* **Mise en œuvre aisée dans les systèmes à architecture redondante** (duplex, triplex, TMR) : comparaison aisée entre les différents microprocesseurs du contenu des registres de compaction.

* **La mise en œuvre du test en ligne de l'application avec HYETI :** le test en ligne des applications réalisées sur HYETI consiste en la compaction de la suite des codes opération dont est constitué le programme. Avec HYETI, la mise en œuvre d'une telle stratégie est réalisée grâce à l'insertion de l'instruction d'ajustement (AJS ou AJST) à la fin de chaque séquence du programme d'application. Cette instruction produit à partir des séquences de code opération parcourues à l'aide du dispositif de compactage, un nombre invariant (signature) quel que soit le chemin parcouru entre les 2 points de vérification.



L'insertion des instructions AJS (ou AJST) dans le programme d'application peut être réalisée automatiquement à l'aide d'un logiciel spécifique. Il s'agit d'un contrôle de haut niveau, donc très efficace directement intégré dans le silicium. Cette facilité décharge le concepteur du problème critique de l'insertion de facilités de test en ligne.

1.2 - Structure de la partie opérative

La partie opérative du microprocesseur HYETI est organisée autour de deux bus de seize bits Alpha et Beta. Elle comporte les éléments suivants indiqués dans la figure 4.3 ci–dessous.

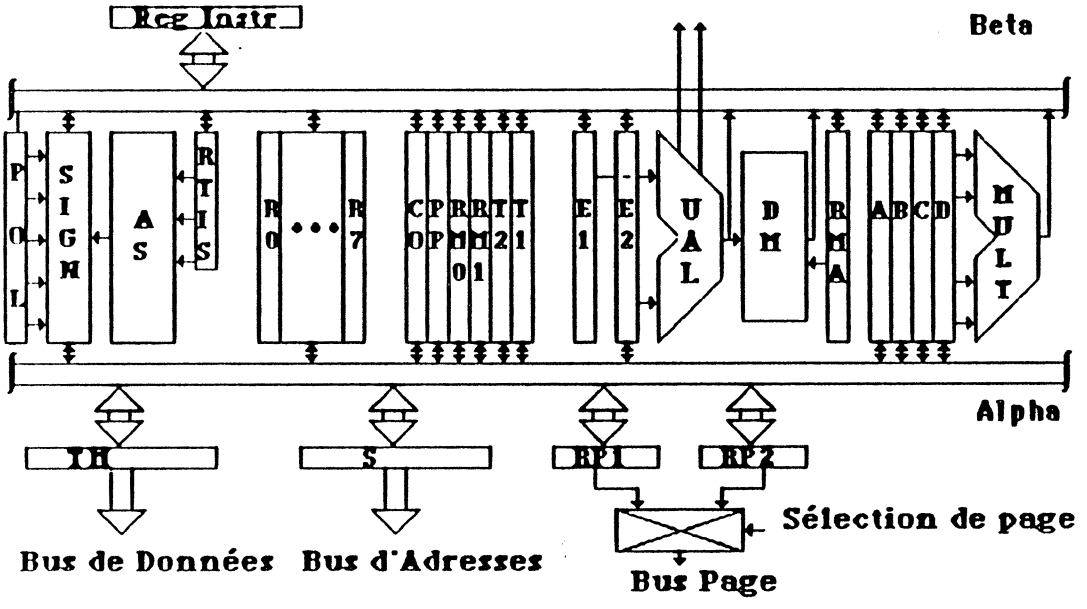


Figure 4.3 : Architecture de la partie opérative de HYETI

1.2.1 - Registres et Logique de bus

Description

Les registres sont divisés en deux groupes :

- Seize registres accessibles avec les **instructions "normales"** :
  * *Huit registres généraux* R0 à R7. Ces registres ont accès en lecture/écriture sur le bus Alpha, et en lecture seulement sur le bus Beta. On entend par accès en *lecture* que le contenu du registre est lu sur le bus, et par accès en *écriture* que la valeur du bus est écrite dans le registre.
  * *Huit registres spécialisés* dont une utilisation pratique nécessite certaines facilités d'accès.

    CO : Compteur ordinal. Il a accès en lecture/écriture sur les deux bus.

    PP : Pointeur de pile. Il a accès en lecture/écriture sur Alpha et en lecture sur Beta.

    RM0 et RM1 : Registres d'adresse de descripteurs de matrice. Ils ont accès en lecture/écriture sur Alpha et en lecture sur Beta.

RP1 : Registre page 1. Il contient le numéro de la page où est situé le programme. Il a accès en lecture/écriture sur Alpha et en lecture sur Beta.

RE : Registre d'état. Il a accès en lecture/écriture sur Alpha et en lecture sur Beta. Ce registre est également chargé avec les indicateurs d'état générés par les opérations de l'U.A.L et les transferts sur le bus Alpha.

POL, RTIS Ces deux registres font partie du dispositif de signature programmable utilisé pour compacter les informations sur les bus. Leur rôle est expliqué au paragraphe 1.2.2. Il ont tous deux accès en lecture/écriture sur Alpha et en lecture sur Beta.

- Des registres internes qui ne sont accessibles qu'au travers des **instructions de test** :

T1, T2 : Tampons internes. Ils ont accès en lecture/écriture sur les deux bus.

E1, E2 : Ce sont les tampons d'entrée de l'U.A.L. E1 est le tampon pour les données venant sur Alpha. Il a accès en lecture/écriture sur Alpha uniquement, et il comporte une sortie directe vers l'U.A.L et le dispositif de masquage. E2 est le tampon pour les données venant sur Beta. Il a accès en lecture/écriture sur Alpha et en écriture uniquement su Beta. Il est également muni d'une sortie directe vers l'U.A.L.

RP2 : Registre page 2. Ce deuxième registre page est utilisé pour l'adressage d'éléments de matrices car celles-ci ne sont pas dans la même page que le programme. Il a accès en lecture/écriture sur Alpha et en écriture sur Beta.

S : Tampon d'adresses. Il a accès en lecture/écriture sur les deux bus. Il a également une sortie directe vers le bus adresses externe.

TM : Tampon de données. Il a accès en lecture/écriture sur les deux bus ainsi qu'une sortie directe vers le bus données externe.

RMA : Registre masque (voir dispositif de masquage §1.2.3). Il a accès en lecture/écriture sur Alpha et en écriture sur Beta.

A, B, C, D, CONT, RESM : Ce sont les registres utilisés par le multiplieur. Leur rôle est expliqué au paragraphe 1.2.4. A, C et D ont accès en lecture/écriture sur Alpha et en écriture sur Beta. B a accès en lecture/écriture sur les deux bus. CONT et RESM ont accès en lecture/écriture sur Alpha uniquement.

SIGN: Registre de signature. On peut accéder normalement à ce registre à l'aide des instructions de test. Il a accès en lecture/écriture sur Alpha et en écriture sur Beta.

FLAGS : Sous ce nom sont regroupées plusieurs bascules d'indicateurs d'état de manière à pouvoir être accédées directement pour faciliter le test. Ces indicateurs sont : N, Z, L, CIN, RESET, ALARME.

Réalisation

Les registres simples ont la structure suivante (figure 4.4) :

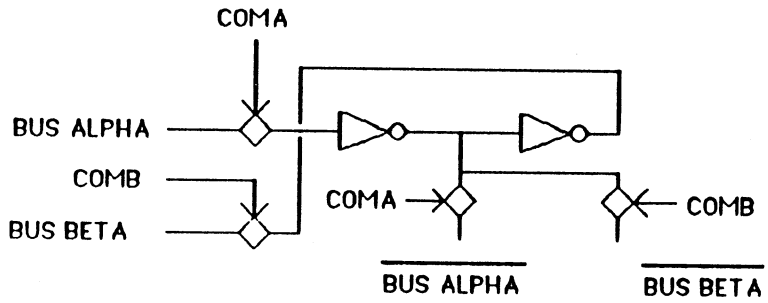


Figure 4.4 : Cellule de registre simple

Les portes de transfert permettant l'accès aux bus sont réalisées à l'aide d'un transistor de type N. La taille de ce type de registre est de **4** transistors pour le point de mémorisation, et de **2** transistors pour chaque connexion à un bus.

Dans les registres à décalage, on introduit une structure maître/esclave où le rebouclage des deux inverseurs du point de mémorisation est coupé par une porte de transfert (figure 4.5).

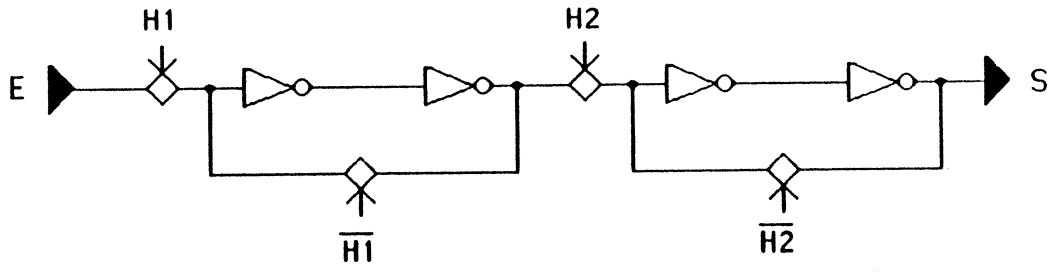


Figure 4.5 : Cellule de registre à décalage mono-directionnel

Les commandes requises pour contrôler ce type de registre sont toujours COMA et COMB avec en plus les commandes de décalage. La taille d'une cellule de ce type est de **16** transistors pour le point de mémorisation maître/esclave, auxquels on ajoute les transistors N d'accès aux bus (**2** par accès).

Dans certains cas (registre E1 par exemple), on a besoin d'avoir un registre à décalage bi-directionnel. Ceci est réalisé comme suit :

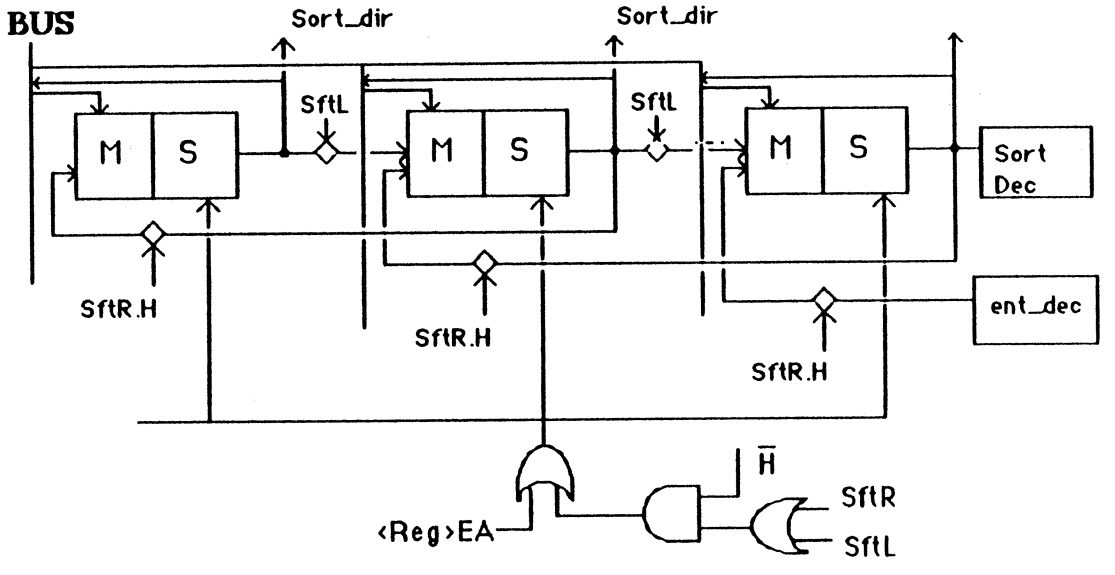


Figure 4.6 : Registre à décalage bi-directionnel

Par rapport au registre à décalage simple, on rajoute une porte de transfert (**2** transistors) par cellule pour le décalage à gauche.

La structure de bus utilisée est celle de bus à précharge avec amplificateur différentiel comme montré dans la figure 4.7.

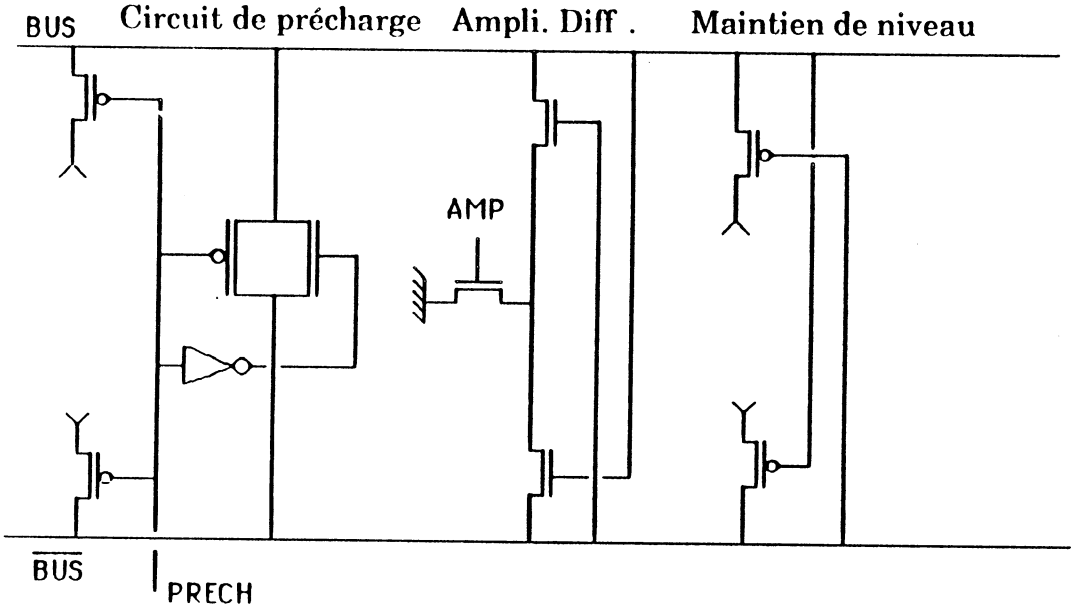


Figure 4.7 : Schéma de la logique de bus

Le fonctionnement du bus pour un échange entre registres se fait selon les étapes suivantes :

- Précharge du bus et du bus complémenté
- Connexion sur le bus du registre lu.
- Lorsque la valeur du bus a commencé à varier, la commande AMP de l'amplificateur différentiel est rendue active de façon à ce que la différence entre BUS et BUSB soit amplifiée sans toutefois risquer de rétro–basculement du registre lu.
- Connexion sur le bus du registre en écriture.

Ce mode de fonctionnement requiert une synchronisation des commandes d'accès aux bus car les temps de connexion ne sont pas les mêmes pour le registre lu que pour le registre écrit (figure 4.8). Ceci implique que les commandes d'accès ne sont pas les mêmes pour une lecture et pour une écriture :

<Reg>EA : Registre en entrée sur Alpha (écriture)
<Reg>SB : Registre en sortie sur Beta (lecture)

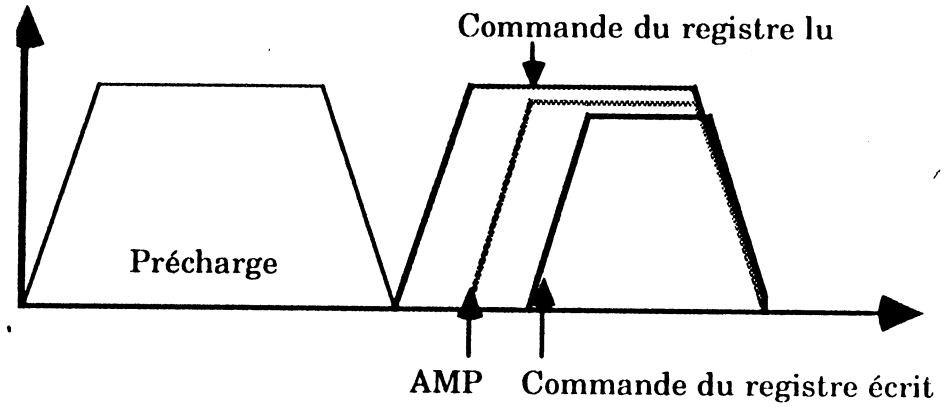


Figure 4.8 : Synchronisation des transferts sur les bus

Les dimensions des transistors composant la logique de bus étant très différentes de celles des autre parties du circuit, nous ferons l'évaluation en donnant une taille équivalente en transistors minimaux. Les cinq transistors de l'amplificateur différentiel et du maintien de niveau ont W/L = 80/1,4 ce qui est équivalent au niveau implantation à 20 transistors minimaux, soit un total de **109** transistors. Les deux transistors P du circuit de précharge ont W/L=40/1,4 ce qui donne une taille équivalente de 10 transistors. Avec la porte de transfert et son inverseur de commandes on arrive au total de **24** transistors. Pour les deux bus la logique de bus aura donc une taille de **496** transistors.

Cas particulier : Registre d'état RE

Il mémorise les valeurs de six indicateurs d'état générés par l'U.A.L et le dispositif de masquage : C, Z, N, V, H et L. Il contient également la valeur de l'indicateur E généré par COMPARMAT ainsi que trois indicateurs composés à partir des quatre premiers :

G = C + Z
T = N OUEX V
W = Z + (N OUEX V)

Le registre d'état est chargé avec les valeurs des indicateurs d'état générés par l 'U.A.L selon les ordres contenus dans les commandes du contrôleur. Ceci n'est toutefois pas le cas quand RE est utilisé comme un registre normal ou comme registre destination. Les indicateurs d'état sont copiés dans RE suivant le contenu du champ ARE des micro-commandes et les commandes Set_RE et Reset_RE. Ces dernières sont utilisées pour forcer les bits de RE à "1" ou à "0". La

retenue entrante de l'U.A.L est positionnée en chargeant la valeur du bit C de RE dans la bascule CIN à l'aide de la commande SCIN. Les différentes configurations de chargement de RE possibles sont les suivantes :

C + Set/Reset
E + Set/Reset
N, Z
C, N, Z, (W, T, G, V)
C, N, Z W, T, G, V, (L, H)
N, Z, L, H

Ceci requiert l'utilisation des commandes ci-dessous :

* **ARE** : Ce champs indique quels bits de RE sont modifiés. Sa structure est décrite ci–dessous.

Chgt H L	Chgt W T G V	Chgt N Z	Chgt E	Chgt C
4	3	2	1	0
CHL	CWTGV	CNZ	CE	CC

L'entrée des bits de RE dans le contrôleur (pour réaliser les instructions de branchement) est multiplexée de manière à n'avoir qu'une seule entrée au lieu de dix (figure 4.9).

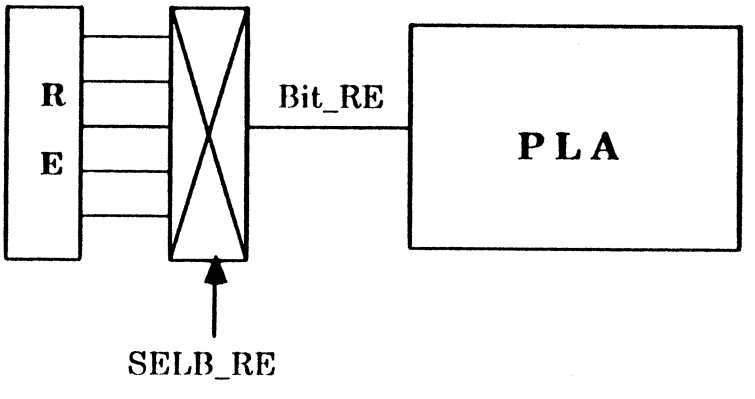


Figure 4.9 : Multiplexage des bits du registre d'état vers le contrôleur

Le schéma complet du dispositif du registre d'état est donné dans la figure 4.10.

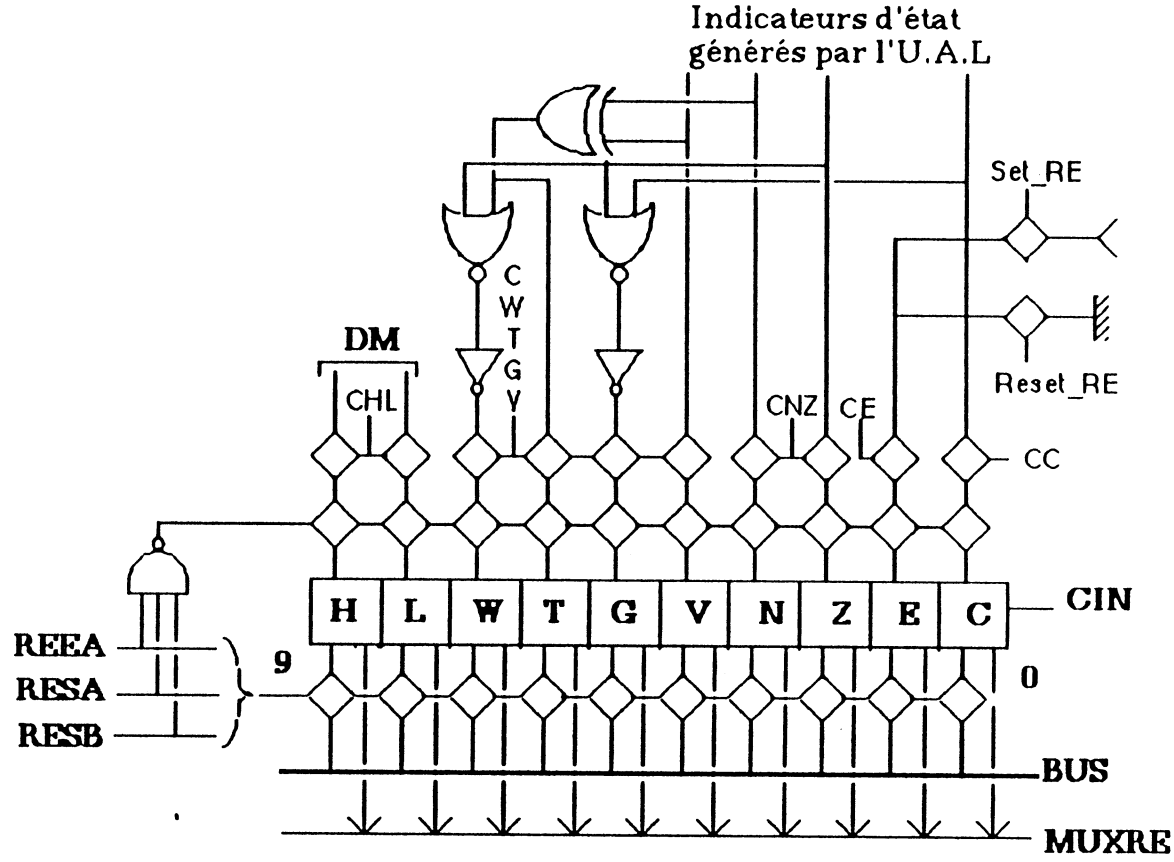


Figure 4.10 : Schéma logique du registre d'état

Evaluation : La taille des portes de génération de W, T, G est négligeable devant la taille du reste du circuit. La taille d'une cellule de RE est celle d'une cellule de registre simple à laquelle on ajoute deux portes de transfert, soit 12 transistors.

1.2.2 - Dispositif de signature programmable

La signature est le résultat d'une de compaction de données. Elle peut être faite des données arrivant en série ou en parallèle [HAS82]. Dans notre cas la structure en tranches de la partie opérative et la grande quantité de données ont conduit au choix d'une signature parallèle. Ce résultat est obtenu en effectuant la division du polynôme représentant les données par un polynôme diviseur appelé polynôme caractéristique [WAN84]. Le dispositif matériel réalisant cette opération est un registre à décalage rebouclé (en anglais : L.F.S.R, Linear Feedback Shift Register).

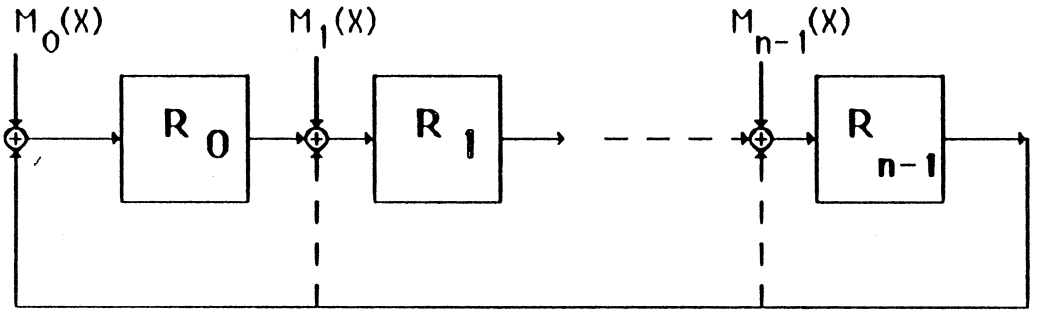
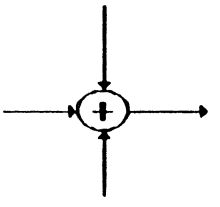


Figure 4.11 : Schéma général d'un registre de signature



représente un OU Exclusif à 2 ou 3 entrées

* Les $M_0, \ldots, M_{n-1}$ représentent un mot des entrées X.
* $R_0, \ldots, R_{n-1}$ sont les éléments de mémorisation du registre. Ce sont des cellules maître/esclave (à cause du rebouclage).

$P(X) = X^n + h_{n-1}.X^{n-1} + h_{n-2}.X^{n-2} + \ldots . + h_1.X + 1$

où : $h_i = 1$ si la sortie R_{n-1} est connectée à l'entrée de la porte OUEX de R_i $(0 < i < n)$

$h_i = 0$ autrement

représente le polynôme caractéristique du registre.

Réalisation

L'originalité du dispositif de signature intégré dans HYETI est sa grande souplesse d'utilisation. Celle-ci est obtenue en offrant à l'utilisateur la possibilité de choisir son polynôme de signature ainsi que le type d'informations saisies pour la signature.

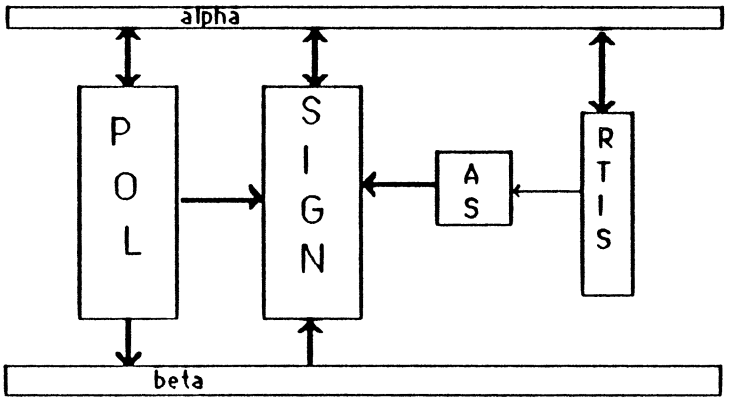


Figure 4.12 : Schéma général du dispositif de signature de HYETI

- **POL** est un registre contenant le polynôme de signature représenté avec les conventions suivantes :
  * *ième bit de POL = 1* signifie qu'il n'y a pas de rebouclage au ième bit de SIGN.
  * *ième bit de POL = 0* signifie qu'il y a un rebouclage au ième bit de SIGN.

- **RTIS** est le registre contenant le type d'informations à prendre en compte pour réaliser la signature :

7	6	5	4	3	2	1	0
Adresses programme	Autres adresses	Données lues	Données écrites	Codes instruction	Tout sur ALPHA	Tout sur BETA	Rien

Les différentes possibilités peuvent être conbinées selon l'ordre de priorité suivant : bit 0, bit 1 et bit 2, adresses, données. Si un conflit apparaît entre Alpha et Beta pour l'autorisation de signature, la priorité est donnée à Alpha.

- **SIGN** est le registre de signature. Son schéma logique est donné dans la figure 4.13.

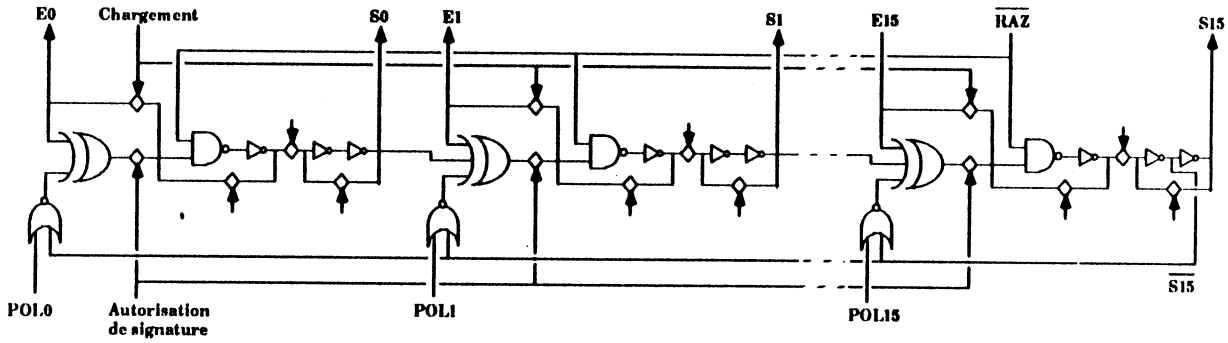


Figure 4.13 : Schéma logique du registre de signature

- **AS** est un circuit combinatoire qui autorise ou non la saisie de données à signer sur l'un ou l'autre bus. Cette autorisation de signature est générée à partir du contenu de RTIS et des valeurs des champs de commandes INFA et INFB qui indiquent le type des informations circulant sur les bus durant un cycle. Ils sont codés de la manière suivante :

	0	1	2	3	4	5
INFA	Adresse programme	Autre adresse	Donnée interne	Code instruction	Donnée lue	Donnée écrite

	0	1	2	3
INFB	Donnée lue	Donnée interne	Adresse	Donnée écrite

Il faut noter que lorsque SIGN est accédé comme un registre normal, aucune signature n'est possible. Les équations des commandes d'autorisation de signature générées par AS sont :

$$\text{Sign ALPHA} = \overline{\text{SIGNEA}} \cdot \overline{\text{SIGNSA}} \cdot \overline{\text{SGNEB}} \cdot \overline{\text{POLEA}} \cdot \overline{\text{POLSA}} \cdot \overline{\text{POLEB}} \cdot \overline{\text{RTISEA}} \cdot \overline{\text{RTISSA}} \cdot \overline{\text{RTISEB}} \cdot \overline{\text{RTIS\_0}} \cdot [\text{RTIS\_2} \cdot (\text{INFA\_0} + \text{INFA\_1} + \text{INFA\_2} + \text{INFA\_3} + \text{INFA\_4} + \text{INFA\_5}) + \text{RTIS\_3} \cdot \text{INFA\_3} + \text{RTIS\_4} \cdot \text{INFA\_5} + \text{RTIS\_5} \cdot \text{INFA\_4} + \text{RTIS\_6} \cdot \text{INFA\_1} + \text{RTIS\_7} \cdot \text{INFA\_0}]$$

$$\text{Sign BETA} = \overline{\text{Sign\_ALPHA}} \cdot [\text{INFB\_3} \cdot \text{RTIS\_4} + \text{RTIS\_2} \cdot \text{INFB\_0} + \text{RTIS\_5} \cdot \text{INFB\_2} + \text{RTIS\_6} \cdot \text{RTIS\_1} \cdot (\text{INFB\_0} + \text{INFB\_1} + \text{INFB\_2} + \text{INFB\_3})]$$

Evaluation

POL et **RTIS** sont des registres de type simple ayant accès aux deux bus, la taille d'une de leur cellules est donc de 8 transistors.

Une cellule de **SIGN** est composée des éléments suivants :

- Une cellule de registre maître/esclave comprenant deux points de mémorisation dont le premier inverseur est remplacé par une porte NAND pour permettre une remise à zéro et où une porte de transfert est implantée sur le rebouclage (deux fois 12 transistors).
- Deux portes de transfert pour le décalage (4 transistors) et quatre transistors N d'accès aux bus.
- Une porte OU Exclusif à trois entrées (24 transistors).
- Une porte NOR pour la programmation du polynôme de signature (4 transistors).

D'où la taille d'une cellule de SIGN : **56** transistors.

La taille de **AS** est de **84** transistors.

1.2.3 - Unité Arithmétique et Logique (U.A.L) et Dispositif de masquage (D.M)

* ***U.A.L***

L'U.A.L est réalisée par un opérateur combinatoire défini ci-après.

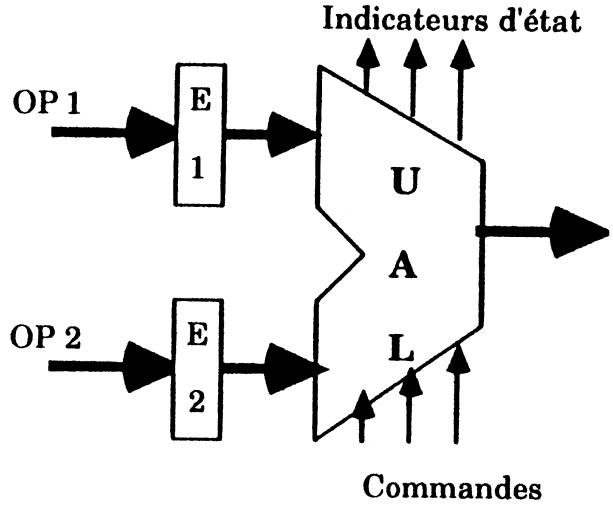


Figure 4.14 : Unité arithmétique et Logique

OP1 et **OP2** sont deux opérandes de seize bits. Le résultat **R** est également codé sur seize bits en complément à deux. Les opérations réalisées par l'U.A.L sont décrites dans le tableau de la figure 4.15.

Mnémonique	OP1	OP2	Résultat R
ADD	A	B	A plus B
ADDC	A	B	A plus B plus C
SUB	A	B	A moins B
SUBC	A	B	A moins B moins C
INC	A		A plus 1
DEC	A		A moins 1
NOT	A		$\overline{A}$
AND	A	B	A . B
OR	A	B	A + B
NAND	A	B	$\overline{A . B}$
NOR	A	B	$\overline{A + B}$
EOR	A	B	A $\oplus$ B
LSR	A		Décalage logique à droite
LSL	A		Décalage logique à gauche
ASL	A		Décalage arith. à gauche
ASR	A		Décalage arith. à droite

Figure 4.15 : Opérations de l'U.A.L

Remarque : Les opérations de décalage ne sont pas réalisées par l'U.A.L elle même, mais par le registre tampon E1 qui est monté en registre à décalage bidirectionnel. On notera que les deux opérations de décalage à gauche (logique et arithmétique) sont identiques.

Quatre indicateurs d'état sont générés par l'U.A.L :

N : N = 1 résultat négatif
N = 0 résultat positif ou nul

Z : Z = 1 résultat nul
Z = 0 résultat différent de zéro

C : C = 1 l'opération arithmétique a généré une retenue sortante
C = 0 pas de retenue sortante

V : V = 1 l'opération arithmétique a provoqué un débordement
V = 0 pas de débordement

Trois bascules sont utilisées pour mémoriser Z, N, ainsi que l'indicateur L (généré par le dispositif de masquage) pour permettre au contrôleur d'effectuer des tests sur des résultats d'opérations.

Structure de l'U.A.L

Le noyau de l'U.A.L est un additionneur seize bits dont les deux entrées sont générées par un circuit combinatoire Q. Ce circuit a A et B (Alpha et Beta) comme entrées. Ses sorties permettent la réalisation des différentes fonctions de l'U.A.L. On pourrait envisager l'utilisation d'une anticipation de retenue pour améliorer les performances, mais se posent alors de gros problèmes pour la reconfiguration. De plus, les simulations électriques ont montré que les objectifs de performances sont atteints sans un tel dispositif. L'U.A.L est composée de six portes complexes comme indiqué dans la figure 4.16.

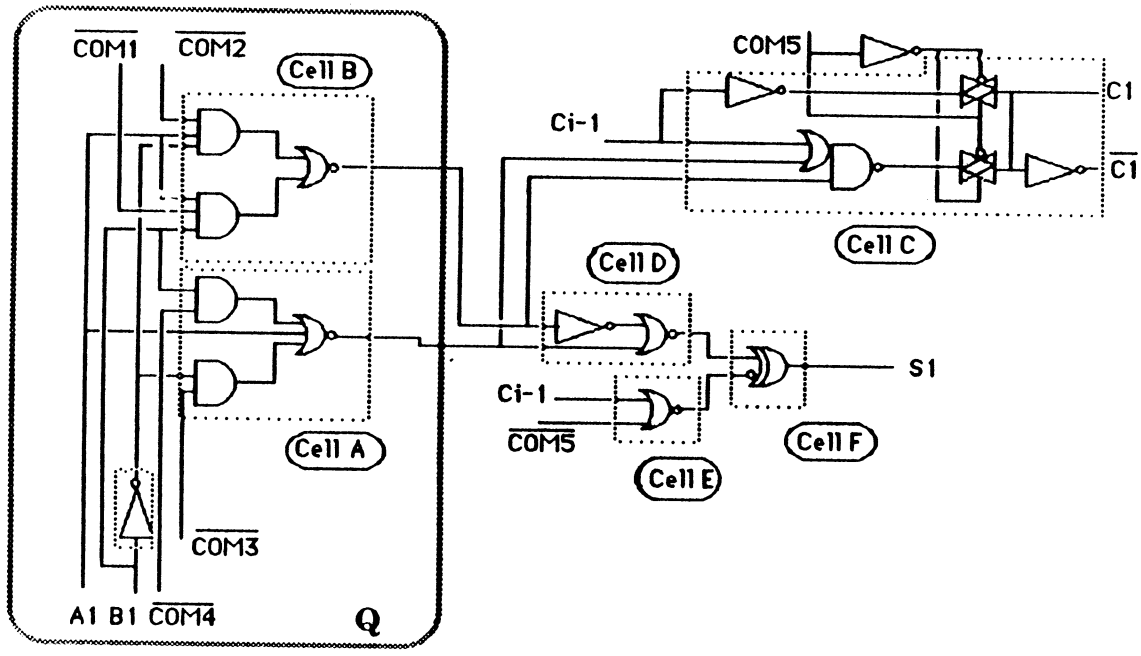


Figure 4.16 : Schéma logique de l'U.A.L

Les registre E1 et E2 sont utilisés comme tampons d'entrée de l'U.A.L. En vue d'une simplification du test et de la reconfiguration de celle-ci, les opérations de décalage sont réalisées par le tampon E1 qui est un registre à décalage bidirectionnel.

Evaluation

La taille d'une tranche de l'U.A.L est de **76** transistors.

* ***Dispositif de masquage***

La présence d'instructions permettant de travailler sur des champs d'opérandes nécessite d'utiliser un dispositif de masquage faisant en sorte que seuls les bits du résultat correspondant à des bits à "1" dans le masque soient pris en compte. Le masque de seize bits est chargé en cours d'exécution de l'instruction dans un registre spécialisé : RMA. Pour les opérations non masquées, la sortie de l'U.A.L sort directement sur le bus (figure 4.17).

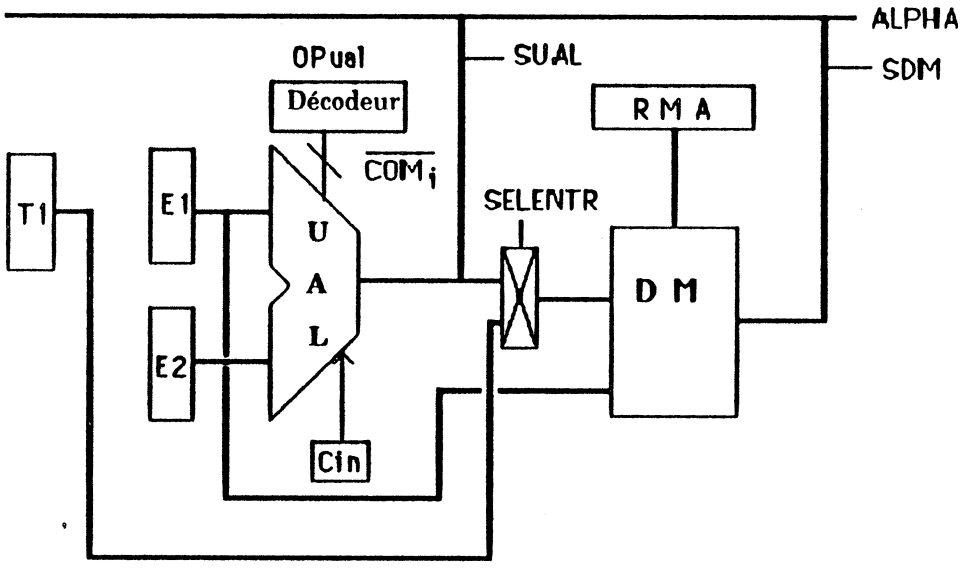


Figure 4.17 : Schéma général de l'ensemble U.A.L - Dispositif de masquage

Quand une opération U.A.L (ANDM, ORM, NANDM, NORM, EORM, COMPM) est effectuée,

- E1 contient la valeur du registre à modifier (reg. dest.). Les bits non masqués de ce registre sont envoyés sur le bus (ils correspondent aux bits à "0" du masque).
- SUAL est le résultat de l'opération entre l'opérande et le registre destination. Les bits non masqués de ce résultat sont envoyés sur le bus (ils correspondent aux bits à "1" du masque).

Dans le cas d'une instruction de transfert (LOADM ou STOREM), le registre tampon T1 est utilisé à la place de SUAL.

- Pour LOADM, T1 contient l'opérande et E1 le contenu du registre à modifier.
- Dans le cas de STOREM, T1 contient la valeur du registre et E1 celle du mot mémoire à modifier. T1 et SUAL sont sélectionnés par la commande SELENTR.

Le dispositif de masquage génère deux indicateurs d'état :

H : H = 1 tous les bits non masqués du résultat valent "1"
H = 0 autrement

L : L = 1 tous les bits non masqués du résultat valent "0"
L = 0 autrement.

On notera que L et H ne sont pas modifiés quand rien n'est masqué. La figure 4.18 présente le schéma logique d'une tranche du dispositif de masquage.

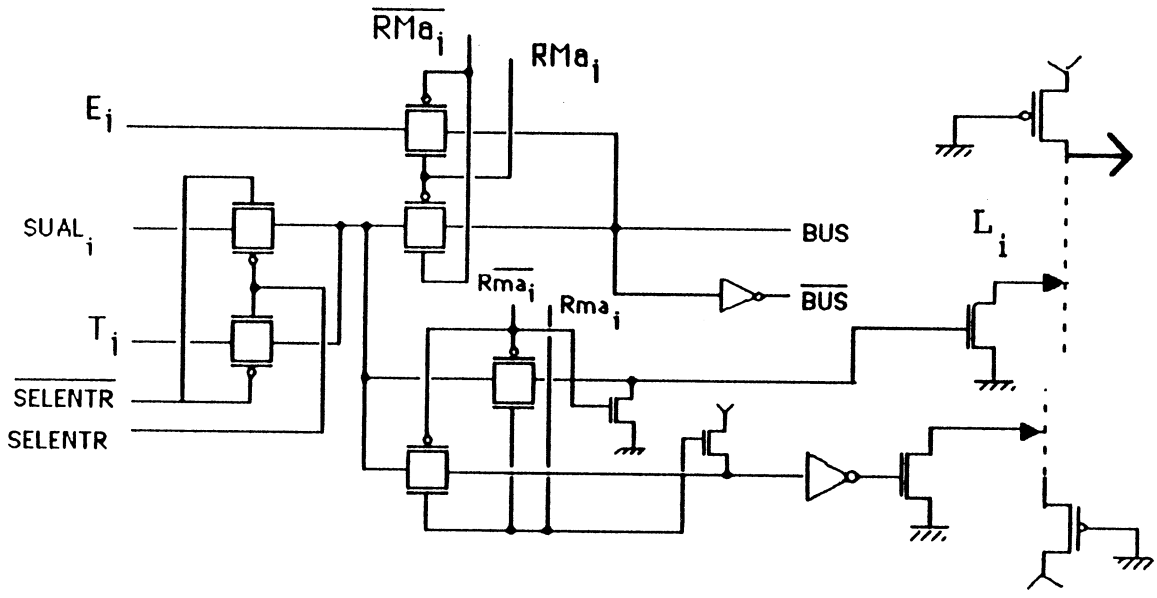


Figure 4.18 : Schéma logique d'une tranche de D.M.

Evaluation

La taille d'une tranche de DM est de **24** transistors.

1.2.4 - Multiplieur

Les calculs d'adresse nécessaires pour accéder aux éléments de matrice imposent, pour des raisons de rapidité, l'utilisation d'un multiplieur réalisant l'opération **A * B + C + D** (voir annexe 2, § 1.2.1). Dans la machine HSURF, ce multiplieur est réalisé de la manière suivante.

L'algorithme de multiplication est basé sur les additions partielles. On a ainsi, par exemple :

```
          1 0 1 0
        * 1 1 0 1
        ---------
          1 0 1 0
      + 0 0 0 0
        ---------
        0 1 0 1 0
    + 1 0 1 0
      -----------
      1 1 0 0 1 0
+     1 0 1 0
      -----------
    1 0 0 0 0 0 1 0
```

Une telle suite d'additions peut être réalisée par un circuit combinatoire cellulaire dont les éléments de base sont des additionneurs complets organisés comme suit :

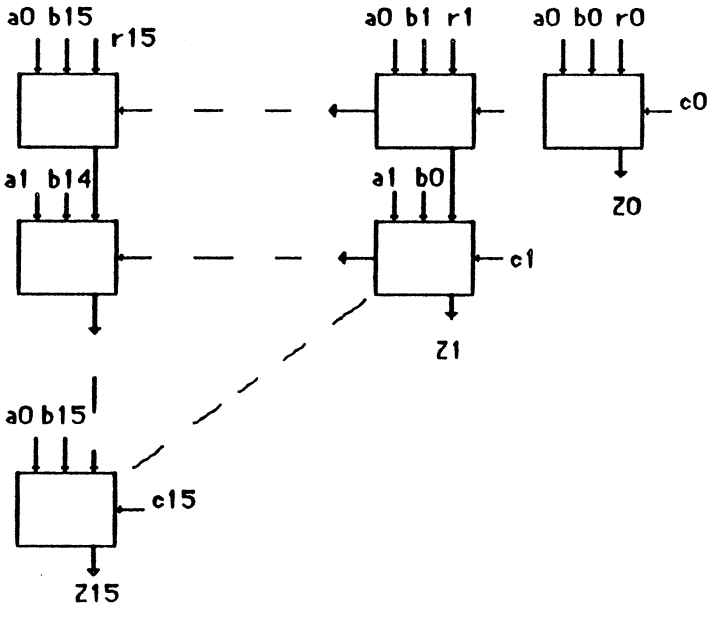


Figure 4.19 : Organisation du multiplieur cellulaire

La cellule de base (additionneur complet) est la suivante :

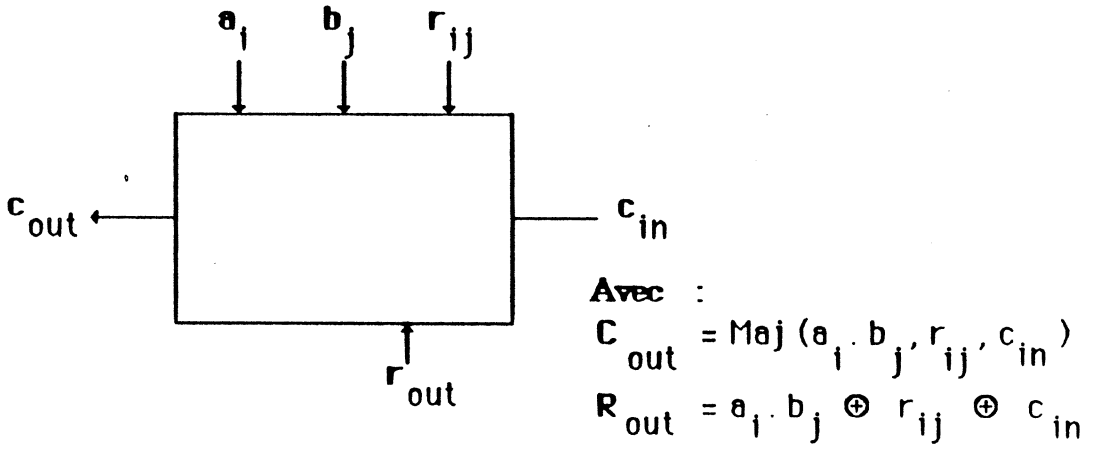


Figure 4.20 : Cellule de base du multiplieur

La multiplication de deux nombres de n bits donne comme résultat un nombre de 2*n bits, or, dans la définition des matrices, on limite la taille d'une matrice à au plus une page de 64K-mots, ce qui entraîne que seule une adresse sur seize bits est nécessaire. Ceci permet de ne pas intégrer l'ensemble du multiplieur mais seulement sa partie triangulaire supérieure (qui permet de calculer les seize bits de poids faibles) comme cela est montré dans la figure 4.19. Les cellules calculant les poids forts du résultat sont remplacées par un circuit de détection de débordement (figure 4.21) qui, lorsqu'il détecte un résultat sur plus de seize bits, met à "1" la bascule ALARME du microprocesseur entraînant par ce

moyen un déroutement et une attente de signal externe de reprise. Cette limitation permet de n'avoir que (16 + 15 + ... + 1) = 136 cellules au lieu des 256 du multiplieur complet. Les opérandes de la multiplication sont mémorisés dans les quatre registres tampons A, B, C et D.

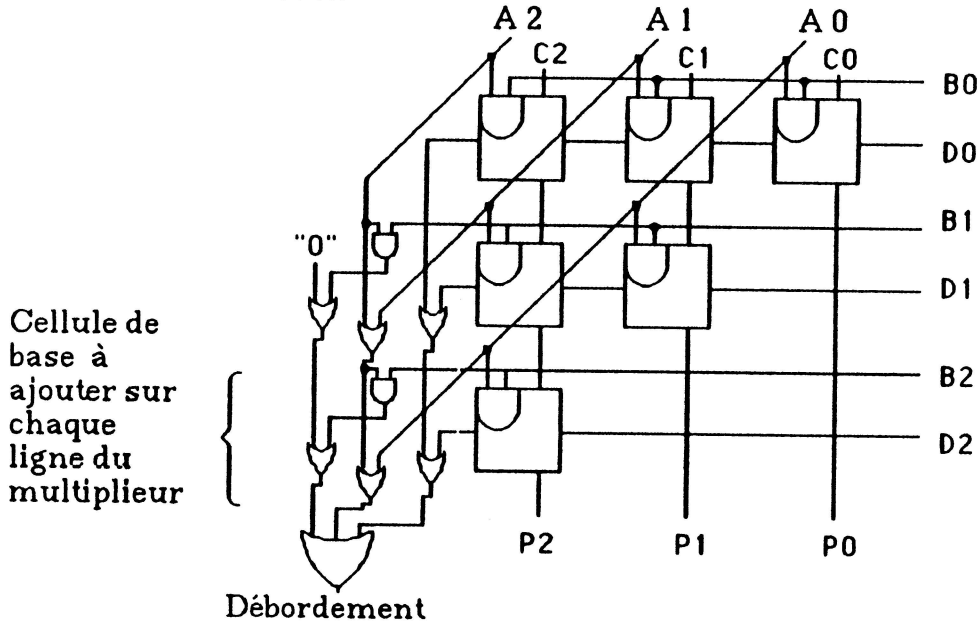


Figure 4.21 : Multiplieur cellulaire et dispositif de détection de débordement

Nous verrons au paragraphe 2.2 que c'est une autre réalisation de ce multiplieur qui a été choisie pour HYETI à cause des problèmes de reconfiguration. Cette autre possibilité de réaliser l'opération A*B+C+D fonctionne selon le principe suivant : il s'agit de remplacer la multiplication parallèle de la première solution (tous les opérandes rentrent en parallèle dans le multiplieur) par un algorithme parallèle/série où certains opérandes rentrent en série. Avec l'opérateur décrit dans le schéma de la figure 4.22 (version 4 bits), l'ordre des opérations est le suivant :

instant t(1) : --> $a_3.b_0 + c_3 + d_3$ --> R_0 (1 = instant 1)

t(2) --> $a_2.b_0 + c_2 + d_2$ --> $R_0(2)$

et $a_2.b_1 + R_0(1)$ --> $R_1(1)$ t(3)

--> $a_1.b_0 + c_1 + d_1$ --> $R_0(3)$

et $a_1.b_1 + R_0(2)$ --> $R_1(2)$

et $a_1.b_2 + R_1(1)$ --> $R_2(1)$ t(4)

--> $a_0.b_0 + c_0 + d_0$ --> $R_0(4)$

et $a_0(1) + R_0(3)$ --> $R_1(3)$

et $a_0.b_2 + R_1(2)$ --> $R_2(2)$

et $a_0.b_3 + R_2(1)$ --> $R_3(1)$

Après quatre cycles d'horloge nous obtenons :

$R0 = a_1.b_0 + c_0 + d_0$
$R1 = a_0.b_1 + a_1.b_0 + c_1 + d_1 R2 = a_0.b_2 + a_1.b_01 + a_2.b_0 + c_2 + d_2$
$R3 = a_0.b_3 + a_1.b_21 + a_2.b_1 + a_3.b_0 + c_3 + d_3$

La réalisation de ces opérations nécessite deux horloges. Elles commandent les registres et l'accès des opérandes à chaque tranche de l'opérateur. Seuls les bits de B sont tous présents durant le calcul. Le résultat sort à travers un registre maître/esclave. Ceci permet la lecture du résultat partiel à l'instant (t-1), à l'instant (t).

La cellule centrale est le même additionneur 1 bit que dans la version précédente. Par contre les registres A, C et D deviennent des registres à décalage, et deux registres à décalage supplémentaires font leur apparition :

RESM : contient le résultat de la multiplication

CONT : registre de contrôle du multiplieur, il permet de compter les cycles d'opération à l'aide d'un "jeton" ("1" logique) qui se décale à chaque cycle.

Evaluation

La taille d'une cellule d'additionneur est de **38** transistors. Les registres à décalage sont mono-directionnels avec en plus les transistors N nécessaires à leurs accès aux bus, soit 4 transistors pour A, C, D et 2 pour CONT et RESM. B est un registre simple ayant accès aux deux bus (4 transistors de connexion). Les dispositifs de détection de débordement et de génération d'horloge sont négligeables par rapport à la taille du reste du multiplieur, ils ne sont donc pas pris en compte dans l'évaluation.

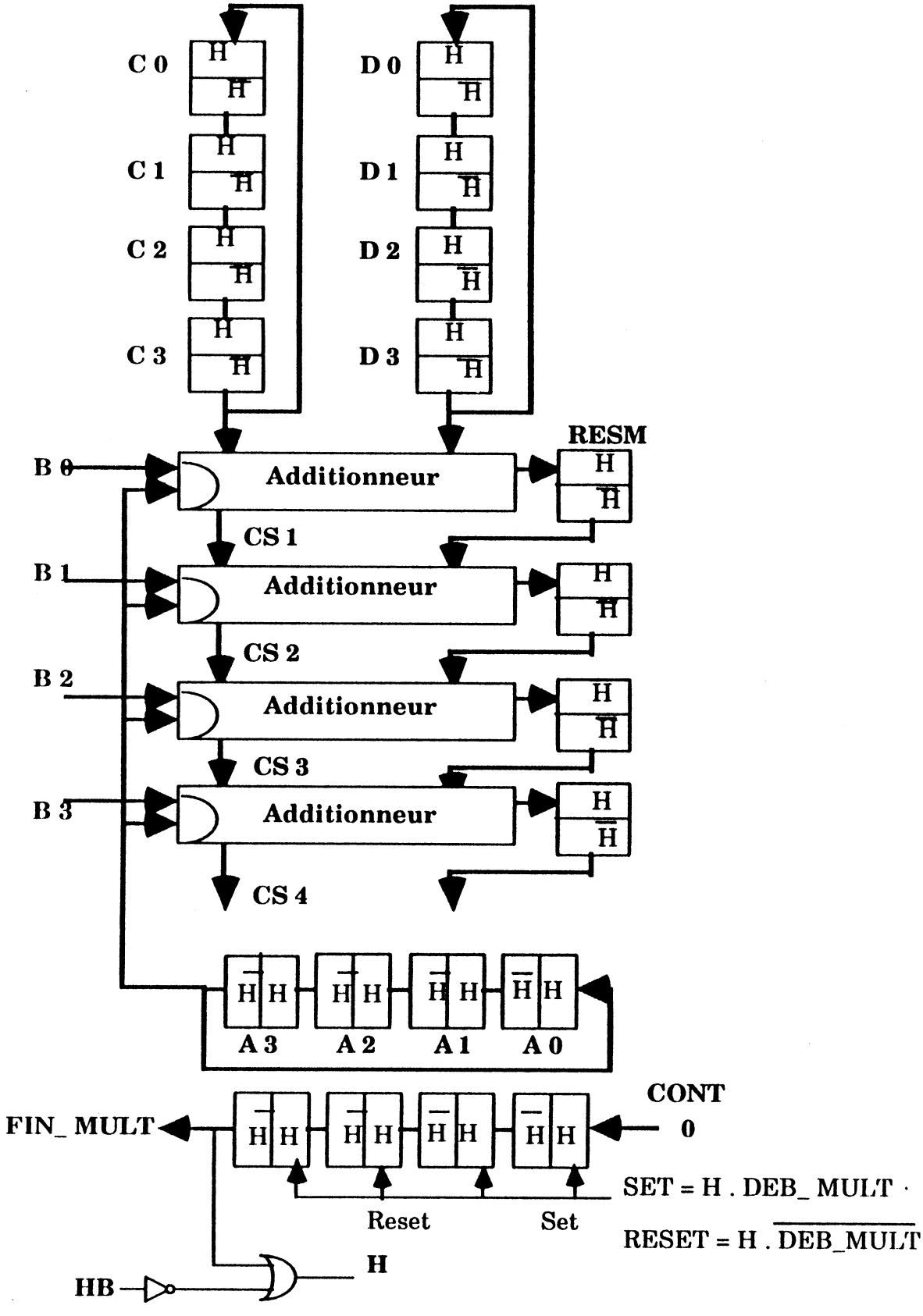


Figure 4.22 : Multiplieur parallèle - série

1.2.5 - Interface mémoire avec l'extérieur, temporisation

Le bus externe du microprocesseur HYETI est compatible avec les périphériques non lents de la famille 6800. La compatibilité avec les périphériques lents ne peut pas être garantie car HYETI ne comporte pas de patte similaire au DBE du 6800. L'interface externe de notre circuit est la suivante :

A0 - A15 : Bus adresses de seize bits
P0 - P3 : Bus pages de quatre bits
D0 - D3 : Bus données de seize bits
Vdd : + 5 V
CLK : Horloge externe (entre 10 et 16 MHz)
VMA : Adresse mémoire valide
R/WB : Ce signal indique aux périphériques et à la mémoire si l'unité centrale fait une *Lecture* (signal haut) ou une *Ecriture* (signal bas).
RESET : Accès à la bascule RESET pour remettre le microprocesseur "à zéro" (page 0, adresse0) ou, en cas d'alarme, pour le redémarrer à une adresse connue (page 0, adresse 2).
ALARME : cette sortie est mise à l'état haut lorsqu'un code opération invalide, un paramètre invalide, ou un débordement du multiplieur est détecté.
Broches de test Plusieurs broches de test sont présentes pour contrôler le test hors–ligne du microprocesseur en permettant un accès direct aux entrées et aux sorties des PLAs.

L'interface avec la mémoire est réalisée comme indiqué dans la figure 4.23.

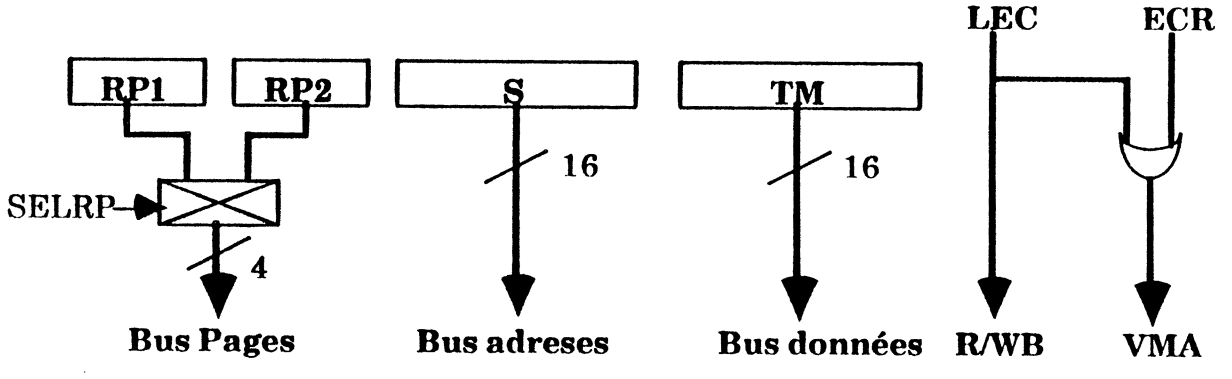


Figure 4.23 : Interface mémoire

Les diagrammes de temporisation interne et externe sont donnés dans la figure 4.24.

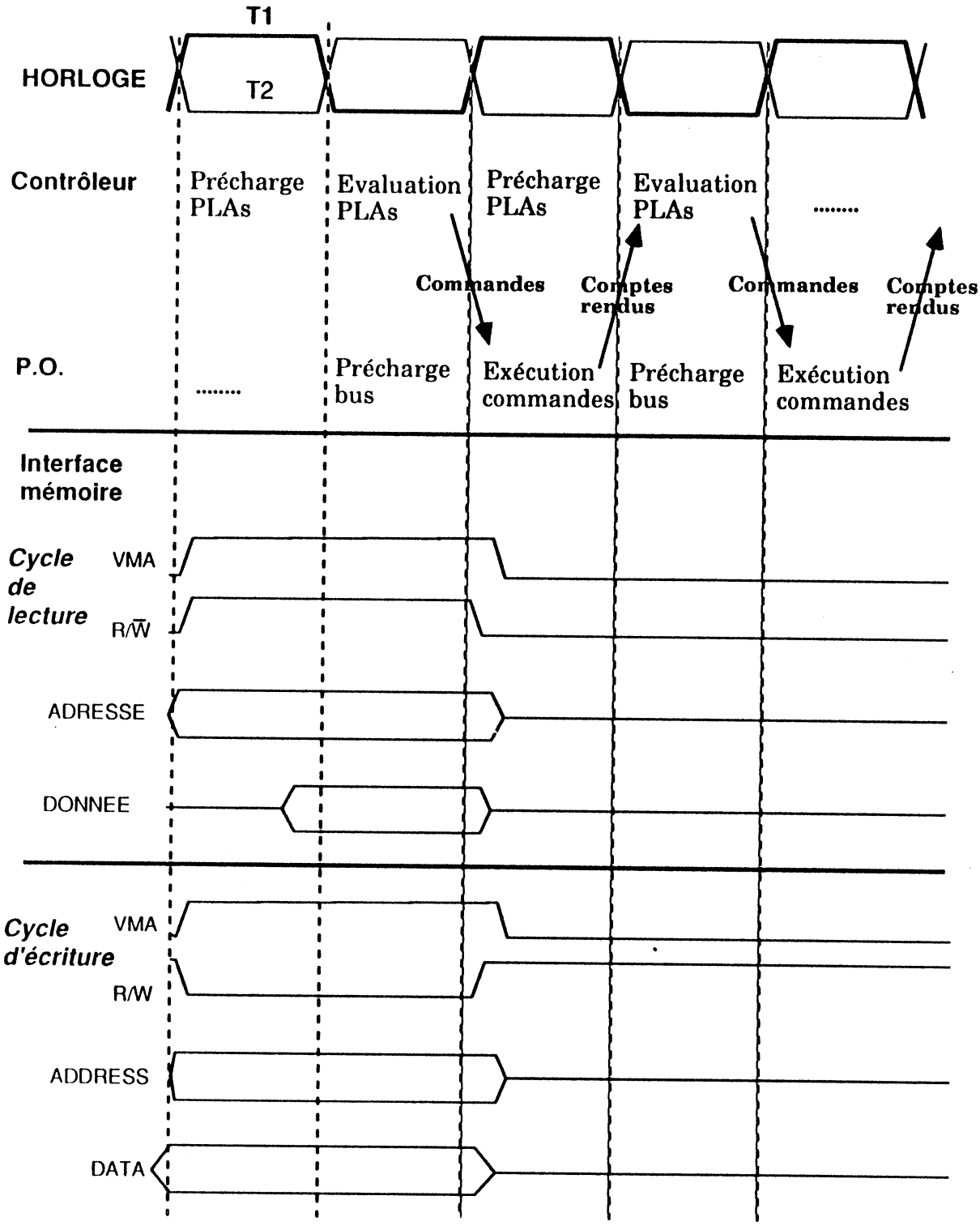


Figure 4.24 : Diagrammes de temporisation de HYETI

1.3 - Structure de la partie contrôle

L'architecture retenue pour la machine HSURF (figure 4.25) est une microprogrammation horizontale (informations de commande et de séquencement présentés dans chaque mot de la ROM).

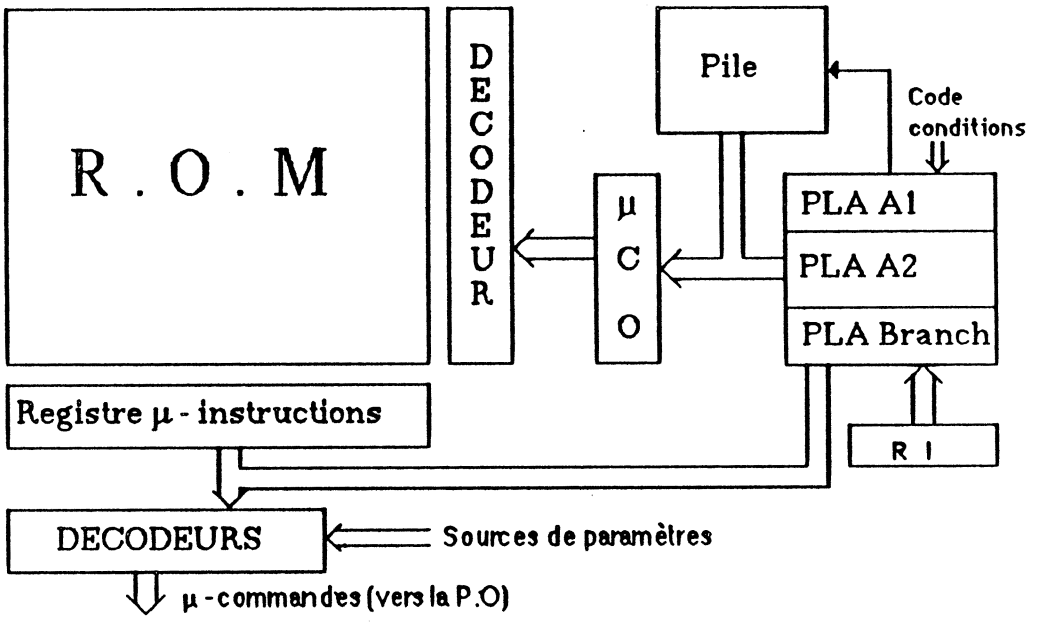


Figure 4.25 : Architecture de la partie contrôle de HSURF

Ce contrôleur comporte les éléments suivants :

* *Mémoire morte (ROM)* : elle contient le microprogramme sous la forme de mots comportant trois champs :
  - un champ de micro–commandes décrivant les opérations à effectuer par la partie opérative.
  - un champ adresse suivante (AS) servant lors des branchements inconditionnels ou des sauts à un sous–microprogramme. Il contient l'adresse de la micro–instruction qui doit être exécutée après la micro–instruction courante. Ce champ a une longueur de huit bits.
  - un champ condition de branchement utilisé pour sélectionner la source de l'adresse de la micro–instruction suivante. En fonction de son codage, on sélectionnera soit le champ AS, soit la sortie du PLA de branchement, soit la sortie de l'un des deux PLAs de décodage (PLA A1 ou PLA A2).

Cette architecture n'est pas adaptée pour la reconfiguration ; en effet elle présente tous les défauts de manque de régularité globale et de difficulté de localisation précise des défauts. De plus, une telle architecture entraîne de nombreuses interconnexions qui peuvent elles-mêmes être sujettes à défauts. Il a donc fallu changer l'architecture de la partie contrôle pour HYETI, de manière à y introduire plus de régularité et moins d'éléments différents. La structure retenue est un contrôleur multi-PLAs hiérarchisé réalisant l'algorithme de contrôle dont l'organigramme est donné dans l'annexe 3. Le schéma de cette partie contrôle est présenté dans la figure 4.26 ci-dessous.

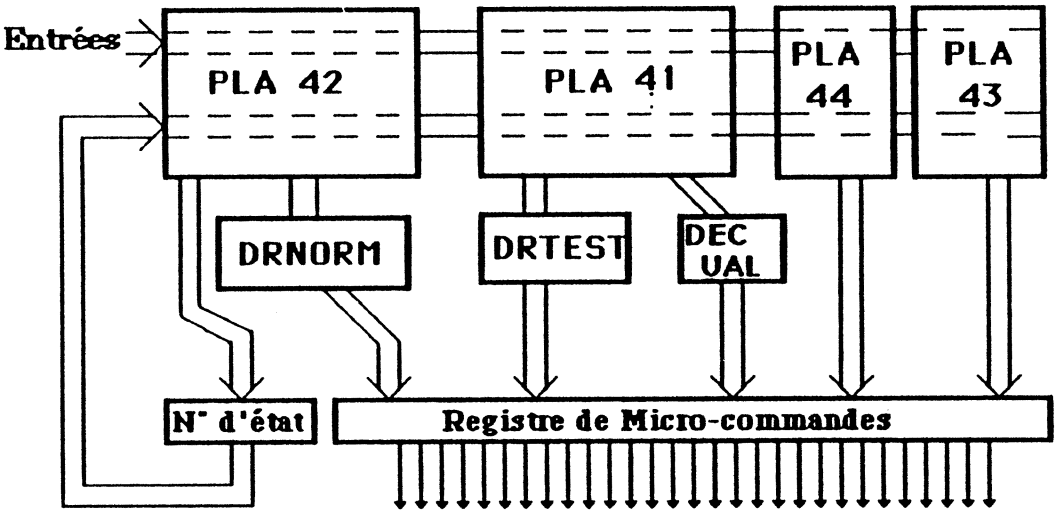


Figure 4.26 : Architecture du contrôleur de HYETI

Comme on peut le voir dans sa description détaillée, le jeu d'instruction autorise de nombreux usages de numéros de registres comme paramètres. La génération des commandes de ces registres nécessite un décodage du numéro de registre paramétré, ce qui impose la présence de monômes exclusivement réservés au décodage, d'où une certaine perte de place. En séparant ce décodage, la hiérarchisation permet d'éviter la présence de monômes "inutiles" dans chaque niveau ce qui entraîne un gain de place. La structure du contrôleur hiérarchisé comporte donc deux niveaux de PLAs :

- le premier niveau est celui de la génération des variables internes et des commandes individuels. Ce niveau comprend 4 PLAs : PLA 41, PLA 42, PLA 43 et PLA 44.

- le deuxième niveau correspond à la génération des commandes des registres dont les numéros sont des paramètres d'instruction ainsi qu'au décodage des commandes de l'UAL. La description détaillée des

commandes générées par la partie contrôle est donnée dans l'annexe 4 avec la répartition de ces commandes dans les différents PLAs. La structure électrique des PLAs est celle indiquée dans la figure 4.27 ci-dessous.

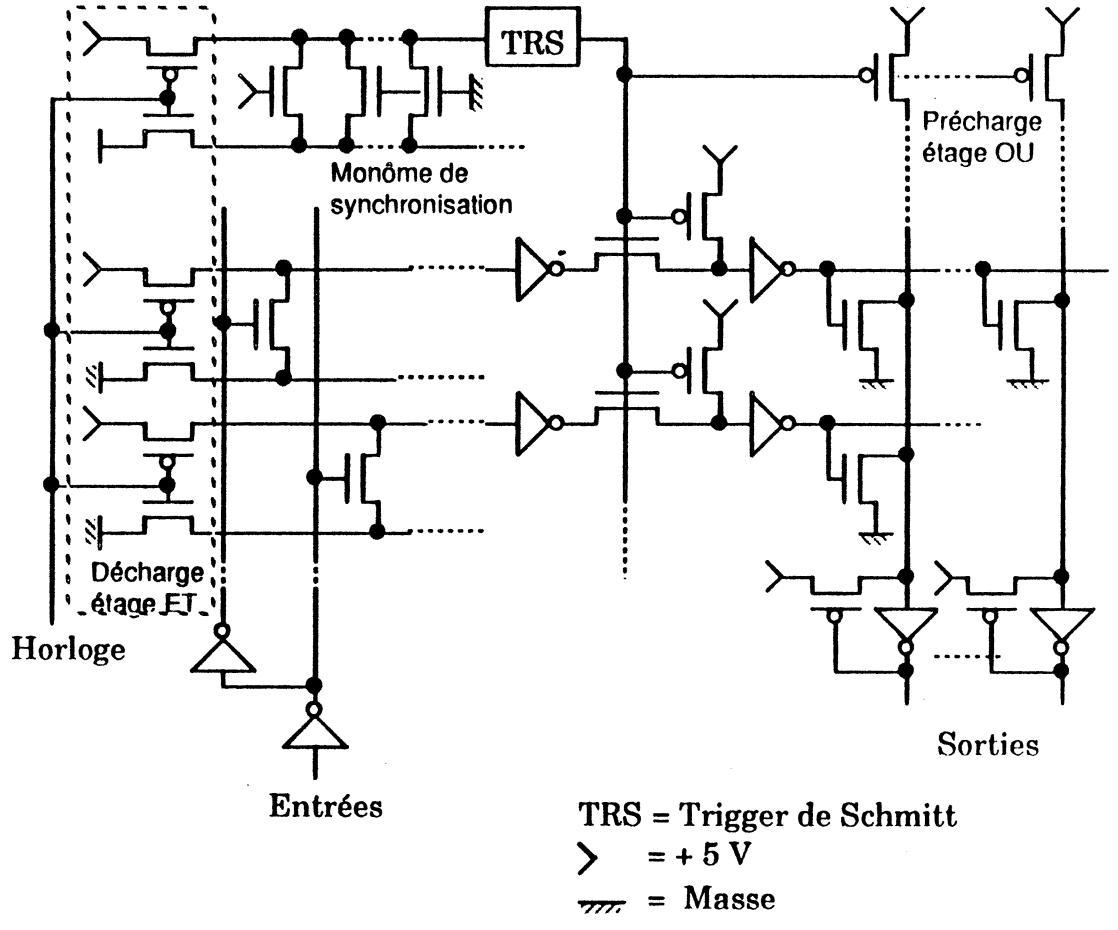


Figure 4.27 : Structure électrique des PLAs de HYETI

A partir de là, il est possible d'évaluer la taille des PLAs en donnant une valeur aux paramètres T_{KM} et T_{KES} de l'expression 3.2. Dans notre cas ces paramètres sont calculés pour le plus gros des PLAs :

T_{KM} = **36** pour les dispositifs de précharge et de décharge des monômes, ainsi que pour les amplificateurs entre l'étage ET et l'étage OU.

T_{KES} = **36** pour les inverseurs des entrées, la précharge et les amplificateurs des sorties.

Les autres éléments du contrôleur sont de deux types :

- le registre de micro–commandes et de rebouclage qui est constitué de cellules maître/esclave et qui mémorise les commandes géné-

rées en fin de traversée de la partie contrôle.

- des registres de signature sur la boucle de réaction (n° d'état suivant) et sur les micro-commandes pour compacter les informations y circulant de manière à faciliter le test.

La synchronisation de la PO et de la PC en fonctionnement normal se fait à l'aide d'une horloge à deux phases T1 et T2. Cette synchronisation est présentée dans les diagrammes de temporisation de la figure 4.24.

2 - Reconfiguration, Architecture finale de HYETI

2.1 - Paramètres généraux

Comme nous l'avons vu au chapitre 1, le choix d'une stratégie de reconfiguration dépend de données telles que la taille des blocs reconfigurables et l'augmentation de surface autorisée. Ces deux paramètres dépendent de la technologie utilisée pour la fabrication du circuit. Les différents circuits réalisés au cours du projet ESPRIT 824 WSI sont fabriqués par Thomson Semiconducteurs en technologie HCMOS 3C (CMOS 1,3 µm de largeur de grille).

L'objectif à atteindre dans notre cas est un rendement de fabrication compris entre 40 et 50 % pour le microprocesseur reconfigurable (alors qu'il est évalué à environ 15 à 20 % pour une version non reconfigurable. L'autre donnée de départ du projet est que cet objectif doit être atteint sans augmenter la surface du circuit de plus de 25 à 30 %. Il reste alors à définir le dernier paramètre, à savoir la taille des blocs reconfigurables. Cette taille dépend de la valeur fixée pour P(X≤1 ; S) car cette valeur conditionne elle même la probabilité d'avoir un circuit bon après reconfiguration.

En effet, le rendement de fabrication du circuit (c'est à dire la probabilité qu'un circuit soit bon) peut s'exprimer comme suit (dans le cas où l'on ne veut pas corriger plus d'un défaut par bloc) :

$$R = P_{nr} \cdot P(X \leq 1 ; S_1).P_{R1} \ldots P(X \leq 1 ; S_i).P_{Ri} \ldots P(X \leq 1 ; S_{nb\ blocs}).P_{Rnb\ blocs}$$

où P_{Ri} est la probabilité de pouvoir réparer le bloc i et P_{nr} la probabilité de ne pas avoir un défaut dans les parties non-reconfigurables du circuit. En l'absence de toute expérience quant aux résultats réels de la reconfiguration, il est difficile de donner une valeur précise pour les P_{Ri}. Ils dépendront notamment du nombre de défauts affectant des parties non reconfigurables des blocs tels que

les lignes de commandes dans la partie opérative, ainsi que du nombre de défauts affectant, par exemple, simultanément deux tranches voisines de partie opérative, ou deux monômes d'un PLA. Si l'on se base sur des réalisations antérieures, la surface occupée par les lignes de commandes par rapport au reste de la P.O. est de l'ordre de 5 à 8 %. En prenant en compte les autres risques de défauts non réparables, on peut faire l'hypothèse que $\mathbf{P_{Ri}= 0,9}$ pour tous les autres blocs.

La surface totale d'un microprocesseur 16 bits tel que HYETI fabriqué en technologie 1,3 µm peut être grossièrement évaluée (d'après celle des circuits de complexité similaire existant dans le commerce) à environ 30 mm^2. On peut estimer la proportion des parties non reconfigurables du circuit (interface de commandes, plots d'E/S, connexions diverses) a environ 20 % de la surface totale, soit dans notre cas 6,5 mm^2. Ceci va nous permettre d'évaluer P_{nr} à $P(X = 0 ; 6,5\ mm^2) <=> \mathbf{P_{nr} = 0,92}$.

Il ne reste plus alors qu'à trouver S tel que

$R = 0,5 \leq 0,92.(P(X \leq 1 ; S)^{nb\ blocs}.0,9^{nb\ blocs})$. La portion reconfigurable du circuit est d'environ 23,5 mm^2. Une première division en

deux blocs de 12 mm^2 donne $R = 0,92 .(0,9.0,78)^2 = 0,45$

pour *trois blocs de 7,5 mm^2* , on a $R = 0,92.(0,9.0,9)^3 = 0,49$

pour *quatre blocs de 6 mm^2*, on a $R = 0,92.(0,9.0,928)^4 = 0,44$

pour *cinq blocs de 4,7mm^2*, on a $R = 0,92\ (0,9.0,95)^5 = 0,42$.

On voit donc que la taille de blocs reconfigurables permettant d'espérer les meilleurs résultats est d'environ 7,5 mm^2. L'évaluation des différentes stratégies de reconfiguration se faisant à un niveau architectural, en estimant la taille du circuit par son nombre de transistors, il nous faut donner une correspondance entre la surface du bloc reconfigurable et son équivalent en termes de transistors. Là encore, il va falloir différencier partie contrôle et partie opérative.

La partie contrôle avec ses structures de PLAs régulières et plus dense que la partie opérative et la densité de transistors peut y être estimée à 12600 unités–transistor par mm^2, ce qui donne une taille de blocs reconfigurables de 94000 unités–transistor (on parle ici d'"unité–transistor" car l'évaluation n'a rien à voir avec le nombre réel de transistors implantés, mais elle donne une bonne estimation de la surface occupée).

Dans la partie opérative, la présence de bus en métal supérieur impose de sévères contraintes pour l'implantation (problème de marche) et la densité de transistors peut y être évaluée à 2700 transistors par mm^2, soit une taille de blocs reconfigurables de 20000 transistors. A partir de là nous allons pouvoir déterminer les stratégies de reconfiguration à appliquer au microprocesseur HYETI.

2.2 Reconfiguration de la partie opérative

Comme cela a déjà été remarqué précédemment, certaines parties de la partie opérative ne pourront pas être rendues reconfigurables. Il s'agit principalement des lignes et de l'interface de commande. Ceci est dû à leur personnalisation qui empêche tout échange (sans parler des problèmes de reconnexion que poseraient les différentes prises de contacts situées sur les lignes de commandes).

Nous allons tout d'abord évaluer la **taille de la partie opérative sans dispositif de reconfiguration**.

* ***Registres*** :

Les registres généraux, CO, PP, RM0, RM1, T1, T2, RP1, RP2, TM, S, RMA, Flags sont considérés comme identiques pour l'évaluation. Ce sont des registres simples ayant accès aux deux bus, leur taille est donc (pour une tranche) : $T_r = 8 * 20 + 12$ (pour RE) <=> $\mathbf{T_r = 172}$ transistors. A cette valeur on ajoute celle de la logique de bus $\mathbf{T_b = 496}$.

* ***Dispositif de signature*** :

POL, RTIS : 2 * 8 = 16 transistorsSIGN : 56 transistorsD'où la taille d'une tranche de dispositif de signature $\mathbf{T_s = 72}$ transistors. Cette évaluation ne prend pas en compte le circuit d'autorisation de signature (84 transistors) qui n'a pas une structure en tranche.

* ***U.A.L et Dispositif de masquage*** :

Tampons d'entrée : E2 est un registre simple avec accès à Alpha et Beta (soit une taille de 8 transistors), E1 est un registre à décalage bi–directionnel n'ayant accès qu'à Alpha (20 transistors).

U.A.L : 76 transistors

D.M : 24 transistors

Ce qui donne la taille d'une tranche de l'ensemble : $\mathbf{T_{ud} = 128}$ transistors.

* ***Multiplieur :***

Multiplieur cellulaire :

Tampons d'entrée : Dans cette version du multiplieur les tampons d'entrée sont tous des registres simples ayant accès aux deux bus, ce qui fait 32 transistors.

Multiplieur : 38 transistors par cellule, pour 136 cellules, d'où une taille totale de 5168 transistors, et une moyenne par tranche de 323 transistors et une taille maximale de tranche de 608 transistors.

D'où la taille moyenne d'une tranche du multiplieur cellulaire

T_{mc} = **640** transistors.

Multiplieur parallèle - série :

Tampons d'entrée : A, C, D sont montés ici en registres à décalage mono–directionnels (20transistors par tranche), B ne change pas (8 transistors), d'où la taille des tampons d'entrée : 68 transistors.

Registres de Résultat et de contrôle : CONT et RESM sont des registres à décalage mono–directionnels n'ayant accès qu'à Alpha, soit une taille de 36 transistors.

Additionneurs : La taille d'une cellule d'additionneur est 38 transistors.

D'où la taille d'une tranche de multiplieur parallèle - série : T_{mps} = **142** transistors.

* ***Evaluation globale :***

P.O avec *multiplieur cellulaire* : 1508 transistors par tranche en moyenne, ce qui fait **24128 transistors pour seize bits**.

P.O avec *multiplieur parallèle - série* : 1010 transistors par tranche, d'où une taille de **16160 transistors pour seize bits**.

Ces résultats montrent deux choses :

La première est que, vu la taille obtenue pour une partie opérative de seize bits, le problème du choix d'une méthode de partitionnement ne se pose pas car il n'y a pas de quoi faire plus d'un bloc. Utilisant la structure en tranches de la P.O, ce bloc sera reconfiguré par ajout de tranches supplémentaires.Le second point concerne la structure du multiplieur. La taille du multiplieur cellulaire, et

surtout les difficultés de sa reconfiguration (il faut rajouter 22 transistors par cellule ce qui fait une taille totale de 8160 transistors) font qu'une telle architecture ne peut être employée ici. Nous utiliserons donc un multiplieur parallèle–série.

A partir de là nous pouvons évaluer la **taille d'une tranche de partie opérative munie de ses dispositifs de reconfiguration** (contournement de tranches en panne, connexions d'éléments non reconfigurables).

Dispositifs de contournement de tranches non utilisées : il en faut un par registre à décalage mono–directionnel, deux pour E1, un pour l'U.A.L, un pour le registre de signature, et un pour la chaîne d'additionneurs du multiplieur, soit 10 dispositifs nécessitant chacun 8 transistors, d'où T_{cr} = **80** transistors.

Dispositifs de connexion des éléments non reconfigurables : bien qu'ayant une structure en tranche, certains éléments de la partie opérative ne sont pas reconfigurables par ajout de tranches supplémentaires, car leurs tranches ne sont pas interchangeables. Il s'agit de RE, de RTIS, du registre instruction dont les bits ont une signification particulière. Egalement non reconfigurable, le dispositif d'autorisation de signature AS qui n'a pas une structure en tranche. Cela nécessite un dispositif de reconnexion pour chacun des registres. La taille du dispositif dépend du nombre de tranches rajoutées. Le moyen le moins coûteux de le réaliser est d'employer des transistors à grille flottante. Les valeurs des paramètres de l'expression 2.16 sont dans ce cas : n_b = **16**, n_l = **4** (car il y a deux connexions par bus).

- pour **K = 1** , on a pour un dispositif et pour une tranche : **16** transistors. D'où pour seize tranches : **256** transistors.
- pour **K = 2** , on a pour un dispositif et pour une tranche : **24** transistors. D'où pour seize tranches : **384** transistors.

De tels dispositifs sont également nécessaires pour les sorties directes des adresses, des numéros de page, et des données vers les bus externes, ce qui en rajoute trois.

* ***Taille d'une tranche de partie opérative reconfigurable :***

C'est la taille de la tranche de P.O non reconfigurable à laquelle on ajoute les dispositifs évalués ci–dessus, soit 1010 + 16*6 + 80 = **1186** transistors dans le cas où K=1, et 1010 + 24 * 6 + 80 = **1234** transistors dans le cas où K=2.

Ces résultats permettent de connaître la taille de seize tranches de partie opérative munies des dispositifs de reconfiguration :

pour **K=1**, on a T_{PO} = **18976** transistors.

pour **K=2**, on a T_{PO} = **19744** transistors.

* ***Choix du nombre de tranches de réserve :***

La limite de l'augmentation de surface pour la redondance (30% de la surface initiale) permet d'ajouter l'équivalent de 4848 transistors, ce qui amène la taille maximum de la partie opérative à **21008** transistors. Les tailles obtenues pour les deux configurations étudiés ci-dessus sont les suivantes :avec **une tranche de réserve : 20162** transistors avec **deux tranches de réserve : 22212** transistors.

Ces résultats montrent que c'est la solution avec une tranche en réserve qui est la meilleure d'après nos critères, en offrant la possibilité de corriger un défaut dans la partie opérative et en laissant une marge de sécurité au niveau de la précision de l'évaluation.

2.3 - Reconfiguration de la partie contrôle

La reconfiguration de la partie contrôle est basée sur l'ajout de redondance dans les PLAs. En effet, dans une structure multi–PLAs telle que celle que nous avons retenue, le nombre des interconnexions et leur singularité rend toute tentative de reconfiguration à leur endroit très difficile (problèmes de localisation des défauts et de remplacement des lignes en panne). De plus, la taille de ces interconnexions par rapport au reste du contrôleur ne justifie pas de tels efforts. Cet argument est encore plus vrai pour les registres de micro–commandes et de boucle de réaction.

Evaluation de la taille des PLAs sans rien pour la reconfiguration

Dans notre cas, les paramètres T_{KM} et T_{KES} sont évalués à **36** unités–transistor, ce qui nous donne pour les PLAs de HYETI (en unité-transistor) :

	E	S	M	Evaluation
PLA 41	30	30	315	42930
PLA42	26	30	212	27968
PLA43	11	8	39	3654
PLA44	16	12	31	4064
DRTEST	14	51	90	13194
DECUAL	4	10	15	1458
DRNORM	6	49	49	6949

soit un total de 100217 unités-transistor. Ce résultat impose de considérer la partie contrôle comme divisée en deux blocs reconfigurables. La façon dont la reconfiguration est possible pour les éléments d'une P.C rend cette division artificielle, chaque élément (PLA) recevant de la réserve. La quantité de lignes redondantes à implanter est variable : si on désire une surface ajoutée minimale, on peut se contenter d'une ou deux lignes par PLA (puisqu'elle permettent de corriger un défaut), par contre les limites de l'augmentation de surface peuvent permettre (vue la petite taille des lignes redondantes) d'en implanter beaucoup plus. Dans le cas de HYETI dont le contrôleur a été dessiné à L'Université de DARMSTADT (R.F.A), la solution retenue a été l'implantation de 10% de monômes de réserve dans les différents PLAs. Cette redondance est complétée par des entrées et des monômes supplémentaires pour le test et la localisation des défauts. Ce degré de redondance correspond à une augmentation de surface de l'ordre de 25%.

On trouvera dans l'annexe 4 le dessin des masques des principaux éléments du circuit dans sa version actuelle.

Conclusion

L'intégration de systèmes à base de microprocesseur sur un seul circuit présente de nombreux avantages pour l'utilisateur : réduction de l'encombrement, de la consommation d'électricité, augmentation de la fiabilité par suppression des sources de pannes sur les cartes, personnalisation aisée. La réalisation de tels circuits regroupant un microprocesseur et ses périphériques n'est pas sans poser de problèmes. Le principal est le rendement de fabrication qui est quasiment nul pour ces tailles de circuit. La solution consiste à implanter du matériel en réserve à l'intérieur du circuit sans toutefois dépasser la limite qui en annulerait l'effet. Cette reconfiguration du circuit dépend très fortement de son architecture.

Nous nous sommes intéressés ici aux stratégies de reconfiguration pour les circuits de type microprocesseur. Une évaluation de la répartition des défauts permet de déterminer la taille des blocs reconfigurables. A partir de là, différentes stratégies sont possibles selon les parties du circuit.

Dans la partie opérative, on répliquera des éléments totalement ou partiellement ou on ajoutera des tranches de bit de réserve. Le choix entre les différentes possibilités est guidé par les évaluations de leurs tailles.

La reconfiguration du contrôleur nécessitera l'emploi d'une architecture simple à base de PLAs. La reconfiguration se traduira par l'implantation de lignes de réserve dans les PLAs. La conception de cette partie du circuit est faite en utilisant une série d'outils de synthèse de contrôleur.

Appliquées au microprocesseur HYETI réalisé dans le cadre du projet ESPRIT 824, ces techniques permettent d'espérer voir le rendement de fabrication passer de 15-20% à 40-50% pour un surcoût en surface de l'ordre de 25 à 30%. Ces résultats seront validés par l'expérimentation sur les circuits en cours de fabrication. Ils permettent d'ore et déjà d'envisager la réalisation de systèmes à base de microprocesseurs sur un seul circuit. Pour compléter ce travail, ce type d'étude devrait être étendue aux différents types de circuits périphériques de microprocesseur. La continuation à long terme de cette étude devrait mener à l'intégration de systèmes personnalisables à l'aide d'un réseau de connexion souple (mer de transistors alias Prédiffusé à Interconnexions Libres) dans lequel sont immergés des blocs "durs" personnalisables selon les besoins de l'application (microprocesseur, mémoire, périphériques divers).

Annexes

Annexe 1

Table des valeurs des différentes modélisation du rendement (figure 1.3)

D*S	Poisson	Murphy	Seeds	Murphy/Seeds	Price
0,1	0,904837	0,905592	0,728893	0,817243	0,905478
0,2	0,818731	0,821463	0,639407	0,730435	0,821030
0,3	0,740818	0,746391	0,578265	0,662328	0,745463
0,4	0,670320	0,679305	0,531286	0,605296	0,677742
0,5	0,606531	0,619272	0,493069	0,556171	0,616961
0,6	0,548812	0,565475	0,460890	0,513182	0,562330
0,7	0,496585	0,517197	0,433155	0,475176	0,513158
0,8	0,449329	0,473810	0,408842	0,441326	0,468839
0,9	0,406570	0,434765	0,387251	0,411008	0,428841
1,0	0,367879	0,399576	0,367879	0,383728	0,392696
1,1	0,332871	0,367819	0,350355	0,359087	0,359991
1,2	0,301194	0,339118	0,334391	0,336754	0,330361
1,3	0,272532	0,313142	0,319763	0,316452	0,303487
1,4	0,246597	0,289600	0,306292	0,297946	0,279082
1,5	0,223130	0,268234	0,293833	0,281033	0,256894
1,6	0,201897	0,248816	0,282264	0,265540	0,236699
1,7	0,182684	0,231144	0,271487	0,251316	0,218299
1,8	0,165299	0,215039	0,261416	0,238228	0,201515
1,9	0,149569	0,200342	0,251980	0,226161	0,186190
2,0	0,135335	0,186911	0,243117	0,215014	0,172182
2,1	0,122456	0,174622	0,234773	0,204697	0,159366
2,2	0,110803	0,163362	0,226901	0,195132	0,147629
2,3	0,100259	0,153031	0,219462	0,186247	0,136869
2,4	0,090718	0,143541	0,212419	0,177980	0,126996
2,5	0,082085	0,134811	0,205741	0,170276	0,117929
2,6	0,074274	0,126771	0,199398	0,163084	0,109594
2,7	0,067206	0,119356	0,193367	0,156361	0,101926
2,8	0,060810	0,112510	0,187623	0,150067	0,094865
2,9	0,055023	0,106181	0,182147	0,144164	0,088357
3,0	0,049787	0,100323	0,176921	0,138622	0,082354
3,1	0,045049	0,094894	0,171928	0,133411	0,076813
3,2	0,040762	0,089857	0,167152	0,128504	0,071694
3,3	0,036883	0,085179	0,162579	0,123879	0,066962
3,4	0,033373	0,080828	0,158198	0,119513	0,062582
3,5	0,030197	0,076777	0,153996	0,115386	0,058528
3,6	0,027324	0,073001	0,149963	0,111482	0,054770
3,7	0,024724	0,069479	0,146089	0,107784	0,051286
3,8	0,022371	0,066188	0,142365	0,104277	0,048053
3,9	0,020242	0,063111	0,138783	0,100947	0,045051
4,0	0,018316	0,060232	0,135335	0,097783	0,042261
4,1	0,016573	0,057533	0,132014	0,094774	0,039667
4,2	0,014996	0,055002	0,128813	0,091908	0,037253
4,3	0,013569	0,052626	0,125727	0,089176	0,035006
4,4	0,012277	0,050392	0,122749	0,086570	0,032912
4,5	0,011109	0,048292	0,119873	0,084082	0,030960
4,6	0,010052	0,046314	0,117096	0,081705	0,029139
4,7	0,009095	0,044450	0,114412	0,079431	0,027440

4,8	0,008230	0,042691	0,111817	0,077254	0,025853
4,9	0,007447	0,041031	0,109307	0,075169	0,024370
5,0	0,006738	0,039463	0,106878	0,073170	0,022984
5,1	0,006097	0,037979	0,104526	0,071253	0,021686
5,2	0,005517	0,036575	0,102248	0,069412	0,020472
5,3	0,004992	0,035245	0,100041	0,067643	0,019335
5,4	0,004517	0,033984	0,097902	0,065943	0,018270
5,5	0,004087	0,032788	0,095827	0,064308	0,017271
5,6	0,003698	0,031652	0,093815	0,062734	0,016334
5,7	0,003346	0,030573	0,091862	0,061218	0,015455
5,8	0,003028	0,029547	0,089966	0,059757	0,014629
5,9	0,002739	0,028570	0,088126	0,058348	0,013853
6,0	0,002479	0,027640	0,086338	0,056989	0,013125
6,1	0,002243	0,026754	0,084600	0,055677	0,012439
6,2	0,002029	0,025909	0,082912	0,054410	0,011794
6,3	0,001836	0,025103	0,081270	0,053186	0,011187
6,4	0,001662	0,024333	0,079673	0,052003	0,010616
6,5	0,001503	0,023598	0,078120	0,050859	0,010077
6,6	0,001360	0,022894	0,076609	0.049751	0,009570
6,7	0,001231	0,022222	0,075137	0,048680	0,009092
6,8	0,001114	0,021578	0,073705	0,047642	0,008640
6,9	0,001008	0,020962	0,072311	0,046636	0,008215
7,0	0,000912	0,020371	0,070952	0,045661	0,007813
7,1	0,000825	0,019805	0,069628	0,044717	0,007433
7,2	0,000747	0,019261	0,068339	0,043800	0,007074
7,3	0,000676	0,018740	0,067081	0,042911	0,006735
7,4	0,000611	0,018239	0,065855	0,042047	0,006414
7,5	0,000553	0,017758	0,064660	0,041209	0,006111
7,6	0,000500	0,017296	0,063494	0,040395	0,005824
7,7	0,000453	0,016851	0,062356	0,039604	0,005552
7,8	0,000410	0,016423	0,061247	0,038835	0,005295
7,9	0,000371	0,016011	0,060163	0,038087	0,005051
8,0	0,000335	0,015615	0,059106	0,037360	0,004820
8,1	0,000304	0,015232	0,058073	0,036653	0,004601
8,2	0,000275	0,014864	0,057065	0,035964	0,004393
8,3	0,000249	0,014509	0,056080	0,035294	0,004196
8,4	0,000225	0,014166	0,055118	0,034642	0,004009
8,5	0,000203	0,013835	0,054178	0,034007	0,003831
8,6	0,000184	0,013516	0,053260	0,033388	0,003663
8,7	0,000167	0,013207	0,052362	0,032785	0,003503
8,8	0,000151	0,012909	0,051484	0,032197	0,003350
8,9	0,000136	0,012621	0,050626	0,031624	0,003206
9,0	0,000123	0,012343	0,049787	0,031065	0,003068
9,1	0,000112	0,012073	0,048966	0,030520	0,002937
9,2	0,000101	0,011812	0,048164	0,029988	0,002812
9,3	0,000091	0,011560	0,047378	0,029469	0,002694
9,4	0,000083	0,011315	0,046610	0,028963	0,002581
9,5	0,000075	0,011079	0,045858	0,028468	0,002473
9,6	0,000068	0,010849	0,045122	0,027986	0,002371
9,7	0,000061	0,010627	0,044401	0,027514	0,002273
9,8	0,000055	0,010411	0,043696	0,027054	0,002180
9,9	0,000050	0,010202	0,043006	0,026604	0,002092
10,0	0,000045	0,009999	0,042329	0,026164	0,002007

Table des valeurs de P (X=m ; S)

S (cm2)	m = 0	m = 1	m = 2	m = 3	m = 4	m = 5
0,001	0,99302792	0,00694425	2,7749E-05	8,3165E-08	2,077E-10	4,5649E-13
0,005	0,96568963	0,03363098	0,00066927	9,9892E-06	1,2424E-07	1,3599E-09
0,01	0,93271805	0,06464383	0,00256015	7,6044E-05	1,8823E-06	4,1E-08
0,02	0,87056018	0,11948865	0,00937166	0,00055127	2,7023E-05	1,1657E-06
0,03	0,81309151	0,16577594	0,0193137	0,0016876	0,00012288	7,8741E-06
0,04	0,75991781	0,20459326	0,03147589	0,00363183	0,00034921	2,9549E-05
0,05	0,71068133	0,23689378	0,04512262	0,00644609	0,00076739	8,0393E-05
0,06	0,66505711	0,2635132	0,05966337	0,01013151	0,0014337	0,00017854
0,07	0,62274974	0,28518446	0,07462771	0,01464656	0,00239547	0,00034477
0,08	0,5834904	0,30255058	0,08964461	0,01992103	0,00368908	0,00060118
0,1	0,51315812	0,32655517	0,11874733	0,03238564	0,00736037	0,00147207
0,15	0,37593704	0,34324686	0,17908532	0,07007686	0,02285115	0,00655729
0,2	0,27908165	0,32559526	0,2170635	0,10853175	0,04522156	0,01658124
0,3	0,15936632	0,2574379	0,23763498	0,16451652	0,09491338	0,04818679
0,5	0,05852766	0,13656455	0,18208606	0,18208606	0,15173839	0,11127482
0,7	0,02437011	0,07024327	0,11569479	0,1429171	0,14712054	0,1332739
1	0,0078125	0,02734375	0,0546875	0,08203125	0,10253906	0,11279297
1,5	0,0016384	0,00688128	0,01651507	0,02972713	0,04459069	0,05885972
2	0,00045725	0,00213382	0,00569019	0,01138038	0,0189673	0,0278187

Table des valeurs de P (X ≤ m ; S)

S (cm2)	m = 0	m = 1	m = 2	m = 3	m = 4	m = 5
0,001	0,99302792	0,99997217	0,99999992	1	1	1
0,005	0,96568963	0,99932061	0,99998989	0,99999987	1	1
0,01	0,93271805	0,99736188	0,99992203	0,99999808	0,99999996	1
0,02	0,87056018	0,99004883	0,99942049	0,99997176	0,99999879	0,99999995
0,03	0,81309151	0,97886745	0,99818115	0,99986876	0,99999164	0,99999952
0,04	0,75991781	0,96451107	0,99598696	0,99961879	0,999968	0,99999755
0,05	0,71068133	0,94757511	0,99269773	0,99914382	0,99991121	0,99999161
0,06	0,66505711	0,92857031	0,98823367	0,99836519	0,99979889	0,99997743
0,07	0,62274974	0,9079342	0,98256191	0,99720847	0,99960394	0,99994871
0,08	0,5834904	0,88604097	0,97568559	0,99560661	0,99929569	0,99989687
0,1	0,51315812	0,83971328	0,95846062	0,99084625	0,99820663	0,9996787
0,15	0,37593704	0,7191839	0,89826922	0,96834609	0,99119724	0,99775452
0,2	0,27908165	0,6046769	0,82174041	0,93027216	0,97549372	0,99207496
0,3	0,15936632	0,41680421	0,65443919	0,81895572	0,9138691	0,96205589
0,5	0,05852766	0,19509221	0,37717828	0,55926434	0,71100273	0,82227754
0,7	0,02437011	0,09461338	0,21030818	0,35322528	0,50034582	0,63361972
1	0,0078125	0,03515625	0,08984375	0,171875	0,27441406	0,38720703
1,5	0,0016384	0,00851968	0,02503475	0,05476188	0,09935258	0,15821229
2	0,00045725	0,00259107	0,00828126	0,01966164	0,03862894	0,06644764

Table des valeurs des courbes de la figure 2.22 :

Proportion de matériel rajouté pour une redondance modulaire

Nb de modules	Taille d'une tranche de module						
	5	10	15	20	30	50	80
5	92,5	66,25	57,5	53,125	48,75	45,25	43,28
10	72,5	46,25	37,5	33,125	28,75	25,25	23,28
15	65,83	39,58	30,83	26,46	22,08	18,58	16,61
20	62,5	36,25	27,5	23,125	18,75	15,25	13,28
25	60,5	34,25	25,5	21,125	16,75	13,25	11,28
30	59,17	32,917	24,17	19,7917	15,417	11,917	9,94
35	58,21	31,96	23,21	18,84	14,46	10,96	8,995
40	57,5	31,25	22,5	18,125	13,75	10,25	8,28
45	56,94	30,69	21,94	17,57	13,19	9,69	7,72
50	56,5	30,25	21,5	17,125	12,75	9,25	7,28
55	56,13	29,88	21,13	16,76	12,38	8,89	6,92
60	55,83	29,58	20,83	16,46	12,08	8,58	6,61
65	55,57	29,32	20,57	16,20	11,82	8,32	6,36
70	55,35	29,10	20,35	15,98	11,60	8,10	6,14
75	55,17	28,917	20,17	15,79	11,42	7,92	5,95

Tableau des valeurs des courbes de la figure 2.23 :

Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec une seule retenue propagée

Taille d'une tranche	Nombre de tranches de réserve				
	1	2	3	4	5
15	62,92	72,5	82,08	91,67	101,25
20	48,75	57,5	66,25	75	83,75
30	34,58	42,5	50,42	58,33	66,25
40	27,5	35	42,5	50	57,5
50	23,25	30,5	37,75	45	52,25
60	20,42	27,5	34,58	41,67	48,75
70	18,39	25,35	32,32	39,28	46,25
80	16,875	23,75	30,625	37,5	44,375
90	15,69	22,5	29,30	36,11	42,91
100	14,75	21,5	28,25	35	41,75
110	13,97	20,68	27,38	34,09	40,79
120	13,33	20	26,67	33,33	40
130	12,79	19,42	26,057	32,69	39,32
140	12,32	18,93	25,53	32,14	38,75
150	11,917	18,5	25,083	31,67	38,25
160	11,56	18,125	24,69	31,25	37,81
170	11,25	17,79	24,33	30,88	37,42
180	10,97	17,5	24,027	30,55	37,083
190	10,72	17,23	23,75	30,26	36,77
200	10,5	17	23,5	30	36,5

Tableau des valeurs des courbes de la figure 2.24 :

Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec deux retenue propagée

Taille d'une tranche	Nombre de tranches de réserve 1	2	3	4	5
15	105,42	117,5	129,58	141,67	153,75
20	80,625	91,25	101,9	112,5	123,125
30	55,83	65	74,17	83,33	92,5
40	43,44	51,875	60,31	68,75	77,19
50	36	44	52	60	68
60	31,04	38,75	46,46	54,17	61,875
70	27,5	35	42,5	50	57,5
80	24,84	32,19	39,53	46,875	54,22
90	22,78	30	37,22	44,44	51,67
100	21,125	28,25	35,375	42,5	49,625
110	19,77	26,82	33,86	40,91	47,95
120	18,64	25,625	32,60	39,58	46,5625
130	17,69	24,61	31,53	38,46	45,38
140	16,875	23,75	30,625	37,5	44,375
150	16,17	23	29,83	36,67	43,5
160	15,54	22,34	29,14	35,94	42,73
170	15	21,76	28,52	35,29	42,059
180	14,51	21,25	27,98	34,72	41,46
190	14,07	20,78	27,5	34,21	40,92
200	13,69	20,375	27,06	33,75	40,44

Annexe 2

Jeu d'Instructions

Introduction

Le jeu d'instructions du microprocesseur HYETI comporte les classes d'instructions suivantes :

- Instructions classiques (arithmétiques et logiques, branchement, transfert)
- Instructions avec masquage permettant de travailler sur des champs de bits
- Instructions de test.

Toutes les opérations à deux opérandes se font entre un des seize registres accessibles en dehors des instructions de test (ou une valeur immédiate pour COMP et STORE avec des éléments de matrice), et un opérande qui est :

* un registre
* un opérande immédiat
* un opérande en mémoire
* un élément de matrice.

1 - Modes d'adressage

La plupart des trente deux opérations arithmétiques et logiques peuvent être utilisées aves douze modes d'adressage (huit modes pour les éléments de matrices et quatre modes "classiques"). Les instructions de branchement travaillent uniquement en adressage relatif. Les instructions pour le test et COMPARMAT sont à part.

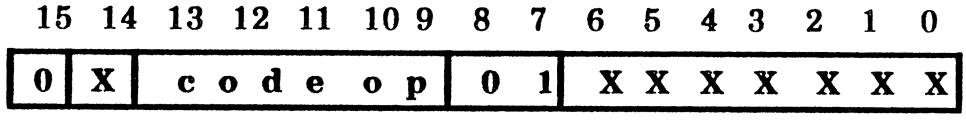
Remarques :

- Dans le cas des instructions avec masquage, un troisième opérande contenant le numéro du registre servant de masque (sur quatre bits est nécessaire. Il est placé dans le dernier mot de l'instruction.
- Les registres dont le numéro est donné sur trois bits appartiennent aux registres généraux R0 à R7.
- Les registres dont le numéro est donné sur quatre bits (registres destination, registre de masque, registres numéro de ligne ou de colonne), sont parmi les seize registres accessibles à l'utilisateur en dehors des instruction de test : CO, RE, PP, RM0, RM1, RP1, POL, RTIS, ainsi que R0 à R7.

1.1 - Modes d'adressage "classiques"

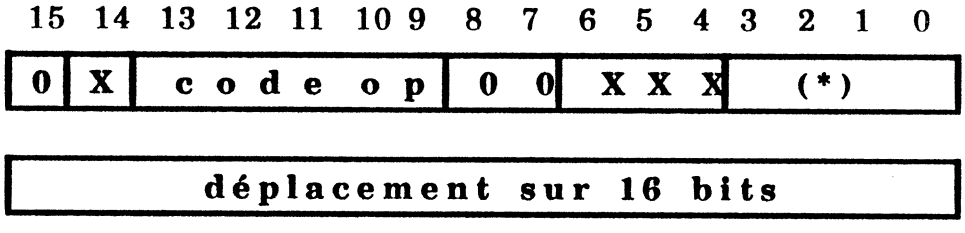
** INHERENT:*

L'opérande est implicitement désigné par le code opération. Ce mode d'adressage n'est utilisé que par RSR, CLC, SEC.



** RELATIF :*

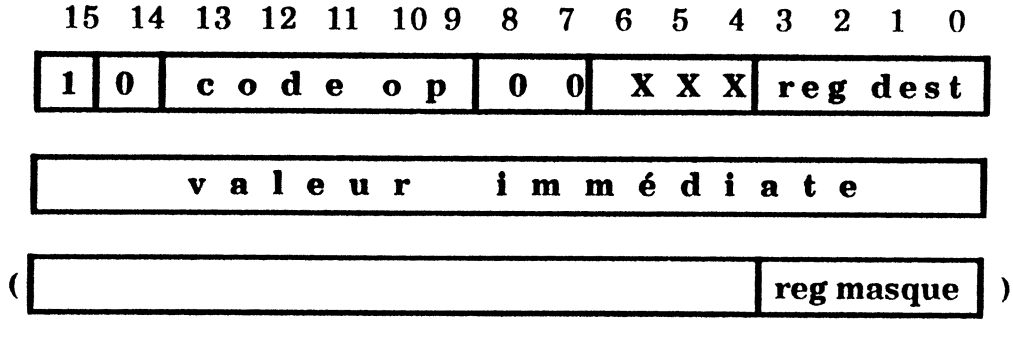
L'opérande est un déplacement sur seize bits codé en complément à deux. Il est placé dans le mot mémoire suivant celui du code de l'instruction. Ce mode d'adressage n'est utilisé que par les instructions de branchement.



(*) : Numéro du bit testé dans le registre d'état (il y en a dix possibles).

** OPERANDE IMMEDIAT :*

Ce mode et les suivants sont utilisés dans pratiquement toutes les instructions arithmétiques, logiques et de transfert. L'opérande est ici une valeur située dans le deuxième mot de l'instruction.

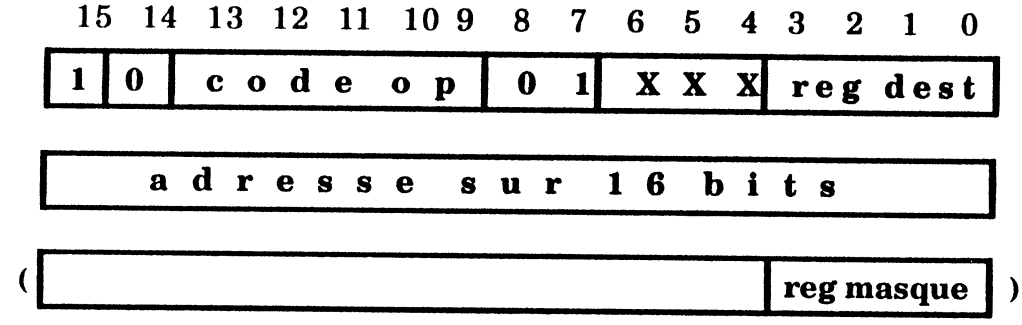


reg dest est le numéro du registre destination de l'opération.

reg masque est le numéro du registre contenant le masque

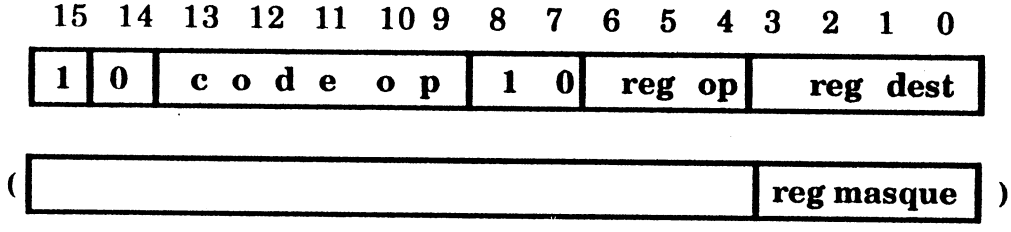
* ***DIRECT :***

Le mot suivant le code de l'instruction contient l'adresse de l'opérande sur seize bits.



* ***REGISTRE :***

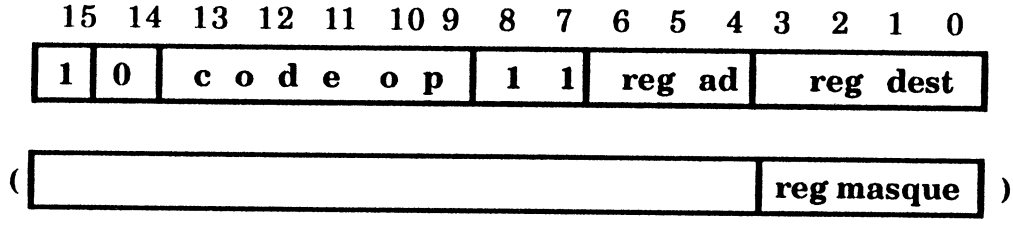
L'opérande est ici dans un registre dont le numéro est donné dans les bits 6 à 4 du mot contenant le code de l'instruction. Ce registre est un des huit registres généraux.



reg op est le numéro du registre opérande

* ***INDIRECT PAR REGISTRE :***

L'adresse de l'opérande est dans le registre dont le numéro est donné dans le champ **reg ad** de l'instruction.



1.2 - Modes d'adressage des éléments de matrice

Ces modes d'adressage permettent d'accéder individuellement aux éléments (mots de seize bits) de tableaux bidimensionnels occupant au plus une page mémoire. Il en existe huit parmi lesquels quatre ne sont utilisés que par deux instructions (STORE et COMP).

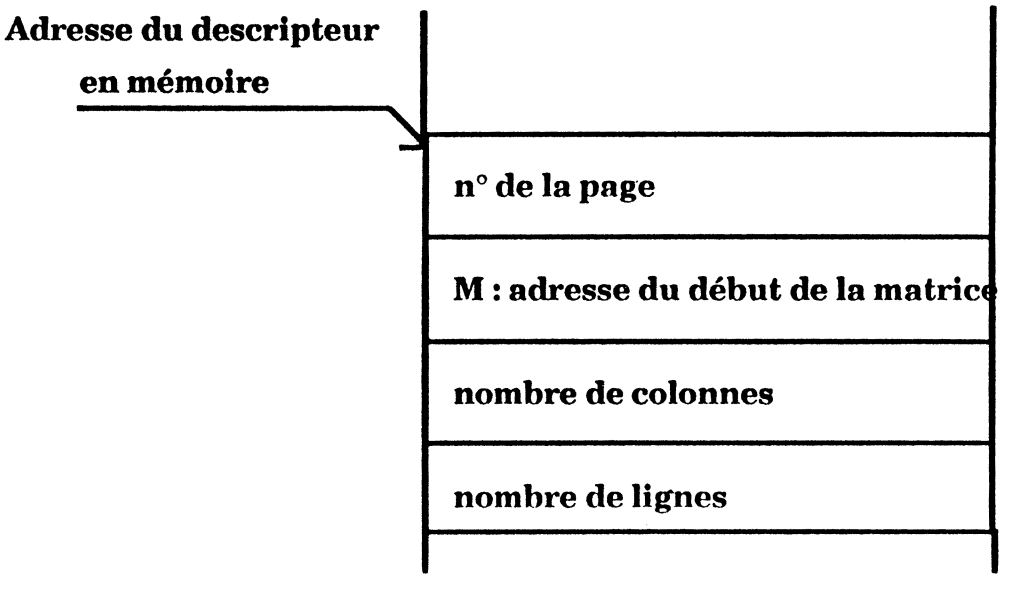
1.2.1 - Organisation d'une matrice en mémoire

Généralement, une matrice n'est pas située dans la même page mémoire que le programme. Cela implique d'avoir parmi toutes les variables du programme les informations nécessaires à l'accès aux éléments de matrices. Les caractéristiques statiques des matrices sont rangées en mémoire sous forme d'un enregistrement situé dans la même page mémoire que le programme et ses données.

Cet enregistrement contient quatre mots successifs :

* Deux mots définissant l'adresse du premier élément de la matrice :
  - Un mot contenant le numéro de la page où se trouve la matrice.
    Une matrice est entièrement incluse dans une page et ne peut être "à cheval" sur deux pages.
  - Un mot contenant l'adresse du début de la matrice dans cette page.

* Deux mots indicant la taille de la matrice :
  - Un mot contenant le nombre de colonnes.
  - Un mot contenant le nombre de lignes.



Les éléments d'une matrice bidimensionnelle sont rangés par lignes dans la mémoire, par exemple la matrice :

M	*0*	*1*	*2*	*3*
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l

sera représentée en mémoire de la façon suivante :

M		n° de l'élément
M + 0	a	*0.0*
1	b	*0.1*
2	c	*0.2*
3	d	*0.3*
4	e	*1.0*
5	f	*1.1*
6	g	*1.2*
7	h	*1.3*
8	i	*2.0*
9	j	*2.1*
10	k	*2.2*
11	l	*2.3*

Pour accéder à un élément d'une matrice, il faut spécifier l'adresse du descripteur (sur seize bits) , ainsi que le numéro de la ligne et le numéro de la colonne de l'élément considéré. L'adresse du descripteur peut être spécifiée de deux façons :

* soit dans un mot mémoire,
* soit dans deux registres spécialisés, RM0 et RM1. Deux registres sont nécessaires afin de pouvoir travailler simultanément sur deux matrices (comparaison).

L'adresse d'un élément de matrice est alors :

adr = M + (n° ligne * nombre de colonnes) + n° de colonne

On a besoin pour un calcul rapide de l'adresse, d'un opérateur spécialisé de multiplication effectuant l'opération a * b + c + d.

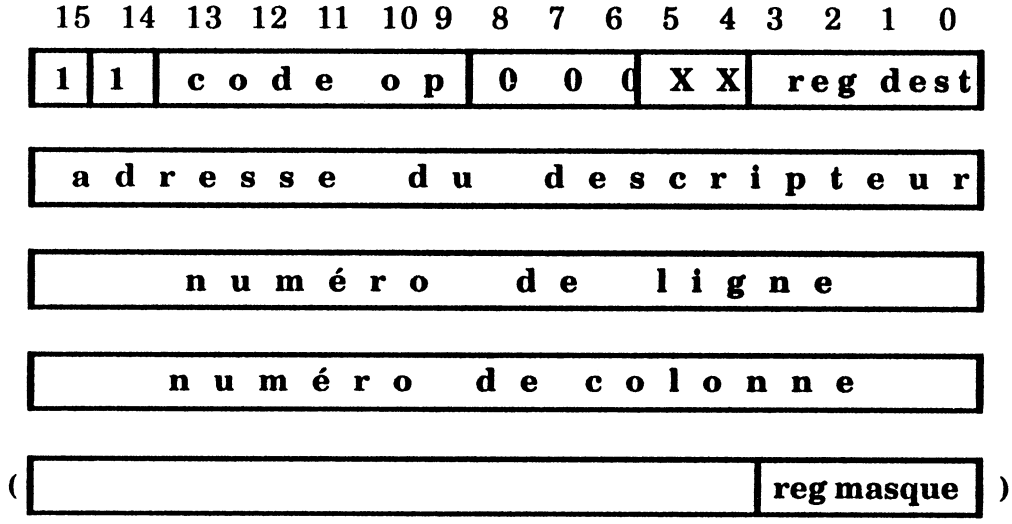
1.2.2 - Modes d'adressage

Les différents modes d'adressage correspondent aux multiples façons de spécifier les paramètres d'accès à un élément de matrice : adresse du descripteur et numéros de ligne et de colonne de l'élément concerné. Toutes les combinaisons possibles sont résumées dans le tableau ci-dessous. Les bits 8 à 6 du code de l'instruction indiquent le mode d'adressage.

Descripteur	***Numéros Ligne - Colonne***	***Mnémonique***
Direct	**Valeur immédiate**	]D]I
Registre	**Valeur immédiate**	]R]I
Direct	**Registre**	]D]R
Registre	**Registre**	]R]R

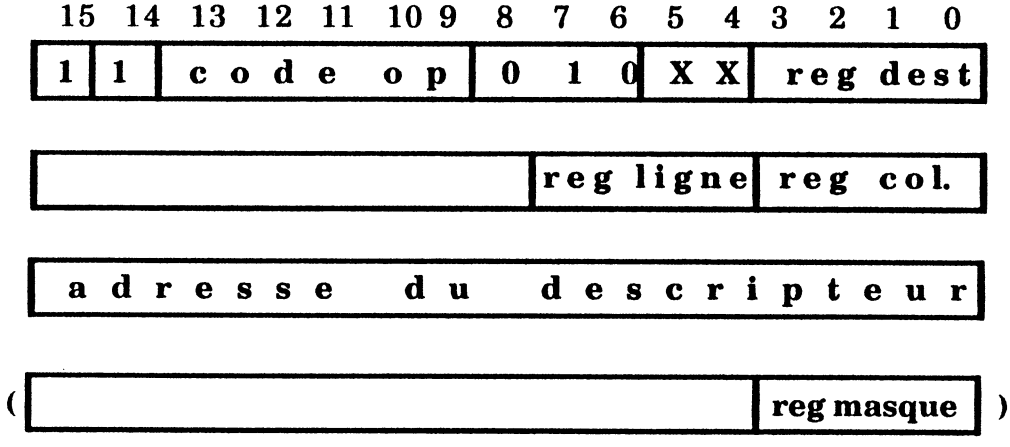
* ***DIRECT, VALEUR IMMEDIATE :***

Les trois paramètres sont donnés dans les mots suivant le code de l'instruction.



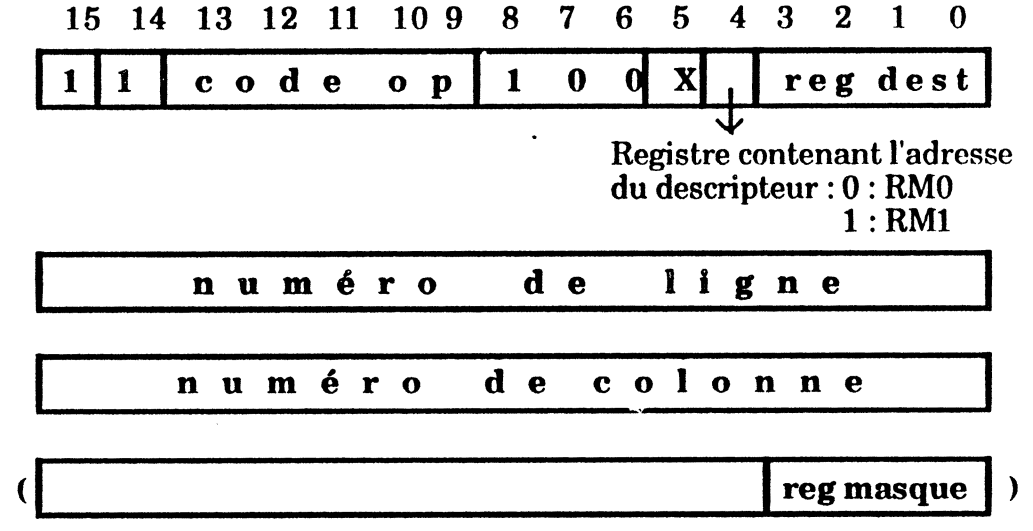
* ***DIRECT, REGISTRE :***

Les numéros de ligne et de colonne de l'élément adressé sont dans deux registres dont les numéros sont donnés dans le mot suivant le code de l'instruction. L'adresse du descripteur est donnée dans le mot suivant.



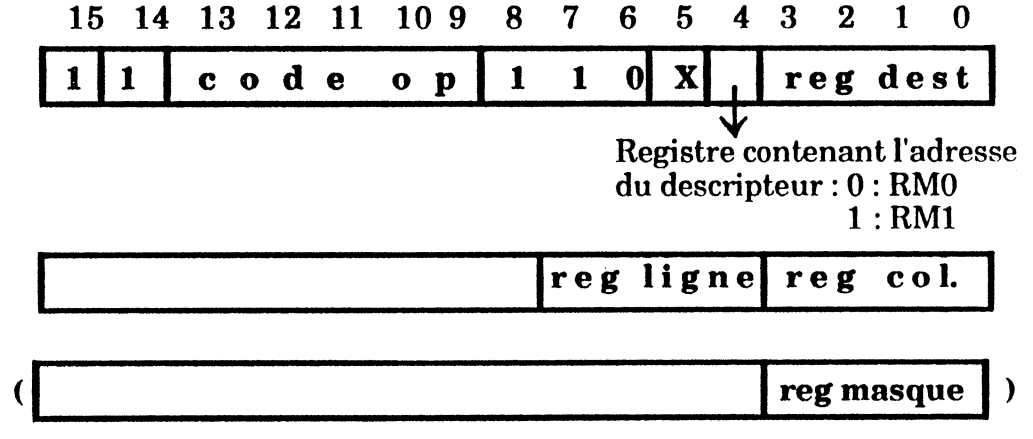
* ***REGISTRE VALEUR IMMEDIATE***

L'adresse du descripteur est dans un des deux registres matrice RM0 ou RM1. Les numéros de ligne et de colonne de l'élément visé sont dans les deux mots suivant le code de l'instruction.



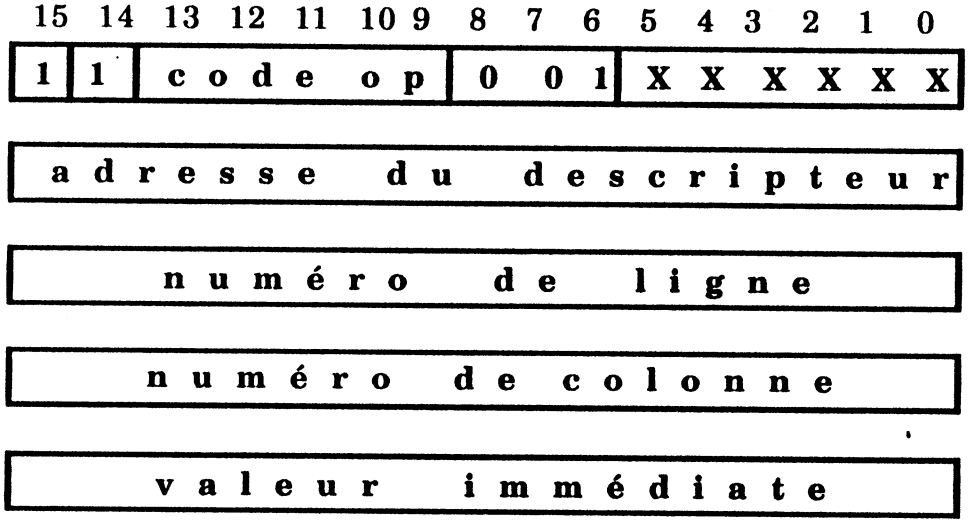
* ***REGISTRE, REGISTRE :***

L'adresse du descripteur est dans RM0 ou RM1. Les numéros de ligne et de colonne sont dans des registres dont les numéros sont dans donnés dans le mot suivant le code de l'instruction.

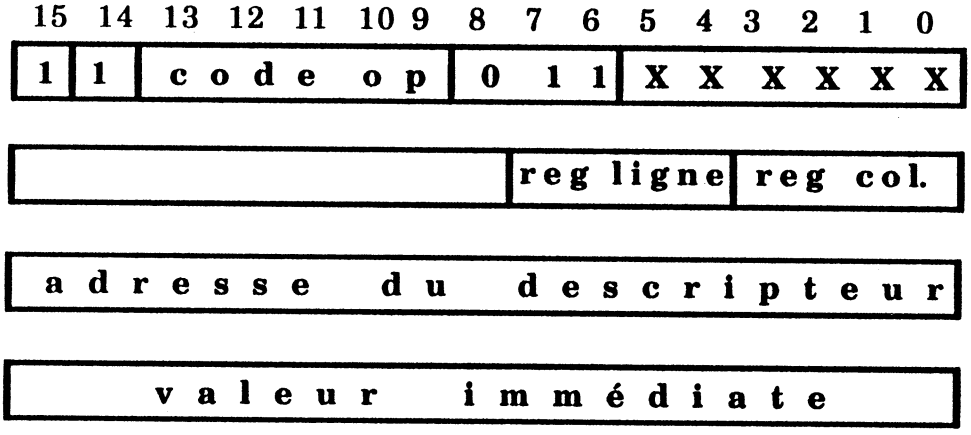


Les quatre modes d'adressage suivants ne sont utilisés que par les instructions STORE et COMP. Ils permettent d'effectuer ces opérations entre un élément de matrice et une valeur immédiate. Ils sont similaires à ceux ci-dessus avec en plus un mot pour la valeur immédiate. Ces modes d'adressage ne sont pas utilisables avec des opérations masquées.

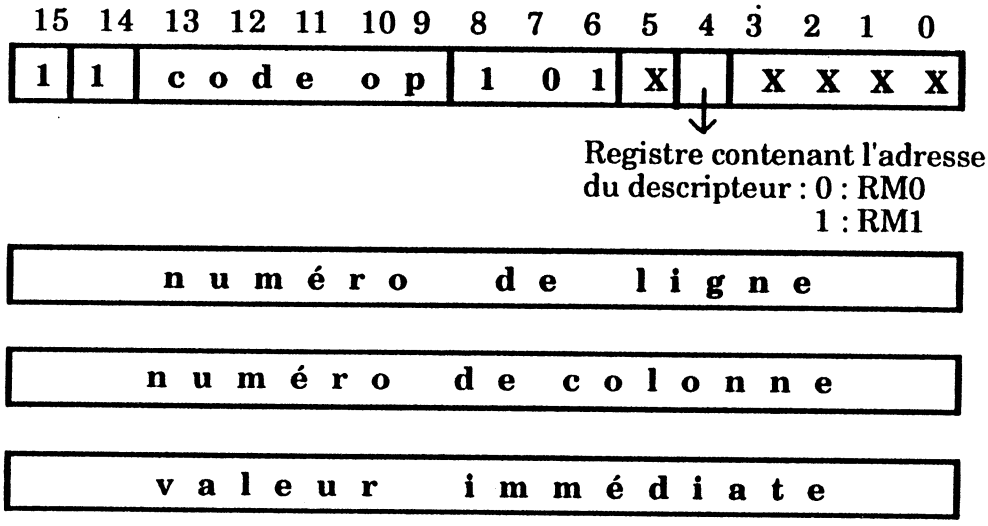
* ***DIRECT, VALEUR IMMEDIATE - VALEUR IMMEDIATE :***



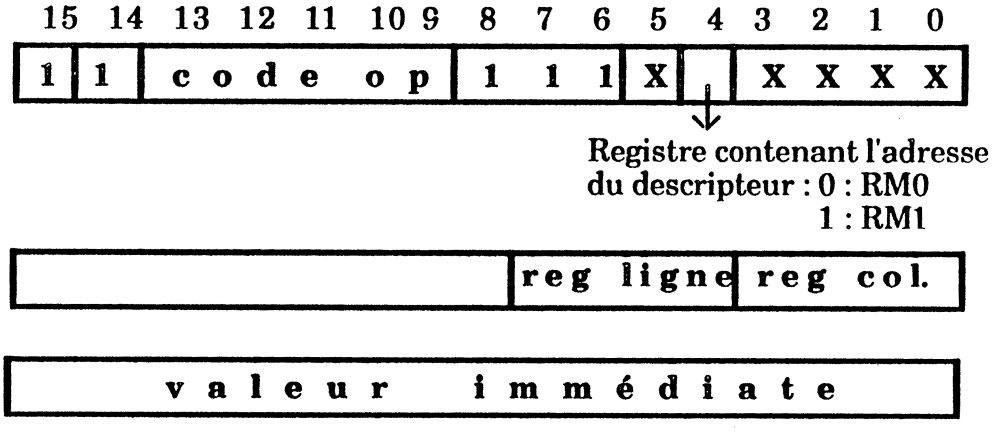
* ***DIRECT, REGISTRE - VALEUR IMMEDIATE :***



* ***REGISTRE, VALEUR IMMEDIATE - VALEUR IMMEDIATE :***



* ***REGISTRE, REGISTRE - VALEUR IMMEDIATE :***



2 - Instructions

On distingue cinq types d'instructions :

Arithmétiques et Logiques :

Addition, soustraction, opérations logiques de base (et, ou, nand, nor, eor), décalages, opérations de transfert (load, store, exchange), opérations masquées et opérations de comparaison.

Branchement :

Il en existe vingt et une, deux par bit du registre d'état (branchement si zéro et branchement si un), plus le branchement inconditionnel (BRA).

Instructions avec opérande inhérent :

CLC, SEC qui agissent sur le bit de la retenue dans le registre d'état.
RSR, retour de sous-programme.

Rupture de séquence et manipulation de pile :

Saut inconditionnel, saut à un sous-programme, empiler, dépiler.

Instruction de comparaison de deux matrices : COMPARMAT

Instructions pour le test :

Elles permettent l'écriture et la lecture de **TOUS** les éléments de mémorisation du microprocesseur ainsi que la sélection des informations à signer.

Notation :

Les registres **destination** sont notés **R.**

Les **opérandes** sont notés **Op.**

L'affectation du registre destination est notée ":="

Exemple :

Une opération à deux opérandes (reg. dest. reçoit reg. dest. oper. opérande) sera notée :

R := R oper Op

Un **X** dans un bit du registre d'état signifie que celui-ci est modifié par l'instruction, un **?** indique que la valeur du bit est indéfinie.
Un **X** dans un bit du code de l'instruction indique que ce bit n'est pas utilisé, et donc que sa valeur est indifférente.

Attention :

Quand le registre destination est RE, on ne prend pas en compte les bascules venant de l'UAL. La signification des bits de RE est donnée au chapitre cinq.

2.1 - Instructions arithmétiques, logiques et de transfert

2.1.1 - Instructions utilisant les modes d'adressage *Opérande immédiat, Direct, Registre, Indirect par registre,* et pour les éléments de matrice *DI, DR, RI, RR*

2.1.1.1 - Instructions ARITHMETIQUES

ADD: addition sans retenue

R := R + Op

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
			X	X	X	X	X	X	X

ADDC : addition avec retenue

R := R + Op + C

Registre d'état : le même que pour ADD

SUB : soustraction sans retenue

R := R - Op

Registre d'état : le même que pour ADD

SUBC : soustraction avec retenue

R := R -Op - C

Registre d'état : le même que pour ADD

2.1.1.2 - Instructions LOGIQUES

AND : et logique

R := R et Op

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
			?	?	?	?	X	X	?

OR : ou logique

R := R ou Op

Registre d'état : le même que pour AND

NAND: non_et logique

R := $\overline{R \text{ et } Op}$

Registre d'état : le même que pour AND

NOR: non_ou logique

R := $\overline{R \text{ ou } Op}$

Registre d'état : le même que pour AND

EOR : ou exclusif

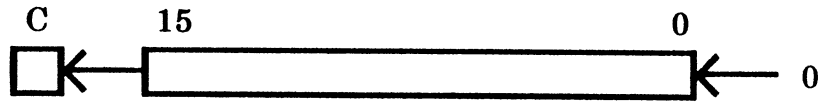
R := R $\oplus$ Op

Registre d'état : le même que pour AND

2.1.1.3 - Instructions de DECALAGE

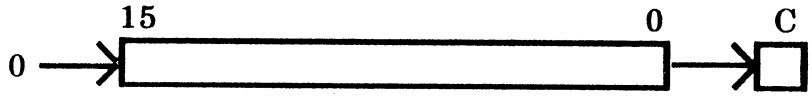
On dispose d'instructions de décalage arithmétique ou logique permettant de décaler un registre de plusieurs positions. Le nombre de positions dont on décale est contenu dans l'opérande Op.

LSL : décalage logique d'un registre (R) à gauche



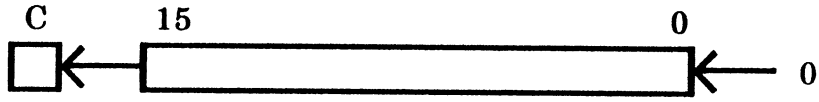
Registre d'état : C est modifié

LSR : décalage logique d'un registre (R) à droite



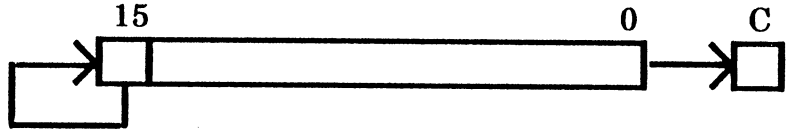
Registre d'état : C est modifié

ASL : décalage arithmétique d'un registre (R) à gauche



Registre d'état : C est modifié

ASR : décalage arithmétique d'un registre (R) à droite



Registre d'état : C est modifié

2.1.1.4 - Instruction de COMPARAISON

COMP: Comparaison entre un registre (R) ou d'une valeur immédiate (pour les éléments de matrice uniquement) et un registre ou un mot mémoire (variable simple, élément de matrice) ou une valeur immédiate.
On fait pour cela R - Op, mais ni R ni Op n'est modifié

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
						X	X	X	X

Remarque : Cette instruction peut utiliser les modes d'adressage suivants en plus de ceux cités plus haut :

Registre, Valeur immédiate - Valeur immédiate

Direct, Registre - Valeur immédiate

Registre, Valeur immédiate - Valeur immédiate

Registre, Registre - Valeur immédiate

2.1.1.5 - Instructions de TRANSFERT

LOAD : chargement d'un registre

R := Op

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
								X	X

STORE: rangement d'un registre ou d'une valeur immédiate (pour les éléments de matrice uniquement) en mémoire (opérande simple ou élément de matrice)

Op := R (ou VALEUR IMMEDIATE)

Registre d'état : le même que pour LOAD

Remarque : MODES D'ADRESSAGE

* Comme COMP, cette instruction peut également utiliser les modes d'adressage suivants pour le rangement d'une valeur immédiate dans un élément de matrice :

  Registre, Valeur immédiate - Valeur immédiate

  Direct, Registre - Valeur immédiate

  Registre, Valeur immédiate - Valeur immédiate

  Registre, Registre - Valeur immédiate

* Le mode d'adressage **Opérande immédiat** n'est pas utilisable avec cette instruction. Son emploi est détecté comme un code d'instruction invalide

2.1.1.6 - Instructions avec MASQUAGE

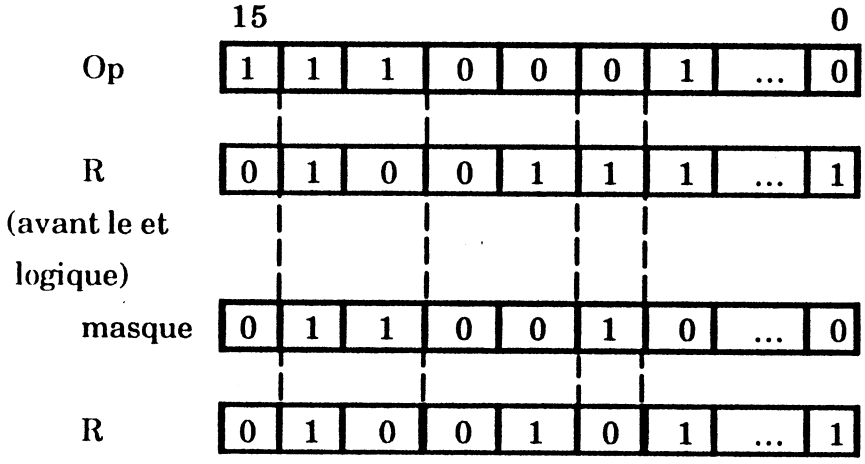
Les instructions avec masquage permettent de travailler directement sur des bits d'un mot ou sur des parties de mot. Le masque est contenu dans un des seize registres accessibles à l'utilisateur en dehors des instructions de test. Le numéro de ce registre est donné dans le dernier mot (supplémentaire) de l'instruction.

Le jeu d'instructions du microprocesseur comprend huit instructions avec masquage : cinq instructions logiques, une instruction de comparaison, deux instructions de transfert.

*i) **Instructions LOGIQUES avec masquage** :*

L'opération effectuée est la suivante : **(R := R oper Op) masqué**

Par exemple, dans le cas où **oper** est le **et logique** :



Les cinq instructions logiques avec masquage sont :

ANDM : et logique avec masquage
(R := R et Op) masqué
Registre d'état :

E	L	H	W	T	G	V	N	Z	C
X	X	?	?	?	?	X	X	?	

ORM : ou logique avec masquage

(R := R ou Op) masqué

Registre d'état : le même que pour ANDM

NANDM: non_et logique avec masquage

(R := $\overline{\text{R et Op}}$) masqué

Registre d'état : le même que pour ANDM

NORM: non_ou logique avec masquage

(R := $\overline{\text{R ou Op}}$) masqué

Registre d'état : le même que pour ANDM

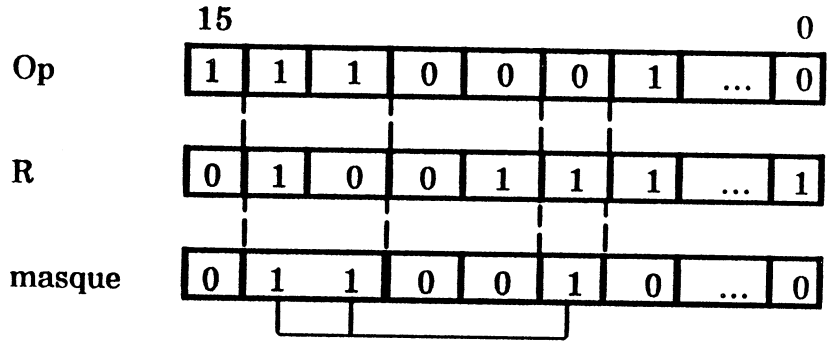
EORM: ou exclusif avec masquage

(R := R ⊕ Op) masqué

Registre d'état : le même que pour ANDM

ii) Instruction de COMPARAISON :

COMPM: comparaison avec masquage



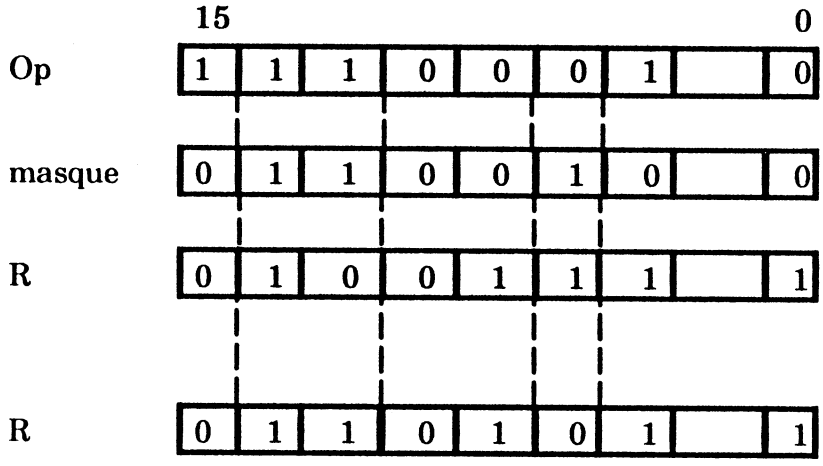
Registre d'état : Si tous les bits comparés sont **égaux**, **Z=1**, sinon Z=0.

La valeur du bit N est indéterminée.

iii) Instructions de TRANSFERT :

LOADM: chargement d'un registre avec masquage

(R := Op) masqué



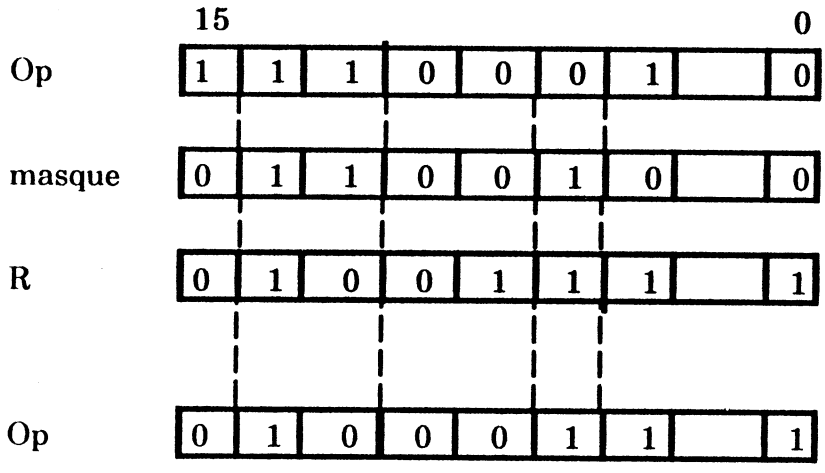
(après LOADM)

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
	X	X					X	X	

STOREM: rangement d'un registre avec masquage

(Op := R) masqué



(après STOREM)

Registre d'état : le même que pour LOADM

Remarque : Comme on l'a vu pour l'instruction STORE, le mode d'adressage **Opérande immédiat** n'est pas utilisable avec STOREM.

2.1.2 - Instructions n'utlisant que le mode d'adressage REGISTRE

INC : incrémentation d'un registre

R := R + 1

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
						X	X	X	X

Remarque : le champ opérande du code de l'instruction n'est pas utilisé.

DEC : décrémentation d'un registre

R := R - 1

Registre d'état : le même que pour INC

Remarque : la même que pour INC

NOT : complément logique d'un registre

$R := \bar{R}$

Registre d'état :

E	L	H	W	T	G	V	N	Z	C
						?	X	X	?

EXCH : échange entre deux registres

R := =: Op

Registre d'état :

Le registre d'état est modifié en fonction du contenu de **Op**. Ceci n'est pas vrai si **R** est le registre **RE**. Si **Op** est le registre d'état **RE**, son contenu sera celui de **R, MAIS** les bits correspondant à **N** et **Z** seront modifiés lors du transfert de **Op** (ancienne valeur) vers **R**.

2.2 - Instructions de branchement

L'opérande **Op** est dans ce cas un déplacement codé en complément à deux. Aucune de ces instructions ne modifie le registre d'état.

BRA : branch always
branchement inconditionnel
CO := CO + Op

BCC : branch if carry clear
branchement si la retenue est à zéro
CO := CO + Op si ***C = 0*** ; CO := CO + 1 si C = 1

BCS : branch if carry set
branchement si la retenue est à un
CO := CO + Op si ***C = 1*** ; CO := CO + 1 si C = 0

BEQ : branch if equal
branchement si égal
CO := CO + Op si ***Z = 1*** ; CO := CO + 1 si Z = 0

BNE : branch if not equal
branchement si différent
CO := CO + Op si ***Z = 0*** ; CO := CO + 1 si Z = 1

BVC : branch if overflow clear
branchement si pas de débordement
CO := CO + Op si ***V= 0*** ; CO := CO + 1 si V = 1

BVS : branch if overflow set
branchement si débordement
CO := CO + Op si ***V= 1***; CO := CO + 1 si V = 0

BMI : branch if minus
branchement si moins
CO := CO + Op si ***N= 1***; CO := CO + 1 si N = 0

BPL : branch if plus

branchement si plus

CO := CO + Op si $N = 0$; CO := CO + 1 si N = 1

BHI : branch if higher

branchement si supérieur

CO := CO + Op si $C + Z = 0$

CO := CO + 1 si C + Z = 1

BLS : branch if less

branchement si inférieur

CO := CO + Op si $C + Z = 1$

CO := CO + 1 si C + Z = 0

BGE : branch if greater or equal

branchement si positif ou nul

CO := CO + Op si $N \oplus V = 0$

CO := CO + 1 si $N \oplus V = 1$

BLT : branch if lower

branchement si négatif

CO := CO + Op si $N \oplus V = 1$

CO := CO + 1 si $N \oplus V = 0$

BGT : branch if greater

branchement si positif

CO := CO + Op si $Z + (N \oplus V) = 0$

CO := CO + 1 si $Z + (N \oplus V) = 1$

BLE : branch if lower or equal

branchement si négatif ou nul

CO := CO + Op si $Z + (N \oplus V) = 1$

CO := CO + 1 si $Z + (N \oplus V) = 0$

BLH : branch if L high
branchement si L = 1
CO := CO + Op si ***L= 1*** ; CO := CO + 1 si L = 0

BLL : branch if L low
branchement si L = 0
CO := CO + Op si ***L= 0*** ; CO := CO + 1 si L = 1

BHH : branch if H high
branchement si H = 1
CO := CO + Op si ***H= 1*** ; CO := CO + 1 si H = 0

BHL : branch if H low
branchement si H = 0
CO := CO + Op si ***H= 0*** ; CO := CO + 1 si H = 1

BEH : branch if E high
branchement si E = 1
CO := CO + Op si ***E= 1*** ; CO := CO + 1 si E = 0

BEL : branch if E low
branchement si E = 0
CO := CO + Op si ***E= 0*** ; CO := CO + 1 si E = 1

2.3 - Instructions avec opérande inhérent

CLC : clear carry
mise à zéro de la retenue
C := 0
Registre d'état :

E	L	H	W	T	G	V	N	Z	C
									X

SEC : set carry
mise à un de la retenue
C := 1
Registre d'état : le même que pour CLC

NOP : non opération
passage en séquence à l'instruction suivante

2.4 - Instructions de rupture de séquence et de manipulation de pile

Ces instructions ne modifient pas le registre d'état.

* ***Instructions utilisant les modes d'adressage Opérande immédiat, Direct, Registre, Indirect par registre,*** et pour les éléments de matrice ***DI, DR, RI, RR :***

JMP : saut inconditionnel
CO := Op

JSR : saut à un sous-programme

PP := PP - 1	)	sauvegarde en pile de
MEM (PP) := CO	)	l'ancienne valeur de CO
CO := Op	)	chargement de CO avec l'adresse du sous–programme

PUSH : empiler
PP := PP - 1
MEM (PP) := Op

* ***Instruction sur Registre uniquement :***

PULL : dépiler
Le sommet de la pile est mis dans le registre dont le numéro est donné dans le champ **reg dest** du code de l'instruction.
R := MEM (PP)
PP := PP + 1

**** Instruction avec opérande Inhérent :***

RSR : retour de sous-programme
CO := MEM (PP)) restauration de CO
PP := PP+ 1)

2.5 - Instruction de comparaison de deux matrices : COMPARMAT

Cette instruction compare séquentiellement deux matrices (de même nature, c.a.d ayant le même nombre de lignes et le même nombre de colonnes), ou une matrice et un registre (dont le numéro est donné dans les bits 7 à 4 du mot suivant le code de l'instruction) à partir de l'élément (i, j).

La comparaison est masquée, c'est à dire que les seuls bits des éléments de matrice à être comparés sont ceux dont le rang correspond aux bits à un du masque. Celui-ci est contenu dans un registre dont le numéro est donné dans les bits 3 à 0 du mot suivant le code de l'instruction. La comparaison est faite par lignes en suivant le rangement des matrices en mémoire. Le résultat de l'opération est formé de deux choses :

* Les numéros de ligne et de colonne du premier élément différent (◆ comparaison entre deux matrices) ou le premier élément dont le contenu est le même que celui du registre (◆◆ comparaison entre une matrice et un registre). Dans le cas où un tel élément n'existe pas, le résultat est formé des coordonnées du dernier élément de la matrice.

* Le positionnement du bit **E** du registre d'état :

E := 1 si : ◆ les matrices diffèrent après l'élément (i, j).
◆◆ la valeur contenue dans le registre a été trouvée après l'élément (i, j).

E := 0 si : ◆ les matrices sont égales après l'élément (i, j).
◆◆ le contenu du registre n'a pas été trouvé dans la matrice après l'élément (i, j).

Structure de l'instruction :

COMPARMAT ADR descr1, (ADR descr2, reg ligne, reg col., reg masque
(reg Op

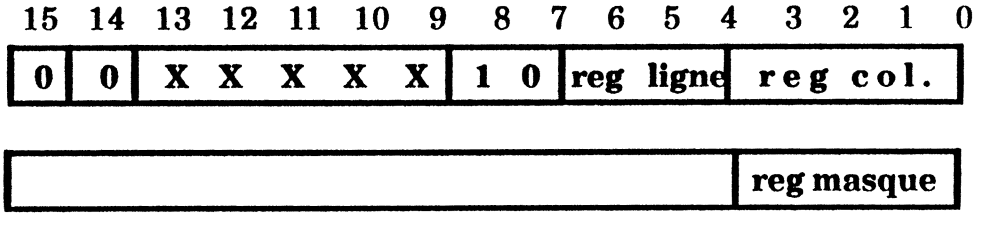
ADR descr1 et **ADR descr2** sont les adresses des descripteurs des matrices (dans le cas d'une comparaison entre deux matrices). Elles sont situées dans **RM0** pour **ADR descr1** et ***RM1*** pour ***ADR descr2***.

Reg Op est le numéro du registre auquel on compare la matrice. Dans ce cas, l'adresse du descripteur est contenue dans RM0.

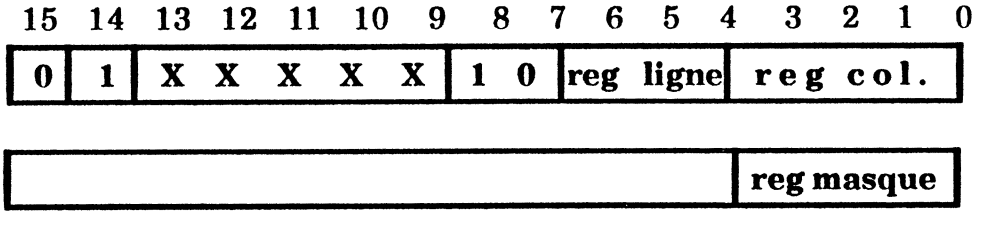
Reg ligne et **reg col.** sont les numéros des registres contenant les numéros de ligne et de colonne (**i** et **j**) du premier élément à comparer. Le numéro de colonne étant celui de l'élément précédent dans la ligne, c'est à dire -1 pour un élément de la colonne 0.Le résultat (numéros de ligne et de colonne du premier élément différent (comparaison de deux matrices)) sera situé dans ces registres. Il est à noter qu'un redémarrage de l'instruction sans réinitialisation de ces registres permet de continuer la comparaison en séquence.

Format de l'instruction :
Les adresses des descripteurs doivent déjà être dans RM0 et RM1.

*** Comparaison de deux matrices**



*** Comparaison entre une matrice et un registre**



Exemple :

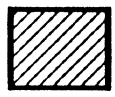
COMPARMAT adr1 adr2 1 1

adr1

	0	1	2
0	X	Z	Y
1	B	O	Y
2	Z	V	K
3	Z	A	N

adr2

	0	1	2
0	C	J	A
1	P	O	Y
2	Z	V	K
3	Z	U	G

 = éléments testés

Premier élément différent : MAT (3, 1)

Résultat :

* Bit E du registre d'état à un.
* Le registre ligne contient 3.
* Le registre colonne contient 1.

2.6 - Instructions de test

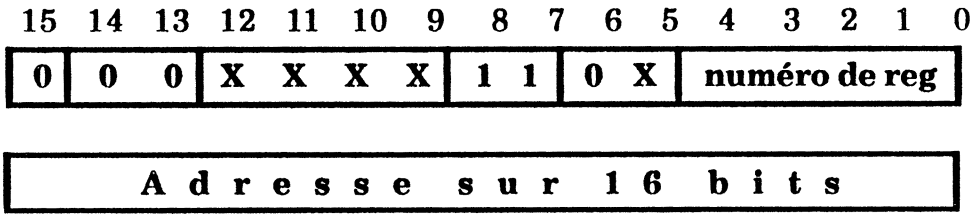
Le rôle de ces instructions est de permettre à l'utilisateur d'accéder à tous les registres du microprocesseur ainsi que d'offrir toutes facilités pour le test en contrôlant les mécanismes de signature et les opérateurs facilement testables. Après l'exécution d'une de ces instructions, le contenu du registre d'état est indéfini. Il y a cinq instructions de test :

LRS : chargement d'un registre interne avec un opérande direct

R := Op

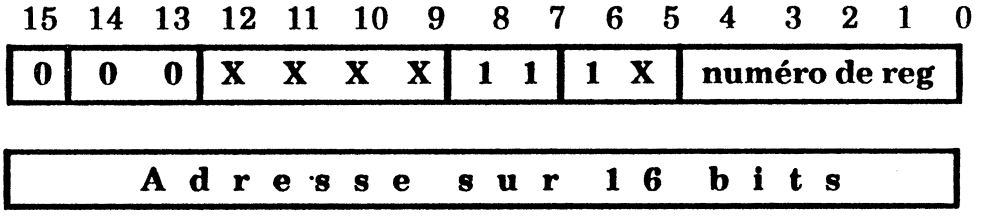
On entend par registres internes ceux qui ne sont pas accessibles avec les instructions normales LOAD et STORE (c.a.d ceux qui ne sont accessibles que pour le test) : T1, T2, E1, E2, RP2, S, TM, A, B, C, D, RMA, SIGN; CONT, RESM,

RSMA, et différentes bascules d'indicateurs d'état internes (N, Z, L, CIN, RESET, ALARME) qui sont vus comme un registre pour ces instructions. Le bit 4 du code de l'instruction à un indique que l'on travaille sur un registre de signature. Dans ce cas, l'instruction n'est pas signée.



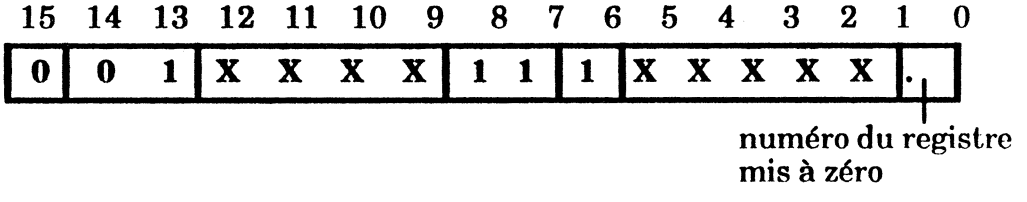
SRS : sortie d'un registre interne dans un mot mémoire (opérande direct)

Op := R



RAZ : remise à zéro d'un registre de signature

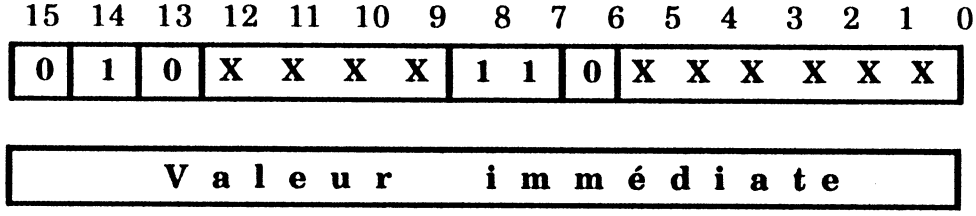
R := 0



AJS : ajustement de signature

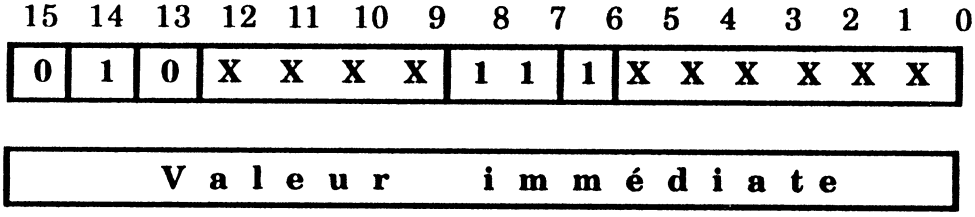
SIGN := SIGN + Op

Cette instruction ajuste le contenu du registre de signature de la partie opérative SIGN pour permettre à l'utilisateur de comparer les signatures de différents chemins d'exécution du programme (arrivant tous au même point) à une seule valeur.



AJST: ajustement de signature et test

Cette instruction est similaire à AJS, mais l'opération d'ajustement est suivie d'un test du contenu de SIGN avec la valeur zéro. Dans le cas où le contenu de SIGN n'est pas nul, le signal ALARME est mis à un et le microprocesseur attend un signal de redémarrage sur l'entrée RESET.



3 - Affectation des numéros de registres et des codes opérations

3.1 - Pour les instructions LRS et SRS

Numéro	Registre	Code binaire 4 0
1	T1	0 0 0 0 1
2	T2	0 0 0 1 0
3	E1	0 0 0 1 1
4	E2	0 0 1 0 0
5	RP2	0 0 1 0 1
6	S	0 0 1 1 0
7	TM	0 0 1 1 1
8	A	0 1 0 0 0
9	B	0 1 0 0 1
10	C	0 1 0 1 0
11	D	0 1 0 1 1
12	Flags : N, Z, L, ALARME, RESET. CIN	0 1 1 0 0
13	RMA	0 1 1 0 1
14	Reg de sign du n° d'état, RSMA	0 1 1 1 0
15	SIGN	0 1 1 1 1
16	CONT	1 0 0 0 0
17	RESM	1 0 0 0 1

3.2 - Pour l'instruction RAZ

Le numéro de registre est situé dans le bit 0 du code de l'instruction :

0 : SIGN,

1 : Registre de signature des numéros d'état : RSMA

3.3 - Pour les autres instructions

Numéro	Registre
0	R0
à 7	R7 Registres généraux
8	CO
9	PP
10	RM0
11	RM1
12	RP1
13	RE
14	POL
15	RTIS

Remarque : Un registre dont le numéro est codé sur trois bits, est un des huit registres généraux.

3.4 - Codes opération (bits 13 à 9 du code l'instruction)

Code	Opération	Code	Opération
0	ADD	16	STOREM
1	SUB	17	JSR
2	ADDC	18	PUSH
3	SUBC	19	STORE
4	INC	20	EORM
5	DEC	21	ANDM
6	EOR	22	ORM
7	AND	23	NANDM
8	OR	24	NORM
9	NAND	25	LOADM
10	NOR	26	COMPM
11	NOT	27	COMP
12	LSR	28	LOAD
13	LSL	29	EXCH
14	ASR	30	PULL
15	ASL	31	JMP

Annexe 3

Organigramme de Contrôle

Caractéristiques générales

L'automate de contrôle de HYETI comporte 175 états. Il a 35 entrées : registre instruction, paramètres (numéros de registres), numéro d'état courant (boucle de réaction), indicateurs d'état. Cet automate est implanté à l'aide d'une structure hiérarchisée comportant 7 PLAs sur deux niveaux.Les 81 sorties des 4 PLAs du premier niveau (PLA41, PLA42, PLA43, PLA44) sont réparties entre des commandes directes, le numéro de l'état suivant, et des commandes en champs servant d'entrées aux 3 PLAs de décodage du deuxième niveau (DRTEST, DRNORM, DECUAL) pour arriver aux 153 commandes qui arrivent à la partie opérative.

Légende de l'organigramme

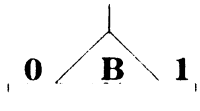
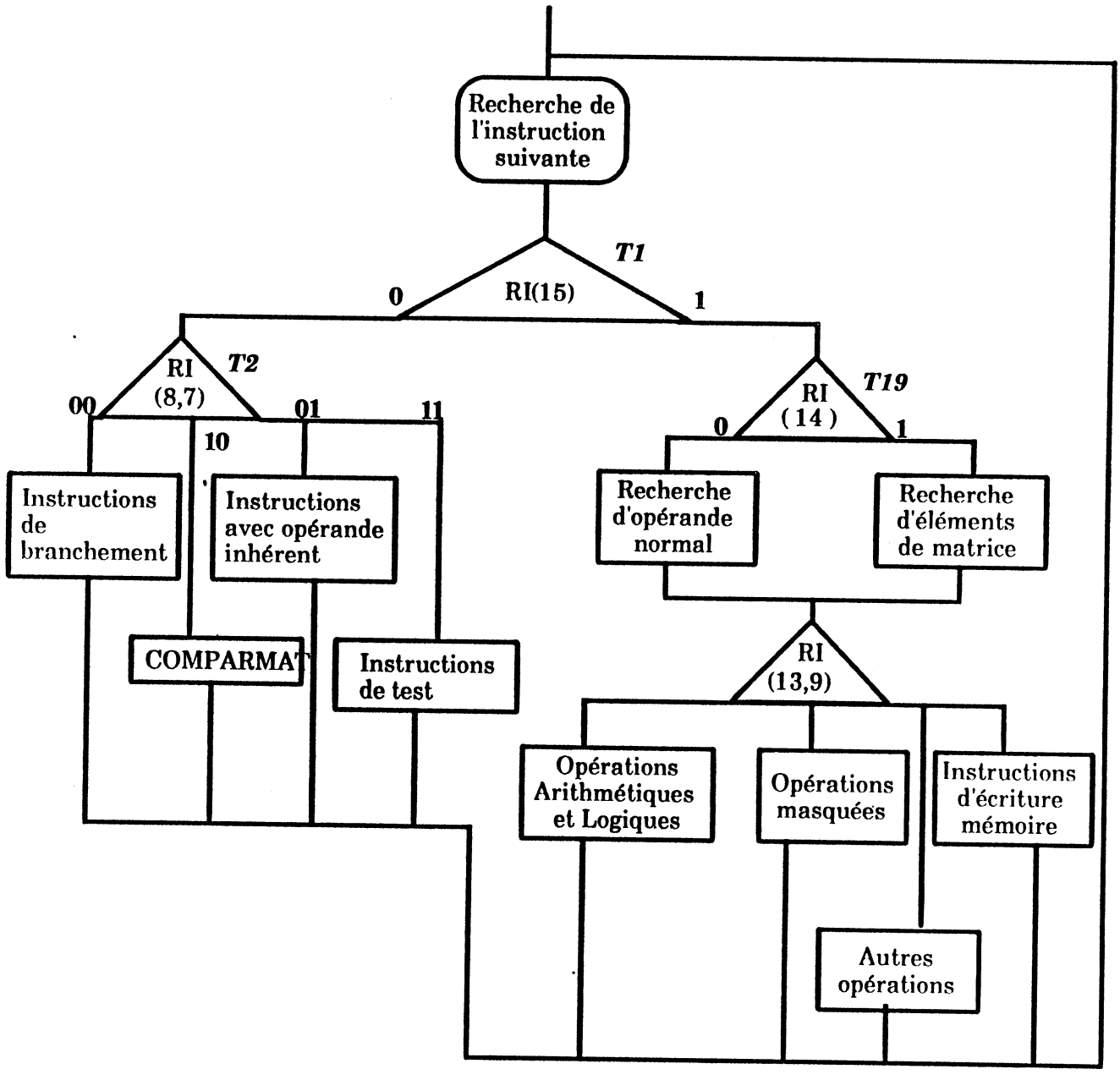
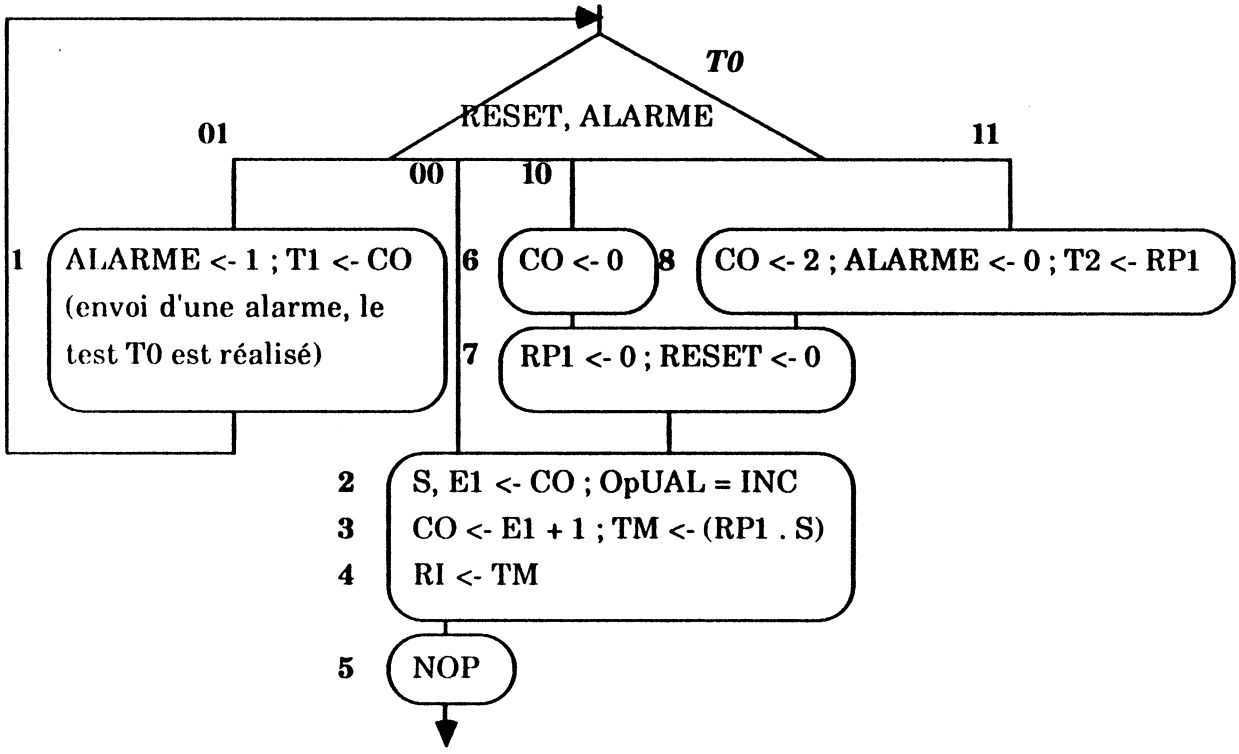
A <- B	Transfert du registre B dans le registre A.
A <- B + C	Le registre A est chargé avec le résultat de reg.B+reg. C.
action 1 ; action 2	Les actions sont exécutées en parallèle.
RE<8 : 4>	représente les bits 8 à 4 de RE.
RE<5, 9>	représente les bits 5 et 9 de RE.
R(RI<3 : 0>)	représente le contenu du registre dont le numéro est donné dans les bits 3 à 0 du registre RI.
(RP1 . S)	représente le contenu du mot mémoire situé à l'adresse contenue dans S, dans la page dont le numéro est contenu dans RP1.
(RP(RI <14>) . S)	Même signification que ci-dessus, mais le registre page à utiliser est paramétré par le bit 14 du registre RI : **RI<14> = 0** : le registre est **RP1** **RI<14> = 1** : le registre est **RP2**.
B - C	Comparaison des registres B et C.
Op(RI<12 : 9>)	représente une opération de l'U.A.L dont le code est donné dans les bits 12 à 9 du registre RI.
OpUAL = INC	indique que l'on envoie à l'U.A.L les commandes nécessaires à une incrémentation.
	test de la valeur de B

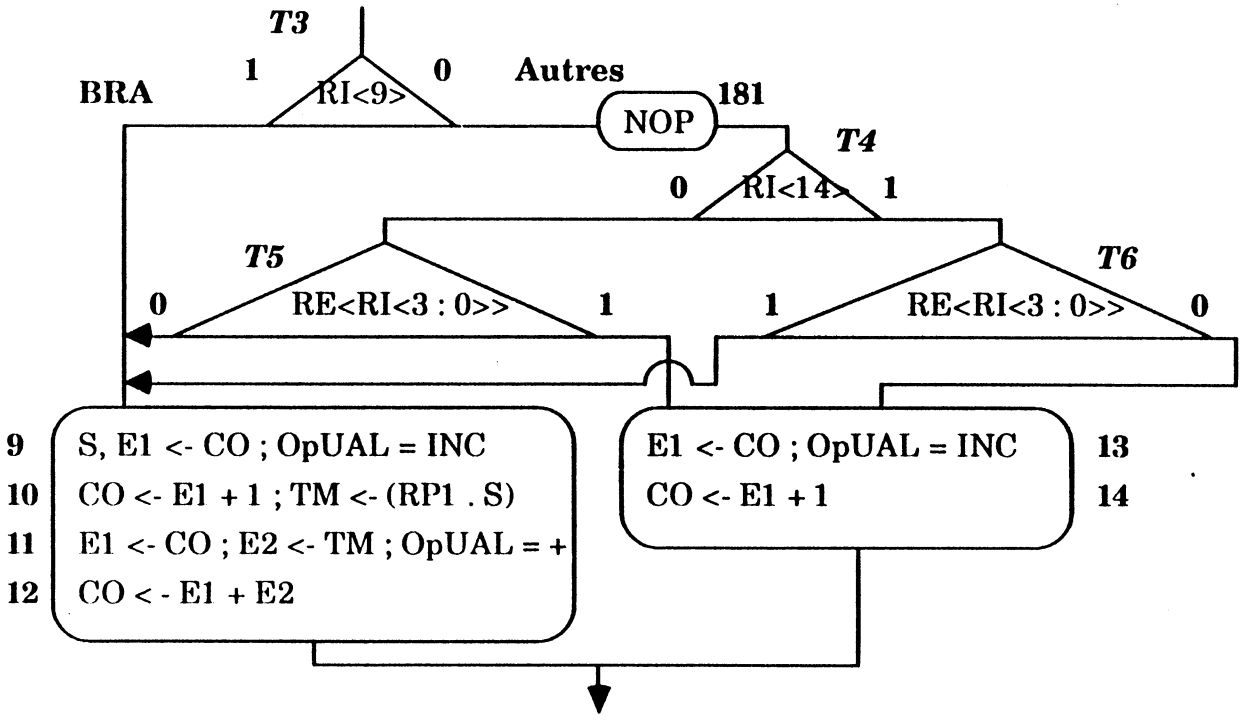
Schéma global de l'organigramme de contrôle



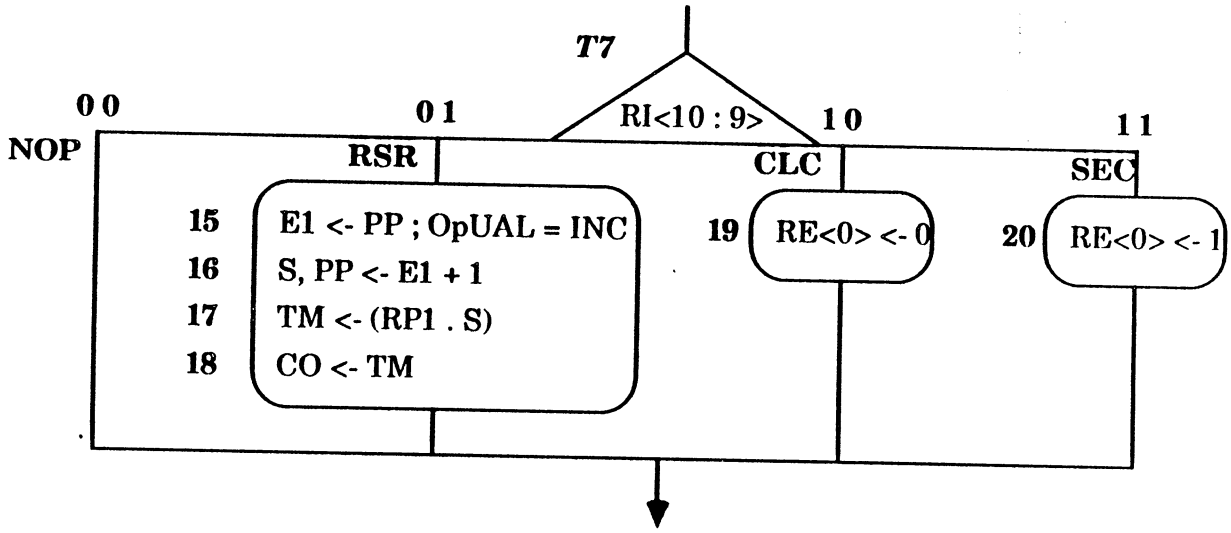
Recherche de l'instruction suivante



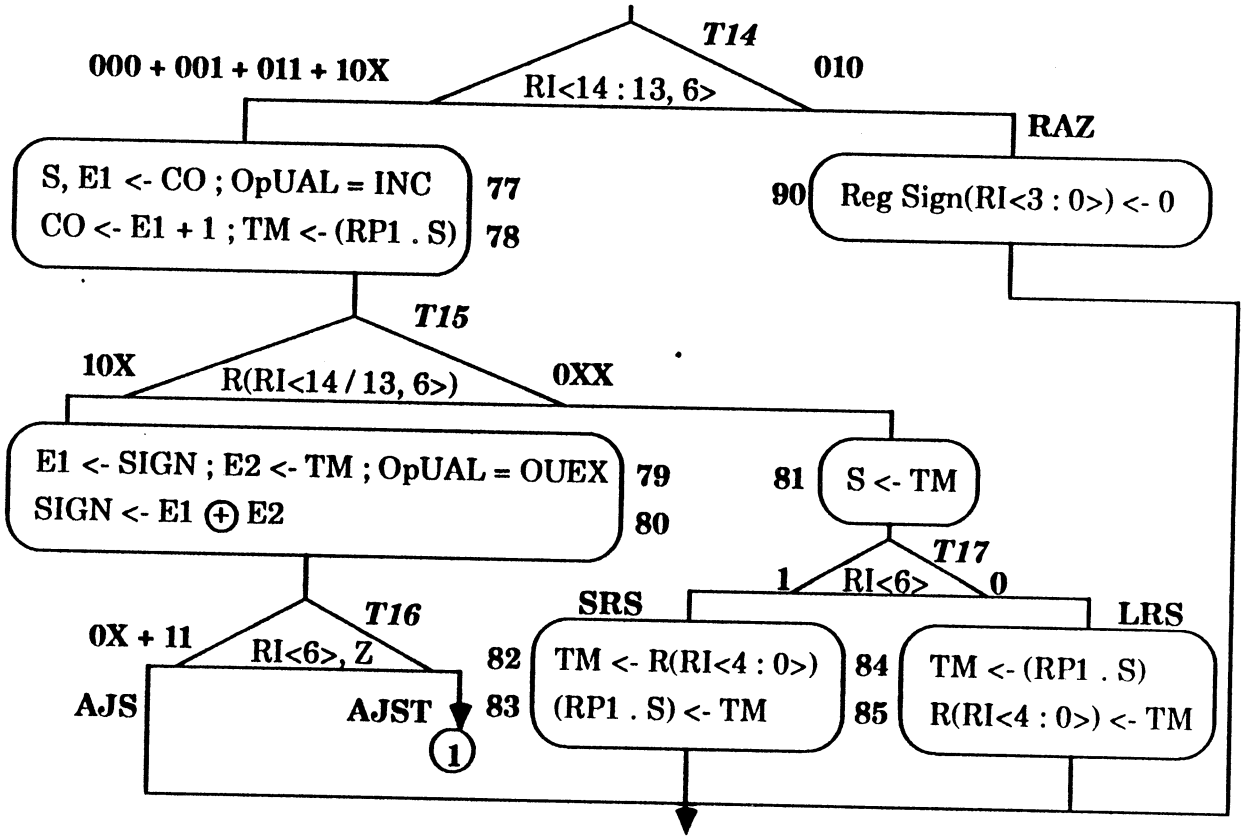
Instructions de branchement



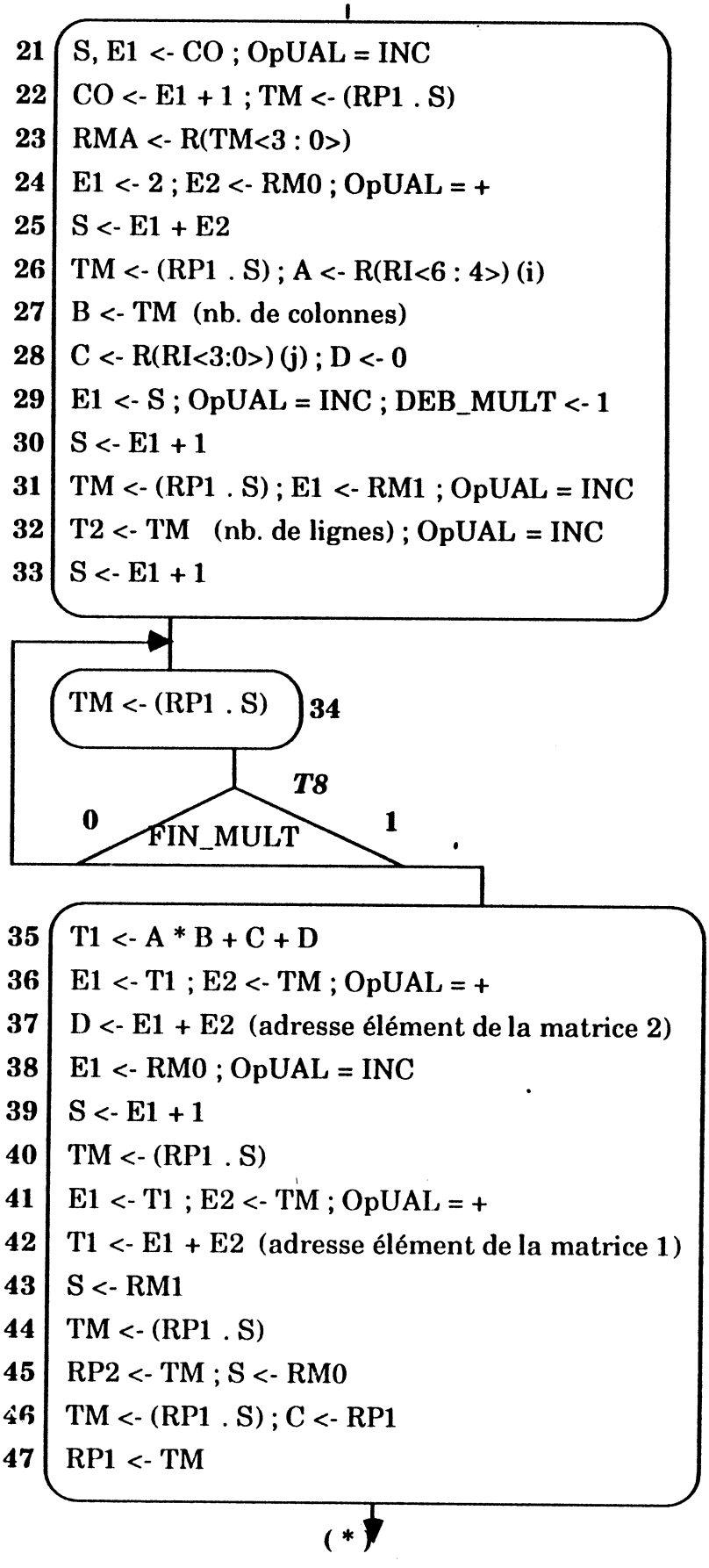
Instructions avec opérande inhérent

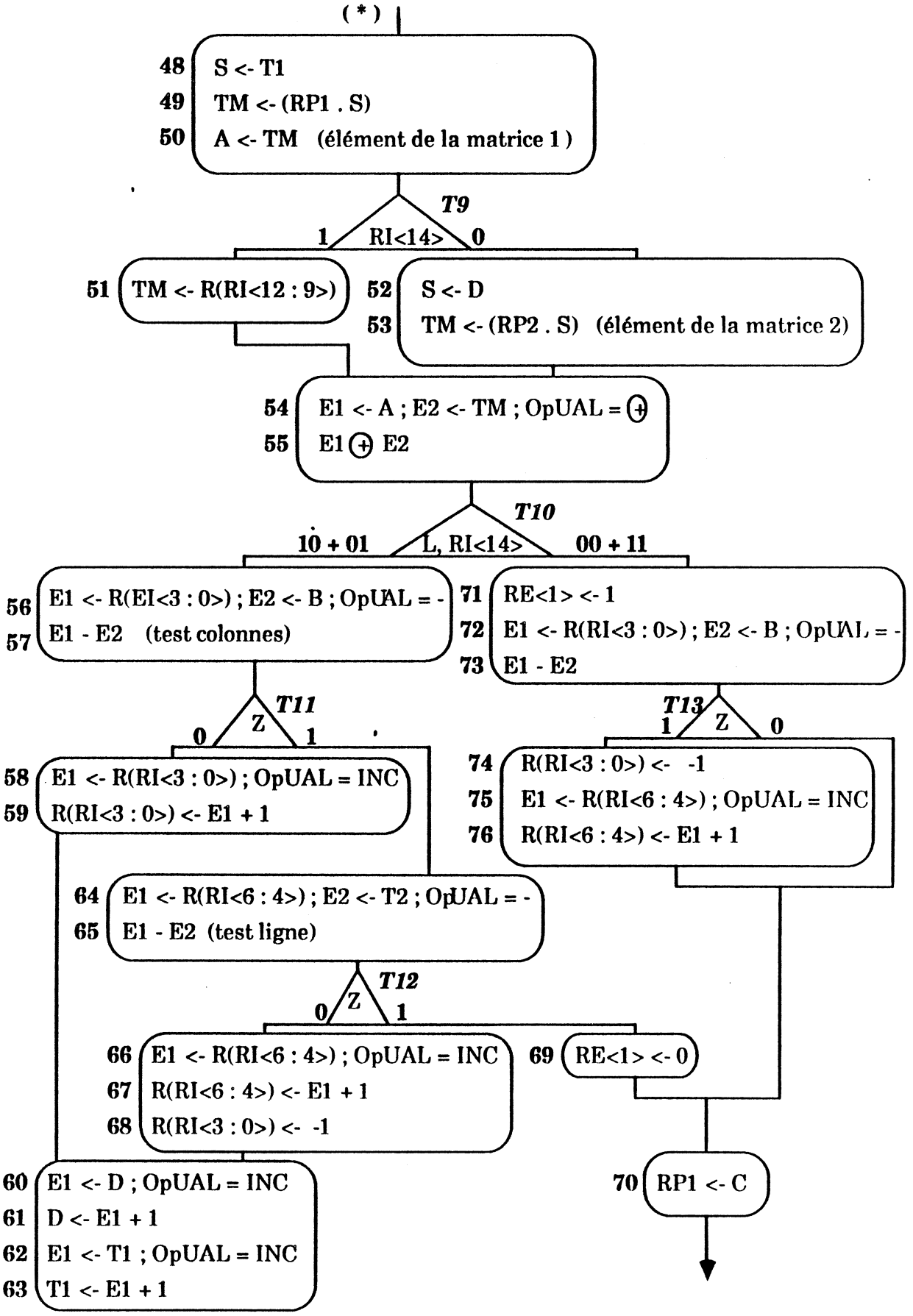


Instructions de test

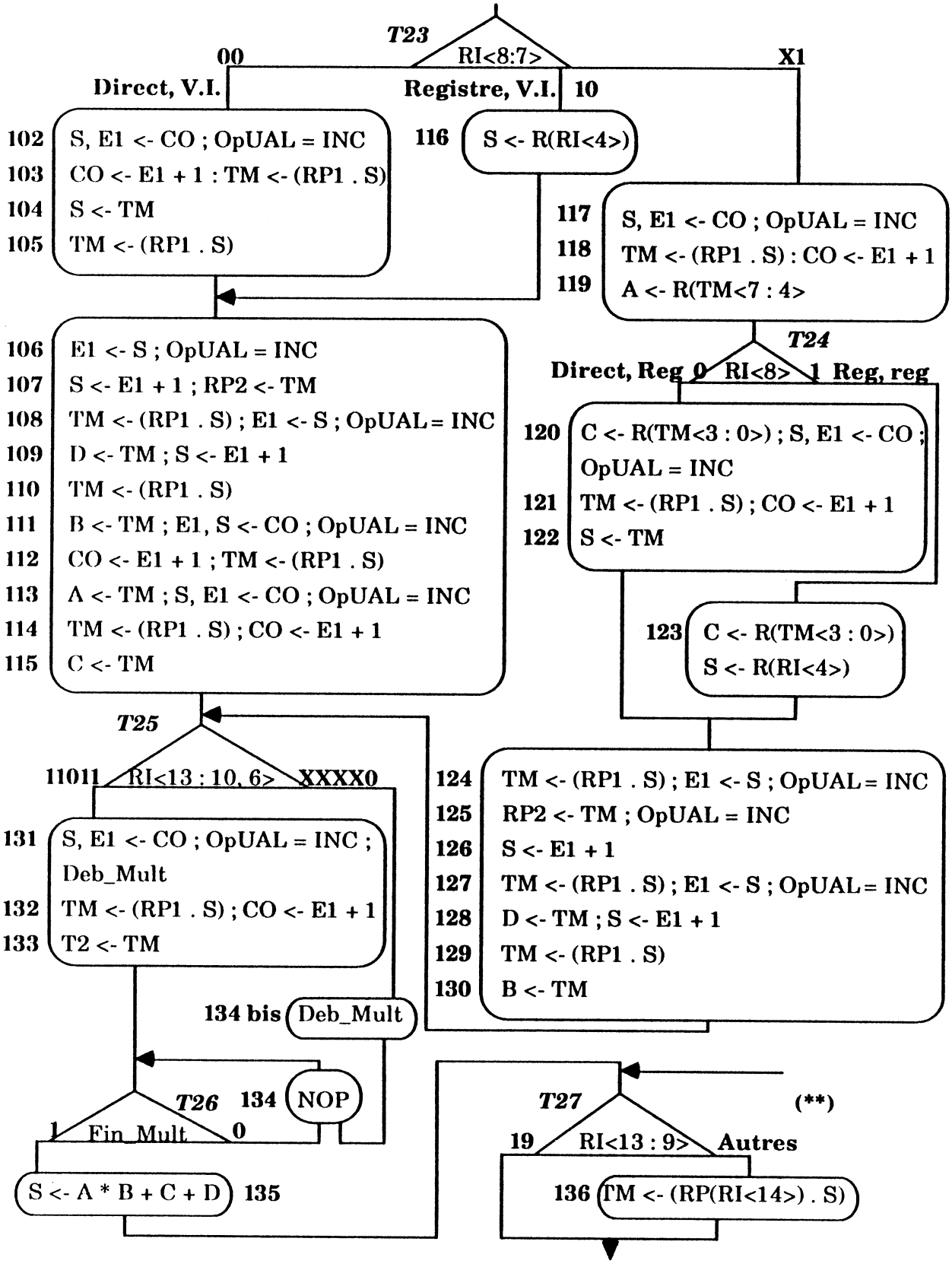


COMPARMAT

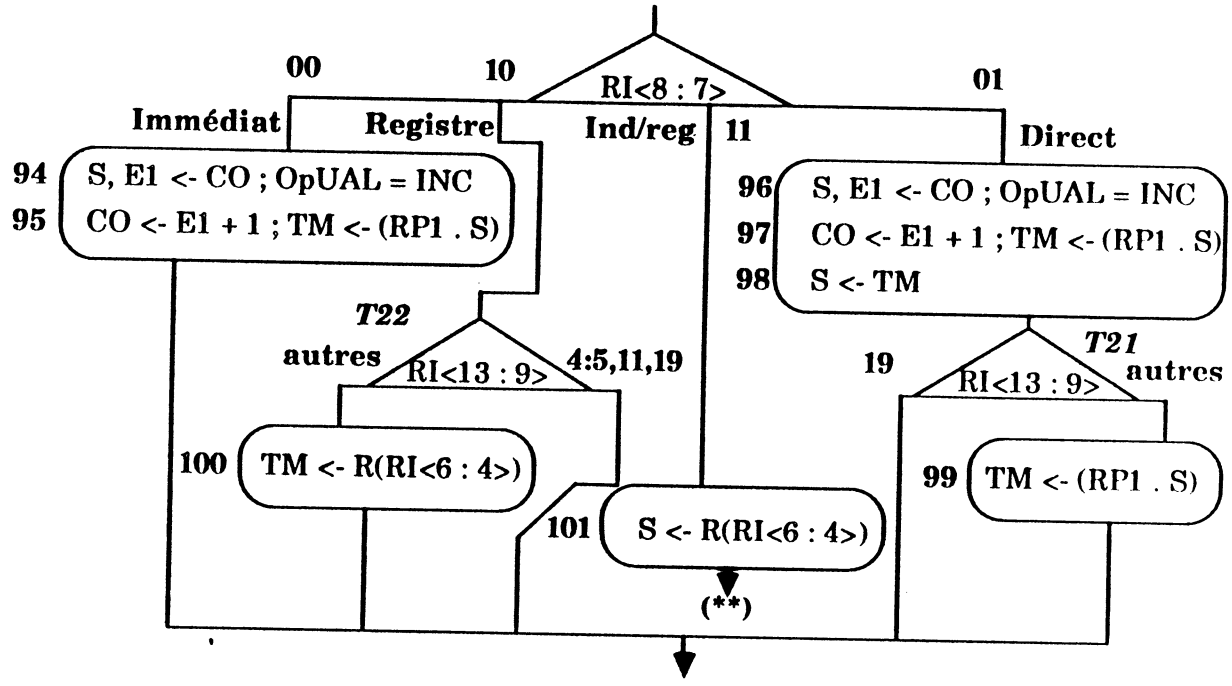

21 S, E1 <- CO ; OpUAL = INC
22 CO <- E1 + 1 ; TM <- (RP1 . S)
23 RMA <- R(TM<3 : 0>)
24 E1 <- 2 ; E2 <- RM0 ; OpUAL = +
25 S <- E1 + E2
26 TM <- (RP1 . S) ; A <- R(RI<6 : 4>) (i)
27 B <- TM (nb. de colonnes)
28 C <- R(RI<3:0>) (j) ; D <- 0
29 E1 <- S ; OpUAL = INC ; DEB_MULT <- 1
30 S <- E1 + 1
31 TM <- (RP1 . S) ; E1 <- RM1 ; OpUAL = INC
32 T2 <- TM (nb. de lignes) ; OpUAL = INC
33 S <- E1 + 1
TM <- (RP1 . S)
34
T8
0
FIN_MULT
1
35 T1 <- A * B + C + D
36 E1 <- T1 ; E2 <- TM ; OpUAL = +
37 D <- E1 + E2 (adresse élément de la matrice 2)
38 E1 <- RM0 ; OpUAL = INC
39 S <- E1 + 1
40 TM <- (RP1 . S)
41 E1 <- T1 ; E2 <- TM ; OpUAL = +
42 T1 <- E1 + E2 (adresse élément de la matrice 1)
43 S <- RM1
44 TM <- (RP1 . S)
45 RP2 <- TM ; S <- RM0
46 TM <- (RP1 . S) ; C <- RP1
47 RP1 <- TM
(*)


(*)
48 S <- T1
49 TM <- (RP1 . S)
50 A <- TM (élément de la matrice 1)
T9
1 RI<14> 0
51 TM <- R(RI<12 : 9>)
52 S <- D
53 TM <- (RP2 . S) (élément de la matrice 2)
54 E1 <- A ; E2 <- TM ; OpUAL = ⊕
55 E1 ⊕ E2
T10
10 + 01 L, RI<14> 00 + 11
56 E1 <- R(EI<3 : 0>) ; E2 <- B ; OpUAL = -
57 E1 - E2 (test colonnes)
71 RE<1> <- 1
72 E1 <- R(RI<3 : 0>) ; E2 <- B ; OpUAL = -
73 E1 - E2
T11
0 Z 1
T13
1 Z 0
58 E1 <- R(RI<3 : 0>) ; OpUAL = INC
59 R(RI<3 : 0>) <- E1 + 1
74 R(RI<3 : 0>) <- -1
75 E1 <- R(RI<6 : 4>) ; OpUAL = INC
76 R(RI<6 : 4>) <- E1 + 1
64 E1 <- R(RI<6 : 4>) ; E2 <- T2 ; OpUAL = -
65 E1 - E2 (test ligne)
T12
0 Z 1
66 E1 <- R(RI<6 : 4>) ; OpUAL = INC
67 R(RI<6 : 4>) <- E1 + 1
68 R(RI<3 : 0>) <- -1
69 RE<1> <- 0
60 E1 <- D ; OpUAL = INC
61 D <- E1 + 1
62 E1 <- T1 ; OpUAL = INC
63 T1 <- E1 + 1
70 RP1 <- C

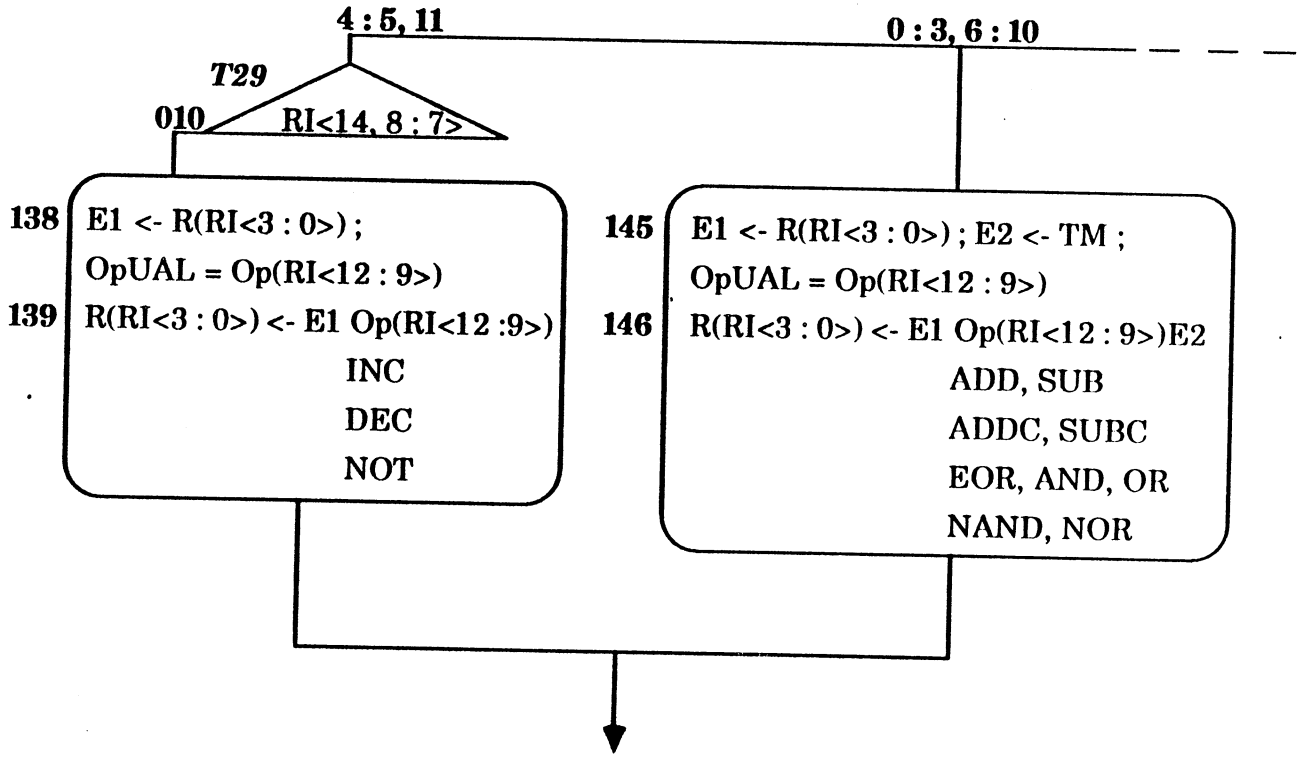
Recherche d'éléments de matrice

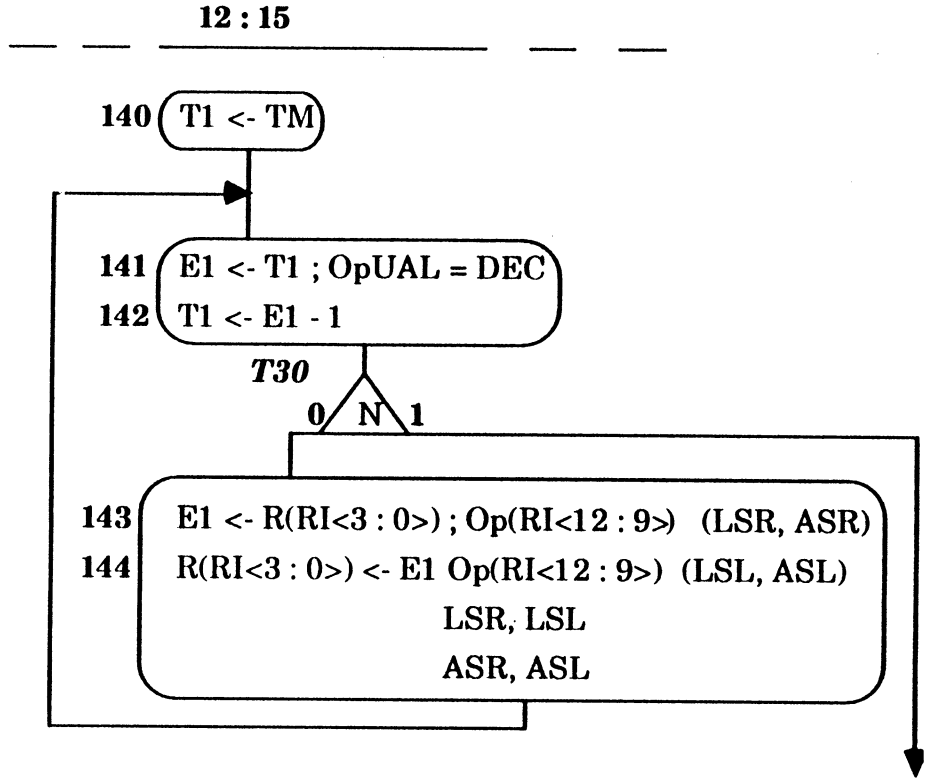


Recherche d'opérande normal

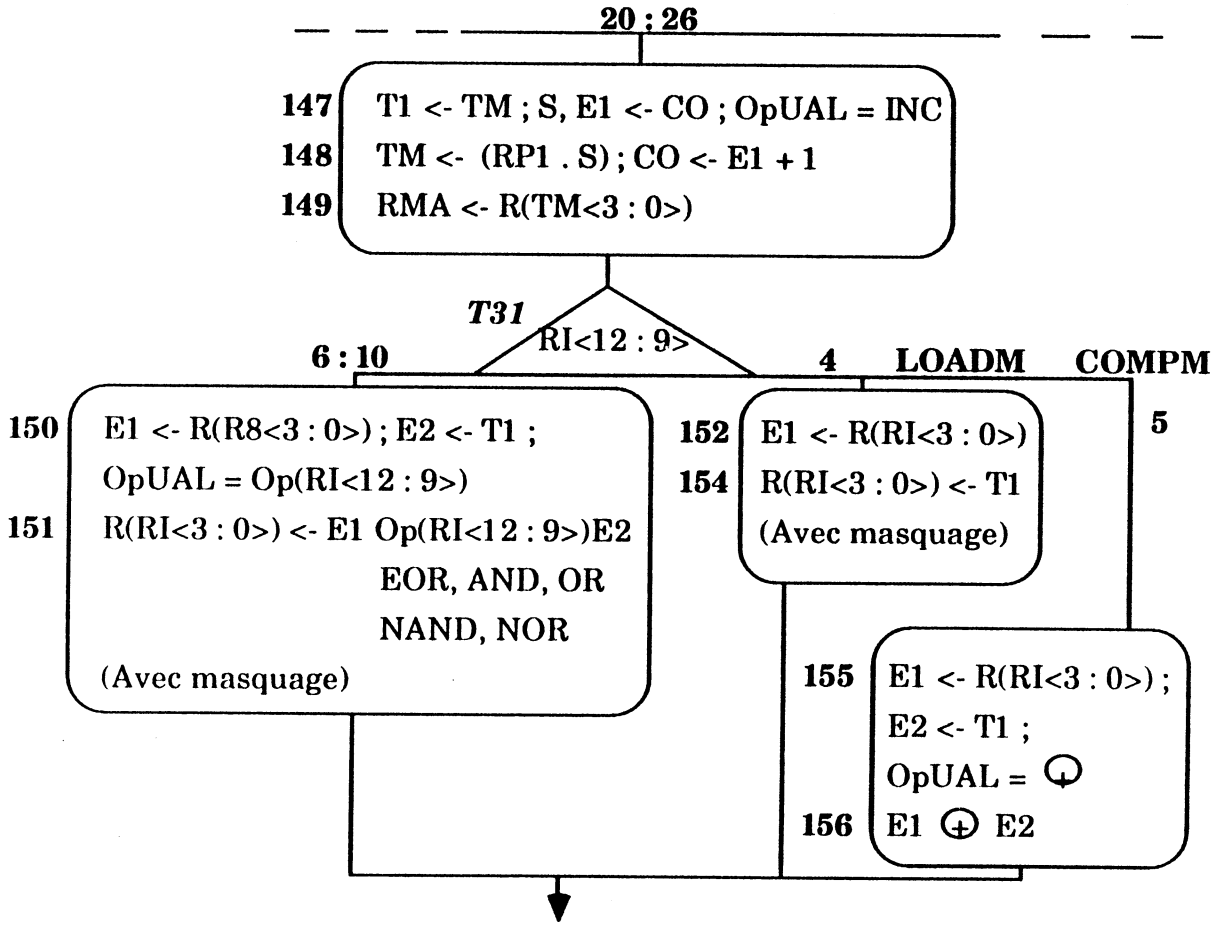


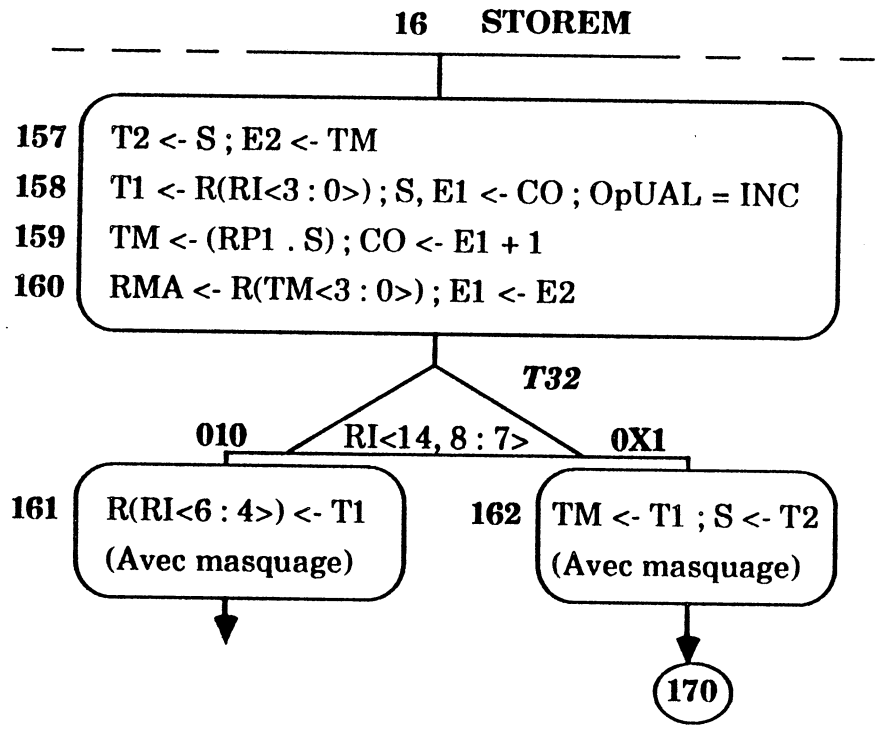
Opérations arithmétiques et logiques



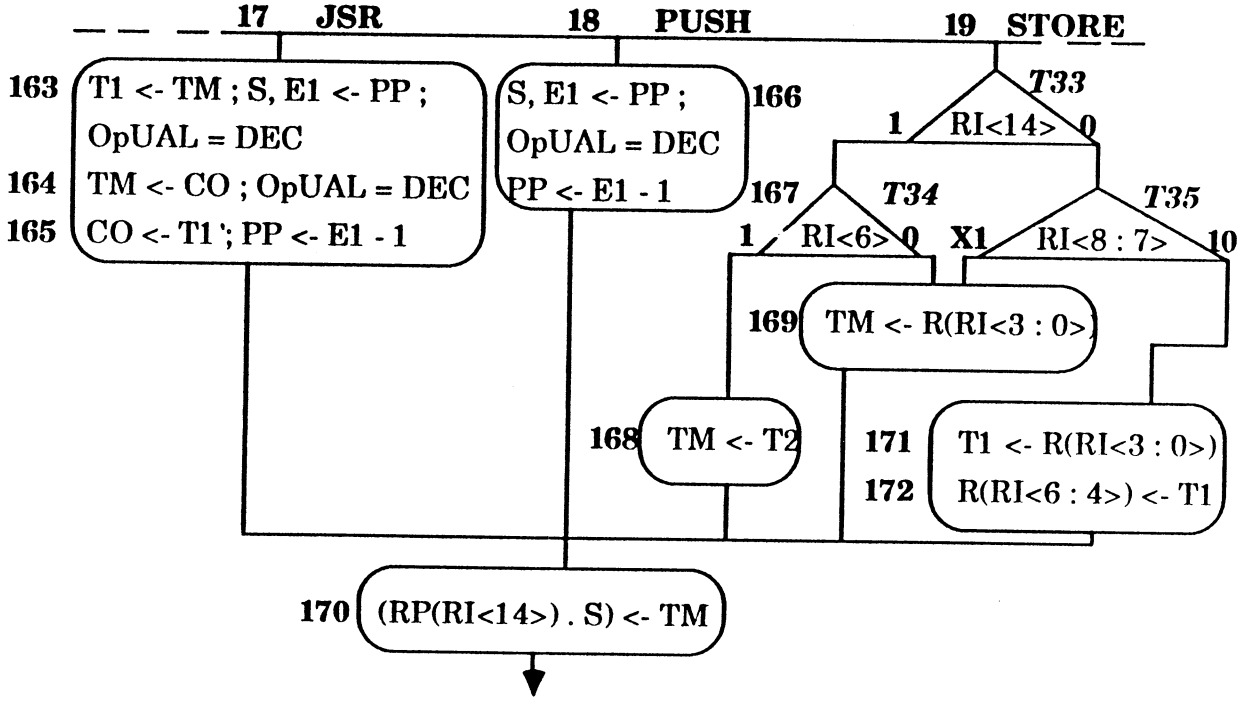


Opération masquées

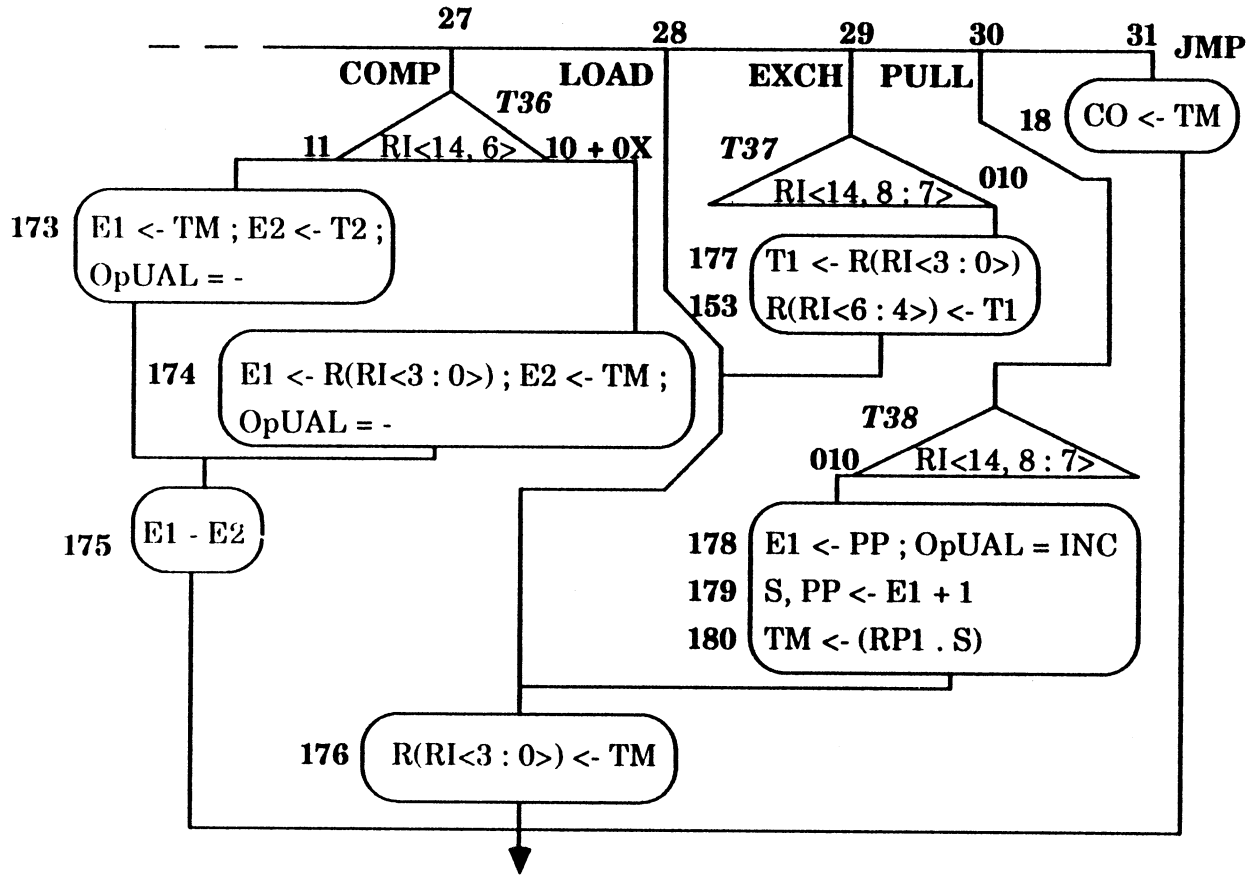




Instructions d'écriture mémoire

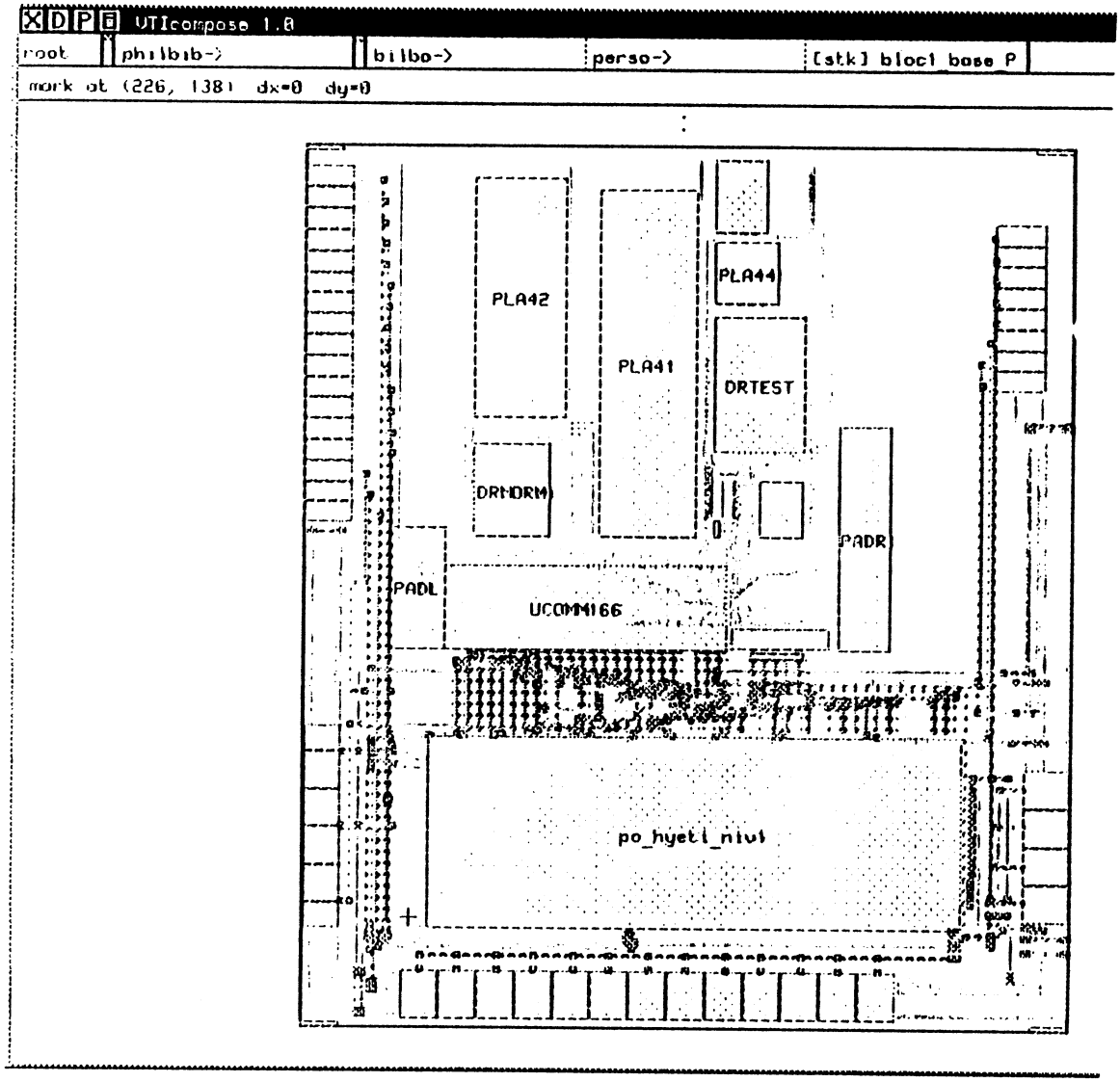


Autres opérations

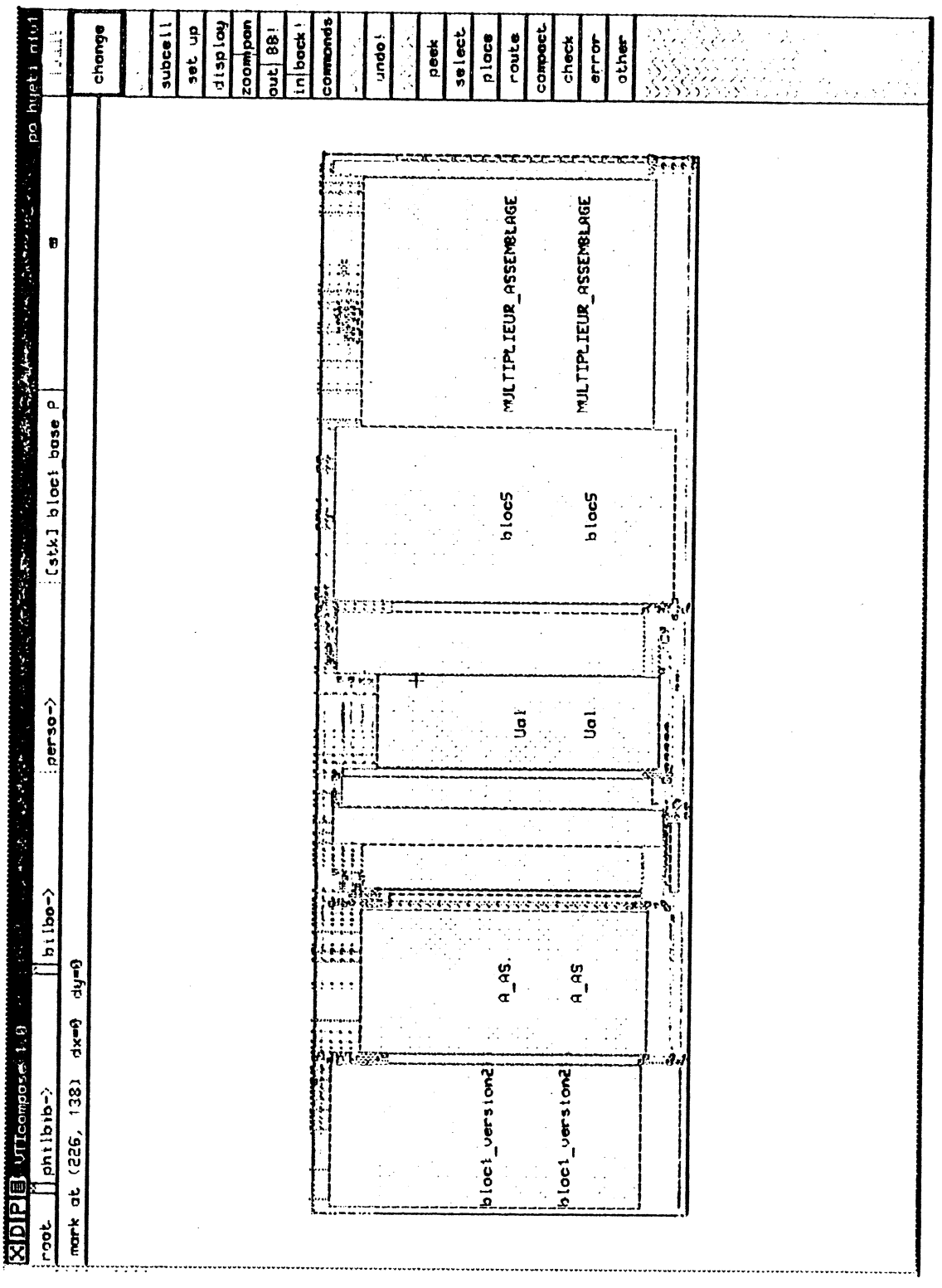


Annexe 4

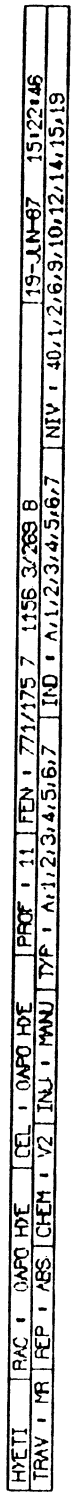
Implantation de HYETI

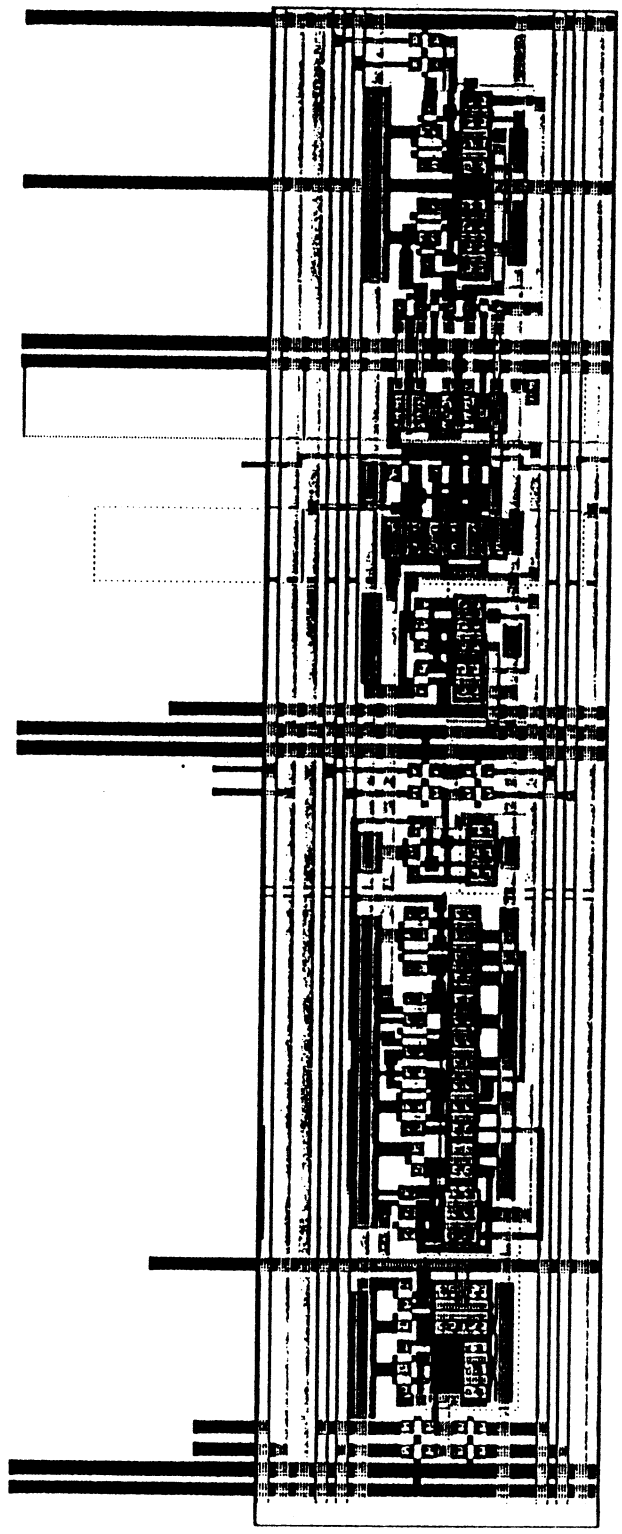


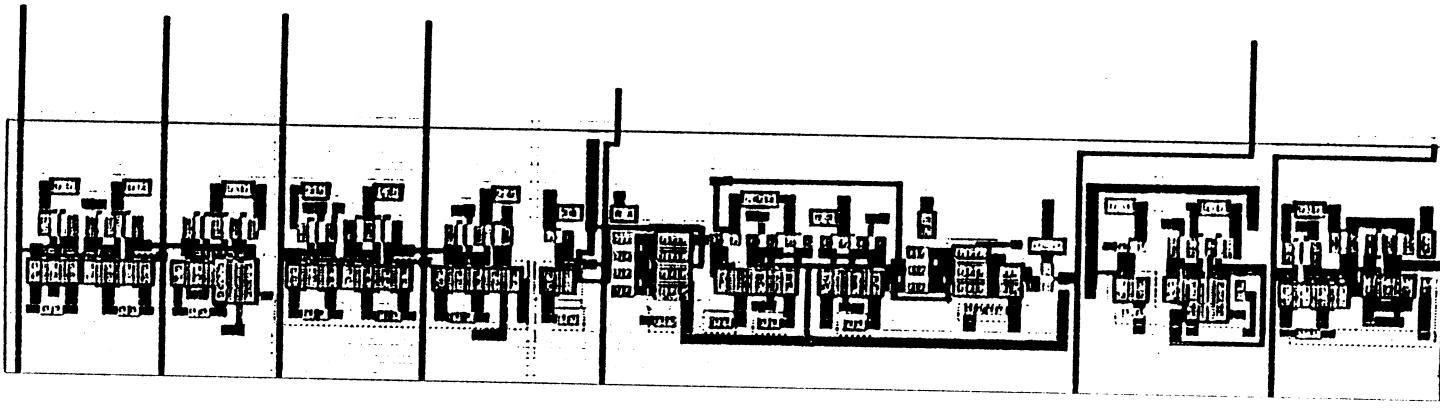
Plan de masse du microprocesseur HYETI dans sa version préliminaire. Toute l'implantation réalisée au LCS a été faite à l'aide du système de CAO VTI de VLSI Technology.



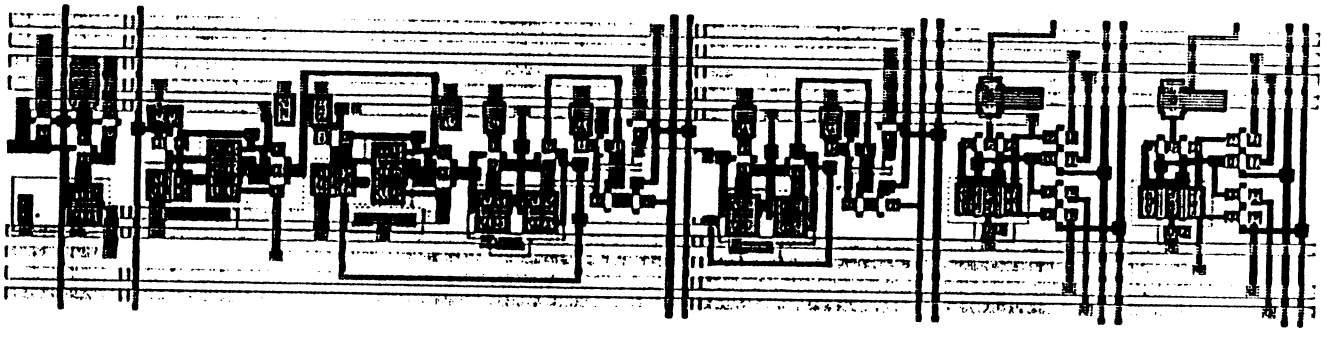
Plan de masse de la Partie Opérative


HYETT
RAC ' OAPO HDE
CEL ' OAPO HDE
PROF ' 11
FEN ' 771/175 7 1156 3/269 8
19-JAN-87 15:22:46
TRAV ' MR
REP ' ABS
CHEM ' V2
INJ ' MANU
TYP ' A:1,2,3,4,5,6,7
IND ' A:1,2,3,4,5,6,7
NIV ' 40:1,2,6,9,10,12,14,15,19

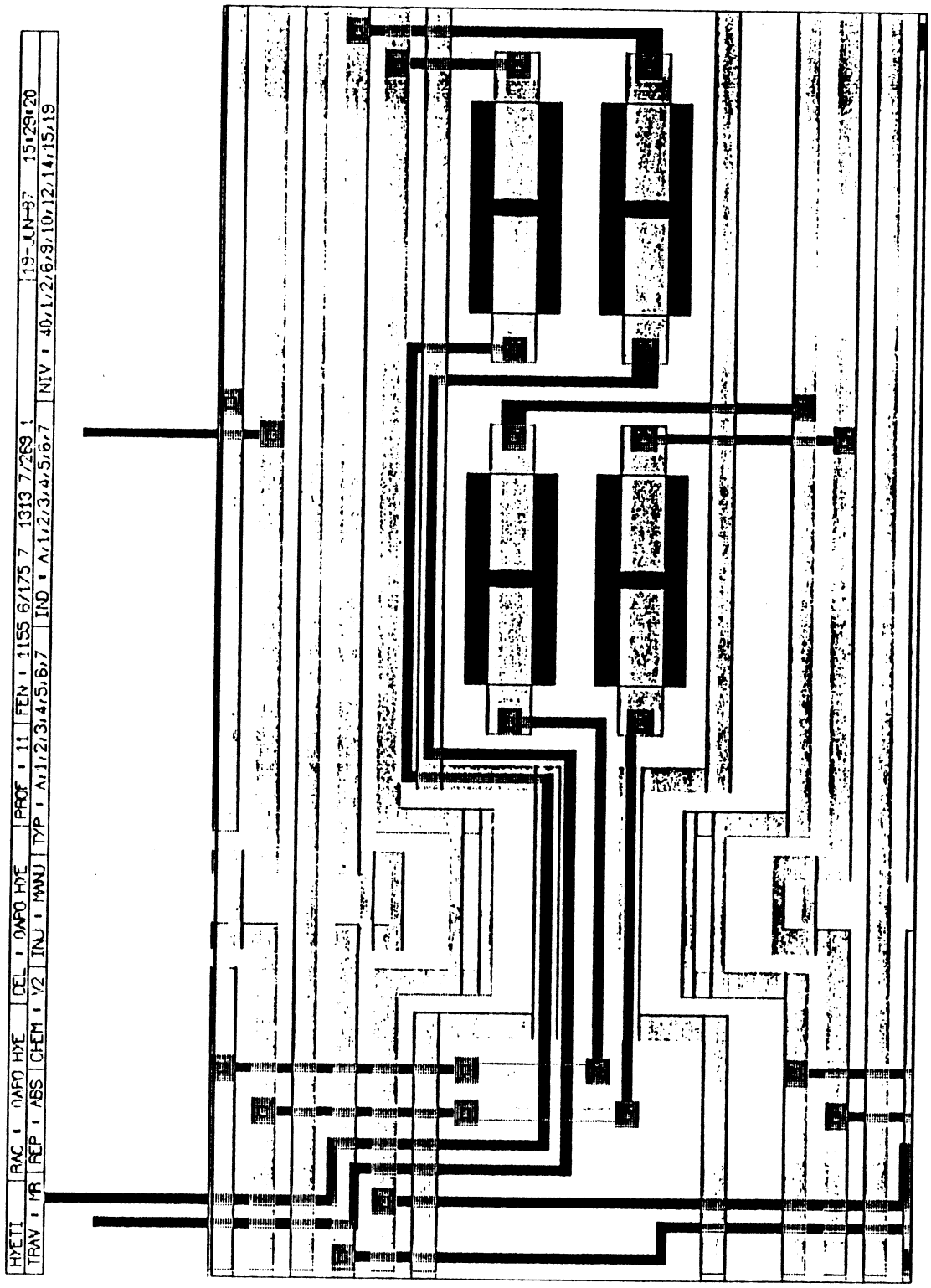




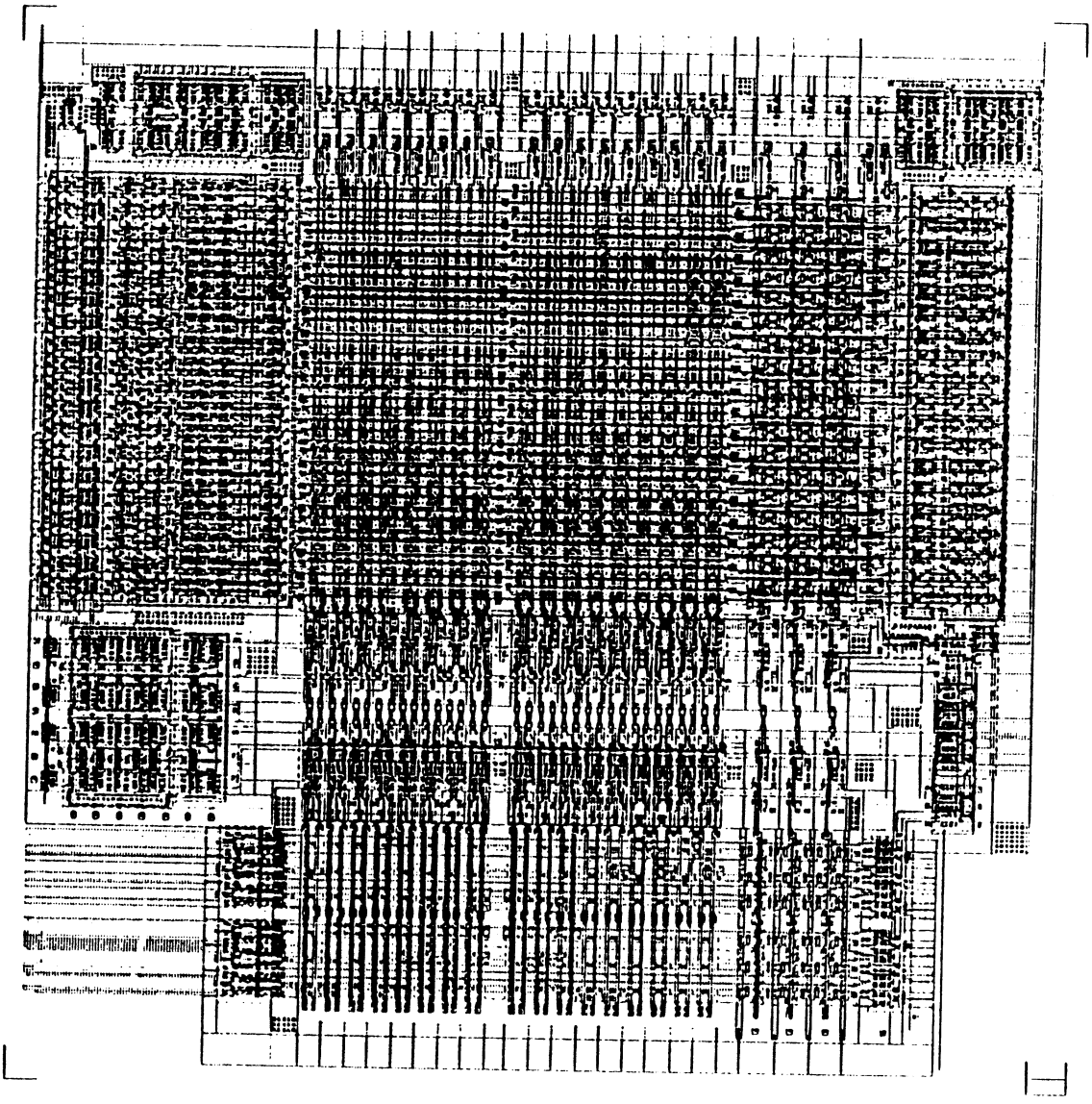
Unité Arithmétique et Logique



Dispositif de Masquage



Dispositif de reconnexion de bus à base de fusibles et d'anti - fusibles.



Exemple de PLA : PLA 44

Références Bibliographiques

[AUB78] "Wafer Scale Integration, Some Approaches to the Interconnection Problem"
R.C. Aubusson, R.J. Gledhill
Microelectronics V-9 1, pp 5 - 10
Janvier 1978

[BEL81] "Conception d'un Séquenceur de Microprocesseur"
C. Bellon, G. Saucier
RAIRO Automatique Vol. 15, n° 2
1981

[CAL72] "A Means of Reducing Custom LSI Interconnection Requirements"
D.F. Calhoun, L.P. MacNamee
IEEE Journal of Solid State Circuits, Vol SC 7, n° 5, pp 395 - 404
Octobre 1972

[CAS76] "Analysis of fault mechanisms in CMOS gates"
G.R Case
Proceedings of the 13th Design Automation Conference
1976

[CAS84] "Architectures for W.S.I, Annex 1 : Fault-tolerant architectures"
P. Caspi
Deliverable B/1, Projet ESPRIT 244 - HYETI
1984

[CHA85] "Laser Linking Technology for R.V.L.S.I"
G.H. Chapman
International Workshop on Wafer Scale Integration
Université de Southampton, Juillet 1985

[CHE85] "A Programmable Switch for Fault-tolerant Wafer Scale Integration of Processor Arrays"
G. Chevalier, G. Saucier
International Workshop on Wafer Scale Integration
Université de Southampton, Juillet 1985

[COU76] "Etude d'un calculateur tolérant les pannes : ses fiabilité, sécurité, performance et coût"
B. Courtois
Thèse de Docteur Ingénieur
INP Grenoble, Décembre 1976

[DAN86] "Conception de PLA CMOS"
A. Dandache
Thèse de doctorat de l'I.N.P.G.
INP Grenoble, Juillet 1986

[FLA86] Two 13-ns 64 K CMOS SRAM's with very low Active Power and Improved Asynchronous Circuit Techniques"
Stephen T. Flanagan, et al.
IEEE Journal of Solid State Circuits, Vol SC 21, n° 5, pp 692 - 703
Octobre 1986

[FRI84] "The R.P.I Wafer Scale Integration Silicon Compiler"
M. Friedman et al.
International Conference on Computer Design
New-York, Octobre 1984

[GEN84] "Conception d'un microprocesseur pour des applications de haute sécurité : HSURF"
P. Genestier, J.F Poirier
Rapport de D.E.A d'Informatique
I.N.P Grenoble, Mai 1984

[HAS82] "Parallel signature analyzers, detection capabilities and extensions"
S.Z. Hassan, D.J. Lu, E.J. Mc Cluskey
CRC technical report, n° 82 - 20, pp 1 - 6
Décembre 1982

[HED82] "Wafer Scale Integration of Parallel Processors"
K.S. Hedlung
PhD Thesis
Purdue University, West Lafayette, IN
Novembre 1982

[HIG76] "A Highly Efficient Redundancy Scheme : Self Purging Redundancy"
IEEE Transactions on computers, Vol C 25, n° 6
Juin 1976

[HYE86] "Bibliography on Yield Analysis"
Deliverable A/1 - Projet ESPRIT 244 - HYETI
1986

[JAY84] "A Testable Microprocessor for Process Control"
C. Jay, G. Saucier, P. Genestier
International Conference on Computer Design , pp 284 - 289
New-York, Octobre 1984

[JAY86] "HSURF, Un microprocesseur facilement testable pour des applications à haute sûreté de fonctionnement"
C. Jay
Thèse de Docteur de l'I.N.P.G
I.N.P Grenoble, Juin 1986

[KAN84] "A 256 K DRAM with Descrambled Redundancy Test Capability"
Dieter Kantz, et al.
IEEE Journal of Solid State Circuits, Vol SC 19, n° 5, pp 596 - 602
Octobre 1984

[KAT86] "Switch Design for Soft-Configurable WSI Systems"
M. Katevenis, M. Blatt
Proceedings of the IFIP WG 10.5 Workshop on WSI, pp 255 - 270
Elsevier Science Publishers
Grenoble Mars 1986

[LEA86] "A W.S.I Image Processing Module"
R.M. Lea
Proceedings of the IFIP WG 10.5 Workshop on WSI, pp 43 - 58
Elsevier Science Publishers
Grenoble Mars 1986

[LEI85] "Wafer Scale Integration of Systolic Arrays"
Tom Leighton, Charles E. Leiserson
IEEE Transactions on Computers, Vol C 34, n° 5, pp 448 - 461
Mai 1985

[LEI86] "A Survey of Algorithms for Integrating Systolic Arrays"
T. Leighton, C. Leiserson
Proceedings of the IFIP WG 10.5 Workshop on WSI, pp 177 - 195
Elsevier Science Publishers
Grenoble Mars 1986

[MAN84] "Sources of failures and yield improvement for VLSI and restructurable interconnects for RVLSI and WSI
Part 1 - Source of failures and yield improvement for VLSI"
T.E Mangir
Proceedings of the IEE, vol 72, n°6, Juin 1984

[MUR64] "Cost Size Optima of Monolithic Integrated Circuits"
B.T. Murphy
Proceedings of the IEEE, Vol 52, pp 1535 - 1545
Décembre 1964

[NEW85] "Is WSI too big ?, Selecting an appropriate componant size for redundant logic"
Michael J. Newman
Panel Session : Is U.L.S.I "for real" ?
Design for Fault Tolerance Conference, 1985

[OBR82] "Etude comparative de différentes méthodes de conception des parties contrôle des microprocesseurs"
M. Obrebska
Thèse de Docteur-Ingénieur
Grenoble - Juin 1982

[SAK84] "A low Power 46 ns 256 Kbit CMOS Static RAM with Dynamic Double Word Line"
Takayosu Sakuraï, et al.
IEEE Journal of Solid State Circuits, Vol SC 19, n° 5, pp 578 - 585
Octobre 1984

[SAU87] "Rule Based Synthesis Environment. Application to a dedicated Microprocessor"
G. Saucier, P. Genestier, C. Jay
Proceedings of the IFIP WG 10.5 Workshop on Fast Prototyping
Elsevier Science Publishers
Grenoble Mars 1987

[SCH78] "Reliability of CMOS integrated circuits"
G.L. Schnable et al.
IEEE Computer, Vol 11, n°10, pp 6 - 17
Octobre 1978

[SEE67] "Yield Economic, and Logistical Models for Complex Digital Arrays"
R.B. Seeds
IEEE Int. Conv. Rec., pt 6, pp 60 - 61
Avril 1967

[SHA84] "Electron Beam Customization, Repair, and Testing of Wafer-Scale Circuits"
D.C. Shaver
Solid State Technology, pp 135 - 139
Février 1984

[SMI81] "Laser Programmable Redundancy and Yield Improvement in a 64K DRAM"
R.T. Smith, et al.
IEEE Journal of Solid State Circuits, Vol SC 16, pp 506 - 514
Octobre 1981

[STA83] "Integrated Circuit Yield Statistics"
C.H. Stapper, F.M. Armstrong, K. Saji
Proceedings of the IEEE, Vol 71, pp 453 - 470
Avril 1983

[STI77] "Coding for Random Access Memory"
J.J. Stiffler
Proceedings of the 1977 International Symposium on Fault–Tolerant Computing

[SUB79] "Integrated Circuit Engineering"
G.E. Subak-Sharpe, A.B. Glaser
Addison-Wesley, Reading, 1979

[TRY] "Quadded Logic"
J.G. Tryon
Redundancy Techniques for Computing Systems
Wilcox & Mann eds, Spartan Book

[VAL86] "Wafer Scale Integration Packaging"
Christian Val
Proceedings of the IFIP WG 10.5 Workshop on WSI, Grenoble Mars 1986
Elsevier Science Publishers

[WAK76] " Microcomputer Reliability Improvement using TMR"
J.F. Wakerly
Proceedings of the IEEE, Vol 64, n° 6
Juin 1976

[WAN84] "Linear feedback shift register design for VLSI system testing"
L.T. Wang, E.J. Mc Cluskey
Proceedings of FTCS 14, pp 360 - 365
Juin 1984

[WSI86] " HYETI microprocessor architecture description"
Deliverable D29, Projet ESPRIT 824

Table des Matières

Sommaire

Introduction 15

Chapitre 1 : Principes généraux de la reconfiguration 17

1 Défauts de fabrication et Rendement des circuits intégrés .. 19

1.1 Défauts de fabrication 19

1.2 Rendement 21

1.2.1 Modèles de rendement existants 22

1.2.2 Modèle adapté à l'évaluation de la taille des blocs reconfigurables 25

2 Redondance et Configuration d'un circuit en fin de fabrication 27

2.1 Redondance dans les systèmes tolérant les pannes 27

2.2 Redondance dans les circuits tolérant les défauts 28

2.2.1 Les dispositifs programmés par LASER 30

2.2.2 Les dispositifs programmés par faisceau d'électrons 31

2.2.3 Les dispositifs programmés par logique 31

3 Architectures intégrables sur tranche entière 31

3.1 Mémoires 32

3.2 Réseaux systoliques 36

3.3 Systèmes à base de microprocesseurs 37

3.3.1 Architecture 37

3.3.2 Reconfiguration d'un circuit de type microprocesseur 39

3.3.2.1 Terminologie 40

3.3.2.2 Stratégie générale de reconfiguration 40

Chapitre 2 : Reconfiguration de la partie opérative d'un microprocesseur 43

1 Partitionnement 45

1.1 Partitionnement fonctionnel 46
1.2 Partitionnement en tranches de bit 47

2 Choix d'une solution 48
2.1 Critères de choix 48
2.2 Taille des portes de base 49

3 Reconfiguration des blocs 50
3.1 Réplication massive 51
3.2 Ajout de réserve pour les éléments modulaires 53
3.3 Ajout de tranches de bit supplémentaires 56
3.3.1 Propagation de données dans un seul sens 56
3.3.2 Propagation de données dans plusieurs directions 57
3.4 Limitations 61

4 Connexion des blocs reconfigurables 61
4.1 Description du problème 61
4.2 Réalisation des différentes possibilités de connexion 64
4.2.1 Utilisation de fusibles et d'anti-fusibles 64
4.2.2 Utilisation de transistors à grille flottante 66
4.2.3 Utilisations de dispositifs logiques 67
4.3 Mémorisation des configurations 73

5 Conséquences pratiques 74
5.1 Choix du style de partitionnement 74
5.2 Stratégie de reconfiguration dans les blocs 74

Chapitre 3 : Reconfiguration de la partie contrôle d'un microprocesseur..... 79
1 Contraintes architecturales 81

2 Conception de parties contrôle reconfigurables 82

3 Reconfiguration des éléments de la partie contrôle 83
3.1 Registres et interconnexions 83

3.2 Réseaux Logiques Programmables (PLAs) 86
3.2.1 Reconfiguration des entrées 86
3.2.2 Reconfiguration des monômes 88
3.2.3 Reconfiguration des sorties 89
3.2.4 Réalisation des commutateurs 90

4 Conséquences pratiques 91

Chapitre 4 : Application au microprocesseur HYETI 93
1 Présentation générale 95
1.1 Caractéristiques principales 95
1.2 Structure de la partie opérative 102
1.2.1 Registres et logique de bus 102
1.2.2 Dispositif de signature programmable 109
1.2.3 Unité Arithmétique et Logique (U.A.L) et Dispositif de Masquage (D.M) 113
1.2.4 Multiplieur 117
1.2.5 Interface mémoire avec l'extérieur, temporisation 122
1.3 Structure de la partie contrôle 124

2 Reconfiguration 127
2.1 Paramètres généraux 127
2.2 Reconfiguration de la partie opérative 129
2.3 Reconfiguration de la partie contrôle 132

Conclusion 137

Annexes 139

Annexe 1 : Valeurs des courbes des figures 1.3, 1.4 ,1.5, 2.22 2.23 et 2.24 141

Annexe 2 : Jeu d'instructions 151
Introduction 153

1 Modes d'adressage 153
1.1 Modes d'adressage "classiques" 154
1.2 Modes d'adressage des éléments de matrice 156
1.2.1 Organisation d'une matrice en mémoire 156
1.2.2 Modes d'adressage 158

2 Instructions 162
2.1 Instructions arithmétiques, logiques et de transfert 163
2.1.1 Instructions utilisant les modes d'adressage Opérande immédiat,Direct,Registre,Indirect par registre, et pour les éléments de matrice : DI, DR, RI, RR 163
2.1.1.1 Instructions arithmétiques 163
2.1.1.2 Instructions logiques 164
2.1.1.3 Instructions de décalage 165
2.1.1.4 Instruction de comparaison 165
2.1.1.5 Instructions de transfert 166
2.1.1.6 Instructions avec masquage 167
2.1.2 Instructions n'utilisant que le mode d'adressage Registre 170
2.2 Instructions de branchement 171
2.3 Instructions avec opérande inhérent 173
2.4 Instructions de rupture de séquence et de manipulation de pile 174
2.5 Instruction de comparaison de deux matrices : COMPARMAT 175
2.6 Instructions de test 177

3 Affectation des numéros de registres et des codes opération 179
3.1 Pour les instructions LRS et SRS 179
3.2 Pour RAZ 180
3.3 Pour les autres instructions 180
3.4 Codes opération 180

Annexe 3 : Organigramme de contrôle 181

Annexe 4 : Implantation de HYETI 195

Références bibliographiques 203

Table des matières 211

Sommaire 213

Table des illustrations 218

Table des illustrations

Chapitre 1 : Principes généraux de la reconfiguration 17

Figures

1.1 : Probabilité de panne d'un circuit au cours de sa vie 20
1.2 : Exemples de fonctions densité de probabilité utilisées par Murphy 23
1.3 : Comparaison de différentes formulation du rendement 24
1.4 : Probabilité d'avoir m défauts dans un circuit 26
1.5 : Fonction de répartition de X 26
1.6 : Anti-fusible laser 30
1.7 : Fusible laser en silicium polycristallin 31
1.8 : Architecture virtuelle de la mémoire vive statique de 4,5 MBits 33
1.9 : Décodage décentralisé des deux fils d'adresse supplémentaires 34
1.10 : Architecture de la périphérie d'une cellule de 64 K x 1 Bit 35
1.11 : Intégration d'un système complet sur un seul circuit intégré 38
1.12 : Architecture générale d'un microprocesseur 41

Chapitre 2 : Reconfiguration de la partie opérative 43

Figures :

2.1 : Structure générale d'une partie opérative en tranches 45
2.2 : Partitionnement fonctionnel de la P.O. 46
2.3a : Partitionnement de la P.O. en tranches de bit 47
2.3b : Exemple de l'utilisation d'une tranche de réserve 48
2.4 : Porte de transfert 49
2.5 : Elémént de mémorisation 50
2.6 : Réorientation des lignes de commande ou d'E/S pour un élément dupliqué 52
2.7 : Réorientation des lignes de commande ou d'E/S pour un élément répliqué plusieurs fois 52

2.8 : Reconfiguration par ajout de deux modules de réserve 53
2.9 : Schéma d'une cellule de réorientation des commandes 54
2.10 : Schéma de la dernière cellule de réorientation des commandes 55
2.11 : Propagation de données à travers les tranches non utilisées 57
2.12 : Reconfiguration d'un opérateur matriciel (exemple d'un multiplieur parallèle) 58
2.13 : Contournement de cellules pour les résultats 59
2.14 : Dispositif de contournement de cellules pour les données circulant en diagonale 60
2.15 : Principe de la connexion de deux blocs n'ayant pas le même nombre de tranches de bits 62
2.16 : Connexion entre un bloc ayant des tranches supplémentaires et un bloc n'en ayant pas 63
2.17 : Connexion de deux blocs ayant chacun des tranches supplémentaires 63
2.18 : Connexion de blocs à l'aide de fusibles et d'anti-fusibles 64
2.19 : Dispositif de connexion avec des transistors à grille flottante 66
2.20 : Principe de la connexion de deux blocs ayant tous des tranches de bit supplémentaires 69
2.21 : Autre réalisation des $C_{i,j}$ 72
2.22 : Proportion de matériel rajouté pour une redondance modulaire 76
2.23 : Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec une seule retenue de propagée 77
2.24 : Proportion de matériel rajouté pour une reconfiguration par ajout de tranches avec deux retenues de propagées 77

Chapitre 3 : Reconfiguration de la partie contrôle 79

Figures :

3.1 : Structure d'une partie contrôle multi-PLAs 83
3.2 : Reconfiguration d'un registre par ajout de cellules supplémentaires 84
3.3 : Structure de PLA reconfigurable 86
3.4 : Avant reconfiguration des entrées 87
3.5 : Après reconfiguration des entrées 87
3.6 : Avant reconfiguration des monômes 88
3.7 : Après reconfiguration des monômes 89
3.8 : Avant reconfiguration des sorties 89
3.9 : Après reconfiguration des sorties 90

Chapitre 4 : Application au microprocesseur HYETI 93

Figures :

4.1 : Organisation des registres 97
4.2 : Espace d'adressage mémoire 98
4.3 : Architecture de la partie opérative de HYETI 102
4.4 : Cellule de registre simple 104
4.5 : Cellule de registre à décalage mono-directionnel ... 105
4.6 : Registre à décalage bi-directionnel 105
4.7 : Schéma de la logique de bus 106
4.8 : Synchronisation des transferts sur les bus 107
4.9 : Multiplexage des bits du registre d'état vers le contrôleur 108
4.10 : Schéma logique du registre d'état 109
4.11 : Schéma général d'un registre de signature 110
4.12 : Schéma général du dispositif de signature de HYETI 110
4.13 : Schéma logique du registre de signature 111
4.14 : Unité Arithmétique et Logique 113
4.15 : Opérations de l'U.A.L 113
4.16 : Schéma logique de l'U.A.L 115
4.17 : Schéma général de l'ensemble U.A.L - Dispositif de masquage 116

4.18 : Schéma logique d'une tranche de D.M 117
4.19 : Organisation du multiplieur cellulaire 118
4.20 : Cellule de base du multiplieur 118
4.21 : Multiplieur cellulaire et dispositif de détection de débordement 119
4.22 : Multiplieur parallèle - série 121
4.23 : Interface mémoire 122
4.24 : Diagrammes de temporisation de HYETI 123
4.25 : Architecture de la partie contrôle de HSURF 124
4.26 : Architecture du contrôleur de HYETI 125
4.27 : Structure électrique des PLAs de HYETI 126

Résumé

Les possibilités d'intégration sans cesse accrues permettent d'envisager aujourd'hui la réalisation de circuits à l'échelle de la tranche de silicium intégrant en particulier des systèmes à base de microprocesseur sur une seule puce. La principale difficulté tient alors au rendement de fabrication pour ces circuits dont la taille implique qu'ils ne peuvent être exempts de défauts.Ceci met en évidence la nécessité d'une tolérance aux défauts de fin de fabrication. Cette dernière dépend étroitement de la structure du circuit considéré.

L'objet de cette thèse est l'étude de stratégies de tolérance aux défauts de fin de fabrication pour des circuits de type microprocesseur. Le principe de base consiste à partitionner le circuit en blocs où pas plus d'un défaut n'est susceptible d'apparaître et à implanter du matériel de réserve dans chaque bloc. Cela se traduira dans la partie opérative par un ajout de tranches de bit supplémentaires, et dans la partie contrôle par le choix d'une architecture régulière et hiérarchisée (à base de PLAs), la réserve étant alors implantée sous la forme de lignes de PLA programmables.

Les résultats de cette étude ont été appliqués à la réalisation (dans le cadre d'un projet ESPRIT) d'un microprocesseur 16 bits pour des applications à haute sûreté de fonctionnement. Ceci est le point de départ pour l'intégration de systèmes personnalisés à base de microprocesseur sur un seul circuit.

Mots clés

Conception de circuits intégrés, défauts de fabrication, rendement, reconfiguration, redondance, microprocesseur, circuits de grande taille.