Conception d'un microprocesseur reconfigurable

Mohammad Soueidan

THESE

Présentée par

Mohammad SOUEIDAN

Pour obtenir le titre de **DOCTEUR**

de **l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

(arrêté ministériel du 5 juillet 1984)

(Spécialité : Microélectronique)

==========

CONCEPTION D'UN MICROPROCESSEUR RECONFIGURABLE

==========

Date de soutenance : le 14 Avril 1989

Composition du Jury :

Francis	DEVOS	Président
Jean-Pierre	CHARVIN	(Invité)
Francis	JUTAND	Rapporteur
Jean-Louis	LARDY	Rapporteur
Gabrièle	SAUCIER	
Jacques	TRILHE	

Thèse préparée au sein du laboratoire Conception de Systèmes Intégrés

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Georges LESPINARD

Année 1988

Professeurs des Universités

BARIBAUD	Michel	ENSERG
BARRAUD	Alain	ENSIEG
BAUDELET	Bernard	ENSPG
BEAUFILS	Jean-Pierre	ENSEEG
BLIMAN	Samuel	ENSERG
BLOCH	Daniel	ENSPG
BOIS	Philippe	ENSHMG
BONNETAIN	Lucien	ENSEEG
BOUVARD	Maurice	ENSHMG
BRISSONNEAU	Pierre	ENSIEG
BRUNET	Yves	IUFA
CAILLERIE	Denis	ENSHMG
CAVAIGNAC	Jean-François	ENSPG
CHARTIER	Germain	ENSPG
CHENEVIER	Pierre	ENSERG
CHERADAME	Herve	UFR PGP
CHOVET	Alain	ENSERG
COHEN	Joseph	ENSERG
COUMES	André	ENSERG
DARVE	Félix	ENSHMG
DELLA-DORA	Jean-François	ENSIMAG
DEPORTES	Jacques	ENSPG
DESRE	Pierre	ENSEEG
DOLMAZON	Jean-Marc	ENSERG
DURAND	Francis	ENSEEG
DURAND	Jean-Louis	ENSIEG
FOGGIA	Albert	ENSIEG
FONLUPT	Jean	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	Jean-Claude	ENSPG
JOURDAIN	Geneviève	ENSIEG
LACOUME	Jean-Louis	ENSIEG
LESIEUR	Marcel	ENSHMG
LESPINARD	Georges	ENSHMG
LONGEQUEUE	Jean-Pierre	ENSPG
LOUCHET	François	ENSIEG
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
PERRET	René	ENSIEG
PERRET	Robert	ENSIEG
PIAU	Jean-Michel	ENSHMG
POUPOT	Christian	ENSERG
RAMEAU	Jean-Jacques	ENSEEG
RENAUD	Maurice	UFR PGP
ROBERT	André	UFR PGP
ROBERT	François	ENSIMAG
SABONNADIERE	Jean-Claude	ENSIEG
SAUCIER	Gabrielle	ENSIMAG
SCHLENKER	Claire	ENSPG
SCHLENKER	Michel	ENSPG
SERMET	Pierre	ENSERG
SILVY	Jacques	UFR PGP
SIRYES	Pierre	ENSHMG
SOHM	Jean-Claude	ENSEEG
SOLER	Jean-Louis	ENSIMAG
SOUQUET	Jean-Louis	ENSEEG
TROMPETTE	Philippe	ENSHMG
VEILLON	Gérard	ENSIMAG
ZADWORNY	François	ENSERG

Personnes ayant obtenu le diplôme

D'HABILITATION A DIRIGER DES RECHERCHES

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

CHERCHEURS DU C.N.R.S

Directeurs de recherche 1ère Classe

CARRE	René	LANDAU	Ioan
FRUCHART	Robert	VACHAUD	Georges
HOPFINGER	Emile	VERJUS	Jean-Pierre
JORRAND	Philippe		

Directeurs de recherche 2ème Classe

ALEMANY	Antoine	KLEITZ	Michel
ALLIBERT	Colette	KOFMAN	Walter
ALLIBERT	Michel	KAMARINOS	Georges
ANSARA	Ibrahim	LEJEUNE	Gérard
ARMAND	Michel	LE PROVOST	Christian
BERNARD	Claude	MADAR	Roland
BINDER	Gilbert	MERMET	Jean
BONNET	Roland	MICHEL	Jean-Marie
BORNARD	Guy	MUNIER	Jacques
CAILLET	Marcel	PIAU	Monique
CALMET	Jacques	SENATEUR	Jean-Pierre
COURTOIS	Bernard	SIFAKIS	Joseph
DAVID	René	SIMON	Jean-Paul
DRIOLE	Jean	SUERY	Michel
ESCUDIER	Pierre	TEODOSIU	Christian
EUSTATHOPOULOS	Nicolas	VAUCLIN	Michel
GUELIN	Pierre	WACK	Bernard
JOUD	Jean-Charles		

Personnalités agréees à titre permanent à diriger des travaux de recherche (décision du conseil scientifique)

ENSEEG

CHATILLON	Christian	SARRAZIN	Pierre
HAMMOU	Abdelkader	SIMON	Jean-Paul
MARTIN GARIN	Régina		

ENSERG

BOREL	Joseph

ENSIEG

DESCHIZEAUX	Pierre	PERARD	Jacques
GLANGEAUD	François	REINISCH	Raymond

ENSHMG

ROWE	Alain

ENSIMAG

COURTIN	Jacques

EFP

CHARUEL	Robert

C.E.N.G

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

Laboratoires extérieurs :

C.N.E.T

DEVINE	Rodericq	MERCKEL	Gérard
GERBER	Roland	PAULEAU	Yves

****

****

***

UNIVERSITE SCIENTIFIQUE TECHNOLOGIQUE ET MEDICALE
DE GRENOBLE

Président de l'Université :

M. PAYAN Jean Jacques

ANNEE UNIVERSITAIRE 1987 - 1988

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ERE CLASSE

ARNAUD Paul	Chimie Organique
ARVIEU Robert	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S.
AURIAULT Jean Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire I.S.N.
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean René	Statistiques - Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean Paul	Mathématiques Pures
BILLET Jean	Géographie
BOEHLER Jean Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire I.S.N.
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean Pierre	Mécanique
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean René	Mathématiques Pure

KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées
LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean Marie	Sciences Nucléaires I.S.N.
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du Solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean Jacques	Mathématiques Pures
PEBAY PEYROULA Jean Claude	Physique
PERRIER Guy	Géophysique
PIERRARD Jean Marie	Mécanique
PIERRE Jean Louis	Chimie Organique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie C.E.R.M.A.V.
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS DE 2EME CLASSE

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BLANCHI Jean Pierre	S.T.A.P.S.
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET Jean François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHIAN Hervé	Géographie
GUITTON Jacques	Chimie
HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean Paul	Biochimie
KERCKHOVE Claude	Géologie
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N
LUCAS Robert	Physique
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
NEMOZ Alain	Thermodynamique C.N.R.S. C.R.T.B.T.
OUDET Bruno	Mathématiques Appliquées
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RICHARD Jean Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean Bernard	Chimie Physique
SARROT REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L'I.U.T. 1

PROFESSEURS DE 1ERE CLASSE

BUISSON Roger	Physique I.U.T. 1
DODU Jacques	Mécanique Appliquée I.U.T. 1
NEGRE Robert	Génie Civil I.U.T. 1
NOUGARET Marcel	Automatique I.U.T. 1
PERARD Jacques	E.E.A. I.U.T. 1

PROFESSEURS DE 2EME CLASSE

BOUTHINON Michel	E.E.A. I.U.T. 1
CHAMBON René	Génie Mécanique I.U.T. 1.
CHEHIKIAN Alain	E.E.A. I.U.T. 1
CHENAVAS Jean	Physique I.U.T. 1
CHOUTEAU Gérard	Physique I.U.T. 1
CONTE René	Physique I.U.T. 1
GOSSE Jean Pierre	E.E.A. I.U.T. 1
GROS Yves	Physique I.U.T. 1
KUHN Gérard, détaché	Physique I.U.T. 1
MAZUER Jean	Physique I.U.T. 1
MICHOULIER Jean	Physique I.U.T. 1
MONLLOR Christian	E.E.A. I.U.T. 1
PEFFEN René	Métallurgie I.U.T. 1
PERRAUD Robert	Chimie I.U.T. 1
PIERRE Gérard	Chimie I.U.T. 1
TERRIEZ Jean Michel	Génie Mécanique I.U.T. 1
TOUZAIN Philippe	Chimie I.U.T. 1
VINCENDON Marc	Chimie I.U.T. 1

PROFESSEURS DE PHARMACIE

AGNIUS DELORD Claudine	Physique	Faculté la Tronc
ALARY Josette	Chimie Analytique	Faculté la Tronc
BERIEL Hélène	Physiologie et Pharmacologie	Faculté la Tronc
CUSSAC Max	Chimie Therapeutique	Faculté la Tronc
DEMENGE Pierre	Pharmacodynamie	Faculté la Tronc
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galenique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté la Tronc
LUU DUC Cuong	Chimie Générale	Faculté la Tronc
MARIOTTE Anne-Marie	Pharmacognosie	Faculté la Tronc
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté la Tronc
ROCHAT Jacques	Hygiène et Hydrologie	Faculté la Tronc
SEIGLE MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice (Mme Hepner)	Pharmacie galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXCEPTIONNELLE ET 1ERE CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatre - Puériculture	C.H.R.G.
BEZES Henri	Orthopédie - Traumatologie	Hopital Sud
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté la Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie - Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital Sud
COUDERC Pierre	Anatomie Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté la Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique/Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie - Virologie	Faculté la Merci
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale/Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté la Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anesthésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté la Merci
VIGNAIS Pierre	Biochimie	Faculté la Merci

PROFESSEURS 2EME CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté la Merci
BENSA Jean Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie - Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	Abidjan
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean Paul	Anatomie - Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté la Merci
CONTAMIN Charles	Chirurgie Thoracique/Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Nephrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques/Informatique Médicale	Faculté la Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté la Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO ALain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté la Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie - Cytogénétique	Faculté la Merci
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirugie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christan	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie-Obstétrique	Hopital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean Marie	Bactériologie - Virologie	Faculté la Merci
SELE Bernard	Cytogénétique	Faculté la Merci
SOTTO Jean Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

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 Francis DEVOS, professeur à l'IEF (Paris), pour m'avoir fait l'honneur d'accepter de présider le jury de cette thèse,

Monsieur Francis JUTAND, professeur à l'ENS des télécommunications (Paris), d'avoir accepté d'être rapporteur de cette thèse,

Monsieur Jean-Louis LARDY, responsable de l'équipe de conception au CNET (Grenoble), d'avoir accepté d'être également rapporteur de cette thèse,

Monsieur Jean-Pierre CHARVIN, responsable du service "conception de microprocesseurs" à SGS-Thomson, d'avoir accepté de faire partie de ce jury,

Monsieur Jacques TRILHE, docteur ingénieur à la direction technique de SGS-Thomson Microelectronics et responsable du projet Esprit 824-WSI, d'avoir accepté de faire partie de ce jury.

Je remercie également tous mes collègues du laboratoire Conception de Systèmes Intégrés pour l'agréable ambiance de travail, et tout particulièrement Régis LEVEUGLE pour son étroite collaboration.

Au CERS (Centre d'Etudes et de Recherches Scientifiques) Syrie

A mes parents

A mon fils Amr

Table
des matières

Sommaire

CONCEPTION D'UN MICROPROCESSEUR RECONFIGURABLE

Introduction générale

L'augmentation de l'intégration des circuits intégrés a pour but la diminution du coût et l'augmentation de la fiabilité. Ceci peut se faire par la diminution de la taille du transistor élémentaire et la croissance de la dimension des puces.

Dans le cas du transistor MOS, la diminution de la taille est limitée par la valeur minimale de la charge stockée dans la grille qui doit rester supérieure au bruit dû à l'agitation thermique des électrons [TRI88].

La taille de la puce est limitée par le rendement de fabrication, pour une technologie donnée, à une époque donnée, pour une ligne de production donnée. Cette limite peut être repoussée si l'on prévoit dans le circuit des éléments redondants qui peuvent remplacer les éléments défectueux. Si la redondance permettait de corriger tous les défauts de fabrication, la taille de la puce pourrait alors, atteindre la taille de la tranche entière, c'est ce que l'on appelle WSI, des initiales de la dénominations anglaise : "Wafer Scale Integration". Ainsi une tranche de 100 mm de diamètre en technologie CMOS 1,2 µm peut contenir 25 millions de transistors effectifs.

Même si le WSI est une limite qui ne sera jamais utilisé pour la production de volume, pour des raisons de coût par exemple, l'augmentation de l'intégration d'un système électronique est un but fondamental pour l'industrie électronique, car elle aporte à moyen terme une diminution du coût et une augmentation de la compacité d'où meilleurs performances et fiabilité.

Ainsi si l'on est capable de remplacer un système composé de plusieurs cartes imprimées, chacune comportant plusieurs circuits intégrés, par un seul composant on n'aura plus qu'un seul boîtier (certes plus gros) et on limitera considérablement le nombre de plots d'entrée sortie. Or le prix d'un composant est pour moitié le prix du boîtier : on peut donc espérer

limiter les coûts de fabrication d'un système par augmentation de l'intégration. De plus, les problèmes de fiabilité des systèmes électroniques viennent essentiellement des soudures permettant de connecter la puce à son boîtier. L'augmentation de l'intégration est donc une nécessité pour les systèmes requérant une fiabilité élevée.

Notre travail, dans le cadre du projet ESPRIT 824, porte sur l'étude de l'augmentation de l'intégration par correction des défauts de fabrication. L'étude de la diminution des dimensions du transistor ne sera pas abordée.

Nous allons maintenant brièvement situer le sujet par rapport aux travaux déjà réalisés à ce jour.

On peut considérer deux types de circuits : les circuits répétitifs et les circuits non répétitifs. Ils correspondent à deux cas très différents quant à la tolérance aux défauts de fabrication.

C'est aux circuits répétitifs qu'ont principalement été appliquées les techniques de reconfiguration. Ainsi toutes les mémoires de grande capacité, réalisées en production de volume, ont aujourd'hui des lignes et/ou colonnes de redondance.

Une augmentation de la taille de circuits répétitifs jusqu'au WSI, a même vu le jour en recherche, dans plusieurs domaines. Citons le cas d'une RAM [FUC88] [NAS88] et de matrices 2D (2 dimensions) de processeurs élémentaires. Deux niveaux de hiérarchies pour la redondance sont nécessaires. Pour chacun, des éléments de réserve permettent de remplacer les éléments défectueux [SAM86] [LEI86].

Dans le cas d'un circuit non répétitif, il devient très difficile de prévoir des éléments de réserve sans avoir recours à la redondance massive (duplication, triplication). Cette dernière approche est valable au niveau système car elle apporte en plus une tolérance aux erreurs "soft" de points de stockage DRAM par exemple. Mais recourir à la redondance massive dans

un même circuit intégré entraîne irrédiablement un effet pervers de baisse de rendement du fait de l'augmentation conséquente de la surface ou du nombre de transistors dessinés.

Aussi très peu de travaux ont touché à la correction de défaut de circuits non répétitifs. Citons [TEE87] qui traîte de l'implantation d'un corrélateur avec redondance en technologie bipolaire, en fait seule la reconfiguration des accumulateurs est considérée.

Par contre de nombreux travaux concernent des domaines de recherche voisins : l'optimisation de la méthodologie de conception par exemple [MAY88] de façon à augmenter le rendement dans une certaine mesure, sans avoir recours à la redondance, l'intégration d'auto-test [GEL87] [KUB84] pour permettre le diagnostic préalable à la reconfiguration, des études de rendement, pour permettre un partitionnement du circuit en blocs de taille optimale [SUM86].

Dans cette thèse, nous présenterons la conception d'un microprocesseur reconfigurable. Il s'agit du microprocesseur HYETI2 pour "High Yield and Error Tolerant Integration" réalisé et fabriqué en technologie HCMOS3 1,2 µm dans le cadre du projet ESPRIT 824. Ce microprocesseur est destiné aux applications de contrôle temps réel : une première version de ce microprocesseur HYETI1 [GEN87] s'est traduit par un composant de dimensions excessives, et une deuxième conception en a modifié le plan de masse.

Le chapitre 1 présente les spécifications fonctionnelles de la dernière version du microprocesseur (HSURF), l'architecture interne, la méthode et les outils CAO employés pour la conception du microprocesseur.

Le chapitre 2 est consacré à la reconfiguration du HSURF donnant naissance au microprocesseur HYETI. Cette reconfiguration a pour but l'augmentation du rendement de fabrication, une fois que le

microprocesseur fera partie d'un système complet intégré sur tranche.

L'accent dans cette thèse a été mis sur la conception de la partie opérative. En particulier, les principales architectures d'UAL seront étudiées dans le chapitre 3 afin de déterminer les bonnes structures candidates pour la reconfiguration.

Enfin, dans le chapitre 4, une étude détaillée porte sur les circuits d'analyse de signature introduits dans le microprocesseur afin d'en faire un dispositif à test en ligne intégré. Dans cette approche, le problème de la testabilité est un point important car il s'agit de reconfigurer le microprocesseur à bon escient.

Chapitre 1 :
un microprocesseur
à des applications spécifiques

1.Introduction

Le composant présenté, appelé HSURF (Haute SUReté de Fonctionnement), est un coeur de microprocesseur dédié principalement à des applications du type automatisme logique nécessitant une haute sûreté de fonctionnement (cf. [JAY86] pour la première version du microprocesseur HSURF). Ce composant est en fait adapté, suivant son usage, à des applications avec des exigences moyennes de sécurité (robotique, automobile ...) ou à des applications à forte exigence sécuritaire et mettant directement en jeu des vies humaines (automatisation des transports terrestres, par exemple). Le jeu d'instructions est constitué par certaines instructions des microprocesseurs usuels, complétées par des instructions spécialement adaptées au domaine considéré.

L'aspect sécuritaire est pris en compte par des facilités de test en ligne et d'auto-surveillance intégrées sur le silicium. Ainsi, le microprocesseur vérifie en permanence la validité des instructions qu'il reçoit et une surveillance de plus haut niveau est également assurée : un dispositif de compaction de données par division polynômiale avec un polynôme diviseur programmable est implanté sur silicium. Ceci permet de compacter des séquences de données telles que, par exemple, la suite des codes opération et d'obtenir des signatures de référence pouvant être pré-calculées en différents points du programme. Rappelons que les programmes d'application dans le domaine visé consistent souvent à répéter cycliquement un certain nombre d'opérations, ce qui se prête particulièrement bien à l'utilisation de références pré-stockées. En ce qui concerne les données, un format de 16 bits a été déclaré suffisant pour assurer un codage sécuritaire [SAU84]. Pour les applications à sécurité moyenne, le microprocesseur a la faculté de s'auto-surveiller de façon

autonome par comparaison interne des valeurs de référence. Pour les applications à haute sécurité, une référence et une comparaison externes sont nécessaires. Ceci peut être réalisé par une redondance massive ou par la comparaison avec des résultats pré-calculés et pré-stockés. Dans ce dernier cas, le comparateur externe est du type "à sécurité intrinsèque" [SAU84].

En ce qui concerne la structure interne, l'architecture régulière choisie pondère la notion habituelle d'optimisation surface/performances par la notion de conception sûre, réduisant de façon considérable la probabilité d'erreurs de conception rémanentes qui est un problème majeur pour l'utilisation des microprocesseurs du commerce dans les applications sécuritaires. De plus, la structure matérielle a été conçue pour assurer une observabilité complète de tous les blocs du circuit. Ceci entraîne une possibilité de validation exceptionnelle en fin de conception et permet un test de fin de fabrication et un test hors ligne avec une couverture optimale.

2. Jeu d'instructions et facilités de test en ligne

2.1. Jeu d'instructions et modes d'adressage

Le jeu d'instructions et les modes d'adressage du microprocesseur ont été déterminés en collaboration avec des usagers potentiels (cf. [Annexe 1.1] [Annexe 1.2]). La part classique du jeu d'instructions comprend les opérations arithmétiques et logiques, les décalages et les rotations, ainsi que les instructions de transfert, de comparaison, de manipulation de la pile, de saut, de branchement et de traitement des interruptions.

L'étude de programmes d'applications utilisés dans le domaine des automatismes logiques a par ailleurs montré qu'un grand nombre d'opérations était effectué sur des champs de quelques bits définis à l'intérieur des mots de 16 bits. Cette remarque a conduit à proposer des instructions "masquées", c'est à dire des instructions n'agissant que sur un certain nombre de bits définis par le programmeur à l'aide d'un masque. Ceci est illustré figure 1.1. Le masque souhaité est placé dans un registre interne avant toute opération masquée. Toutes les opérations logiques sont masquables, ainsi que les instructions de lecture ou écriture en mémoire. L'instruction de comparaison est également masquable, l'opération réalisée pour la comparaison est toutefois dans ce cas un OU exclusif et non une soustraction. Deux bits particuliers du registre d'état permettent de plus de savoir si les bits modifiés au cours de l'opération masquée ont tous été mis à 0 ou à 1.

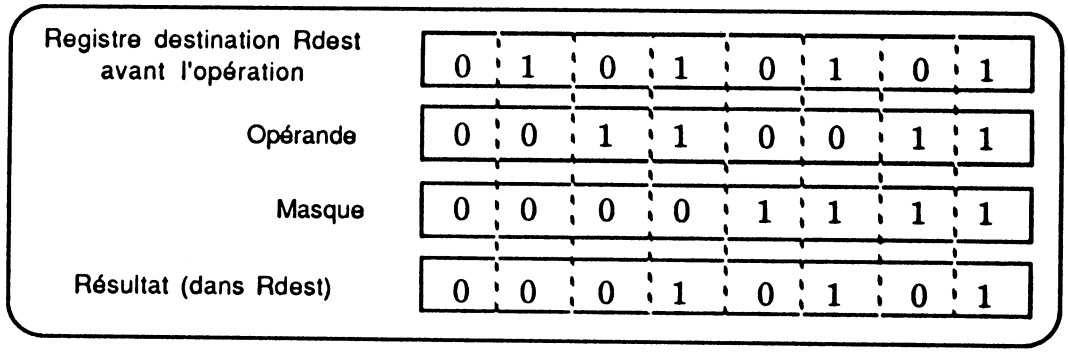


Figure 1.1 : résultat du ET logique masqué. L'opération est réalisée entre le contenu du registre Rdest et l'opérande. Seuls les bits de Rdest correspondant à un 0 dans le masque sont modifiés en fonction du résultat.

Le jeu d'instructions est complété par cinq instructions dédiées au test en ligne et dont l'utilisation est détaillée dans la suite de ce chapitre.

Dans le microprocesseur, une opération est toujours réalisée sur un registre interne (registre destination Rdest). Le numéro du registre

concerné est donné avec le code opération dans le premier mot de l'instruction. Le second opérande, s'il existe, peut avoir quant à lui plusieurs provenances selon le mode d'adressage utilisé parmi les 8 disponibles.

Deux de ces modes ont été spécifiquement introduits pour faciliter la recherche d'éléments dans des matrices. Il s'agit des modes "indirect registre" où le registre d'indirection est modifié automatiquement avant ou après la recherche de l'opérande par ajout d'un pas programmable. Celui-ci est une valeur arithmétique sur 16 bits placée par le programmeur dans un registre interne avant toute utilisation de ces modes d'adressage.

2.2. Les facilités de test en ligne

Le test en ligne d'un système consiste, de manière générale, en des vérifications partielles de la conformité du système testé à son système de référence : ces vérifications ont pour but de détecter au plus tôt, et en cours de fonctionnement normal du système, un certain nombre d'états incorrects [PIL85]. Cette détection consiste à examiner des propriétés qui doivent être vérifiées pour tout état correct du système (mais qui peuvent l'être aussi dans certains états incorrects). Ces propriétés invariantes peuvent se situer à différents niveaux : matériel, informatique ou application. En ce qui concerne le test en ligne du microprocesseur, deux types de propriétés invariantes sont considérés, l'un lié aux possibilités propres du circuit, l'autre lié aux programmes d'application.

2.2.1. Propriété invariante liée aux caractéristiques du processeur

La première propriété invariante prise en compte est liée à la liste des instructions pouvant être employées avec le microprocesseur. Le premier mot de l'instruction indique généralement le mode d'adressage, l'opération à effectuer et une partie au moins des ressources internes du microprocesseur mises en jeu. Or, il existe quelques cas particuliers où le contenu d'un mot mémoire peut ne pas correspondre à une instruction. Par exemple, certains codes opération peuvent ne pas être utilisés, le nombre de codes disponibles étant supérieur au nombre d'opérations à coder (cas des processeurs à jeu d'instructions relativement simple). Il peut également se produire que certaines opérations soient interdites sur certaines ressources, ou avec certains modes d'adressage. Une vérification systématique et automatique des mots mémoire codant les instructions lors de leur chargement dans le registre d'instruction permet de détecter immédiatement toute erreur ayant transformé une instruction valide en une instruction invalide. Il est ainsi possible de s'assurer qu'un code instruction non attribué ne pourra placer le microprocesseur dans un état non prévu par le concepteur, état qui pourrait avoir des conséquences dangereuses.

Une vérification croisée du code opération, du mode d'adressage et des ressources concernées est ainsi réalisée dans le microprocesseur (HSURF). La vérification de validité est prise en compte directement au niveau du graphe de contrôle. La détection d'une instruction erronée conduit à un état particulier qui génère un signal d'alarme.

On remarque que ce type de vérification automatique peut très facilement être implanté. Le coût en silicium est très faible (quelques cas

supplémentaires à considérer dans les équations de séquencement) et aucune pénalité n'est introduite en ce qui concerne les performances. Par ailleurs, une telle vérification est également possible dans un contrôleur pipeline, la détection d'erreur étant alors attachée à l'étage de recherche de l'instruction.

2.2.2. Test en ligne à travers les programmes d'application

Le test en ligne présenté en 2.2.1. permet de vérifier une propriété invariante attachée aux caractéristiques du processeur. Une erreur transformant une instruction valide en une instruction invalide est immédiatement détectée. Cependant, la connaissance des caractéristiques du microprocesseur ne suffit pas pour détecter une erreur transformant une instruction valide en une instruction valide différente. Une telle transformation nécessite, pour être détectée, la vérification d'une autre propriété invariante de plus haut niveau. Cette dernière doit dépendre du traitement souhaité, c'est à dire du programme d'application exécuté.

La méthode proposée est fondée sur les principes de justification par signature [SRI82a], [SHE83]. La stratégie de test consiste alors à compacter les codes des instructions exécutées par le processeur par l'intermédiaire d'une division polynômiale. Des dispositifs spéciaux implantés sur le silicium permettent d'effectuer cette compaction automatiquement sans ralentir le programme d'application.

Associées au dispositif de compaction intégré, trois instructions permettent d'effectuer un test en ligne à travers le programme d'application. La première, RAZ, permet de remettre à zéro le registre de signature. Les deux autres, AJS et AJST, permettent d'ajuster la valeur de la signature obtenue en fonction du chemin suivi à l'intérieur du

programme ; c'est ce que nous allons détailler.

Faisant abstraction des sous-programmes et des exceptions, un programme peut être décomposé en trois structures de base : blocs linéaires, convergences (ou points de jonction) et divergences.

Un bloc linéaire est un segment possédant les caractéristiques suivantes :

- le seul point d'entrée est la première instruction du bloc,
- le seul point de sortie est la dernière instruction du bloc,
- le bloc ne comprend aucun branchement,
- si la première instruction d'un bloc est exécutée alors celui-ci est exécuté dans son entier.

Une divergence est constituée par n'importe quel branchement conditionnel ou par tout branchement inconditionnel terminant un bloc et précédant une jonction. Une divergence est l'interface entre plusieurs blocs linéaires.

Un point de jonction est une instruction faisant suite à au moins deux autres instructions par le jeu de branchements conditionnels ou non. La première instruction d'un bloc linéaire (et seulement celle-ci) peut être un point de jonction. La figure 1.2 illustre ces notions.

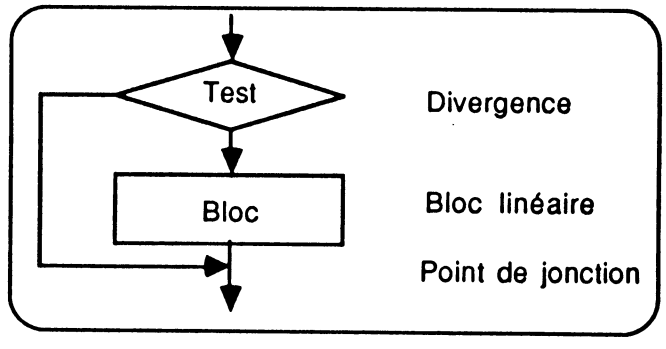


Figure 1.2 : exemple de décomposition d'une structure.

Supposant connue la signature au début d'un bloc linéaire, la signature obtenue en n'importe quel point du bloc est pré-calculable. Un tel bloc est donc parfaitement contrôlable par analyse de signature.

Afin de permettre le pré-calcul des signatures apparaissant au cours de l'exécution d'un programme complet (blocs linéaires, divergences et convergences), il est suffisant d'assurer l'invariance de la signature obtenue à chaque point de convergence. Pour cela, il suffit d'ajuster la signature à la fin de chaque bloc précédant un point de jonction pour obtenir la même valeur à l'issue de chacun d'eux. Remarquons que l'on peut s'abstenir d'introduire cet ajustement dans l'un des blocs, pourvu qu'il soit pris en référence pour le calcul de la signature à obtenir au point de jonction considéré.

La figure 1.3 reprend la structure décrite en figure 1.2 en y insérant l'ajustement nécessaire.

Si l'instruction Test de branchement conditionnel provoque le saut du Bloc, alors Signature_1 est seulement modifiée par cette instruction pour donner Signature_3. En revanche, si le branchement n'a pas lieu, Signature_1 devient Signature_2 lors de l'exécution du Test et du Bloc. Le rôle de l'instruction AJS, placée entre le Bloc et le point de jonction, est donc de transformer Signature_2 en Signature_3. Dans le cas du microprocesseur, l'ajustement consiste à effectuer l'opération OU Exclusif entre la signature courante et le paramètre de AJS. Dans l'exemple de la figure 1.3, on a donc : Val = (Signature_2 XOR Signature_3).

On peut vérifier que l'utilisation d'ajustements de signature avant chaque point de jonction permet bien la signature des structures de programmation classiques. La figure 1.4 illustre ceci pour les très classiques 'If... Then ... Else ...', 'Repeat ... Until ...' et 'While ... Do ...'. Les

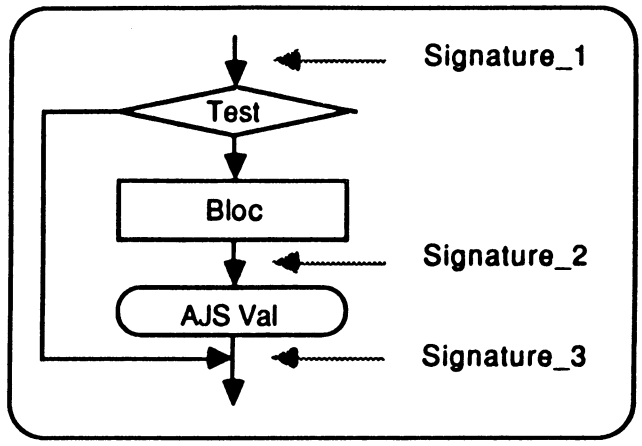


Figure1.3 : modification de la structure
(ajustement de signature au point de jonction)

structures de boucle (Repeat... Until... - figure 1.4(b) - et While... Do... - figure 1.4(c) -) soulèvent plusieurs remarques. Tout d'abord, on note que l'ajustement est certes placé avant le point de jonction de la structure, mais qu'il précède aussi le branchement conditionnel (figure 1.4(b)) ou non (figure 1.4(c)). Ceci est obligatoire puisqu'il est impossible d'insérer une instruction entre un branchement et sa destination. Il est donc nécessaire, lors du calcul de l'ajustement, de tenir compte de l'influence du branchement qui vient se glisser entre l'instruction AJS et le point de jonction. Des calculs, qui ne seront pas développés ici, montrent que la valeur de l'ajustement est toujours calculable, si le terme constant du polynôme générateur du diviseur polynômial est non nul. En revanche, dans le cas où ce terme constant est nul, l'ajustement ne peut être calculé que si le bit de poids faible de la signature au point de jonction est identique au bit de poids faible du code de l'instruction de branchement. Cette dernière condition peut être remplie en insérant un ajustement avant la jonction afin de donner une valeur correcte à la signature en ce point.

Au delà des branchements traités, grâce à l'adjonction d'ajustements avant chaque point de jonction du programme, il faut considérer deux cas

particuliers : les sous-programmes et les exceptions (comprenant les interruptions et les trappes).

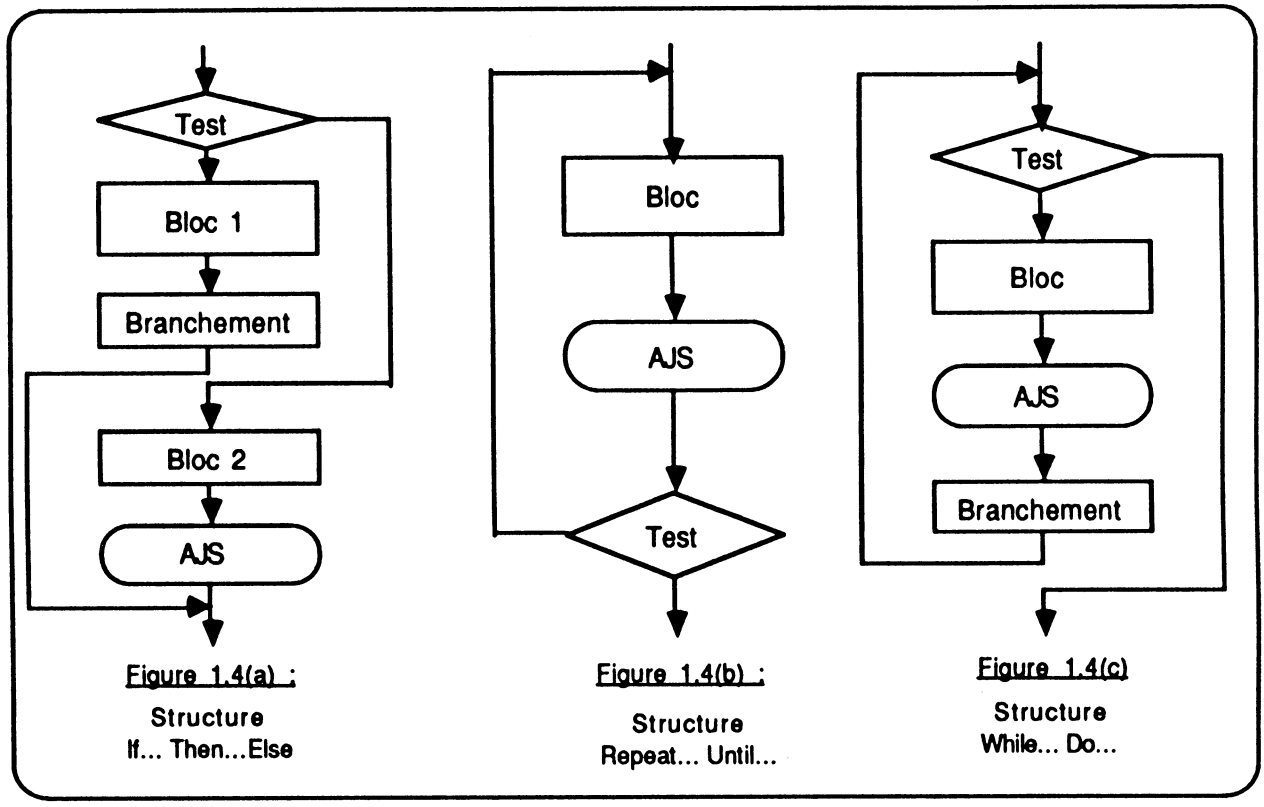


Figure 1.4 : différentes structures.

Le point d'entrée d'un sous-programme peut être considéré comme un point de jonction quelconque. Le branchement à un sous-programme est alors une divergence et marque la fin d'un bloc linéaire.

Il est donc nécessaire de prévoir un ajustement de la signature avant chaque appel de sous-programme afin que, quelle que soit l'origine de l'appel, la signature présente à son entrée soit parfaitement définie et constante. La signature des instructions composant le sous-programme est alors possible. Lorsque le sous-programme se termine, il doit lui aussi fournir une signature fixée pour permettre la continuation de la signature des instructions du programme appelant.

Il faut remarquer qu'un sous-programme ne doit pas modifier l'adresse de retour qui lui a été fournie par le programme appelant. Dans le cas contraire, la prévision, et donc le contrôle, des signatures deviendrait impossible au-delà de la fin du sous-programme.

Le départ en exception est, en beaucoup de points, semblable à un appel de sous-programme. La différence essentielle, en ce qui concerne l'analyse de signature, est l'impossibilité de prévoir un tel départ. Il est donc impossible d'effectuer un pré-calcul de signature tenant compte des exceptions : l'ajustement est donc inutilisable dans ce cas.

Une solution possible à ce problème consiste à utiliser une pile de signatures. Lorsque l'occurrence d'une exception est détectée, la signature courante est empilée et un nouveau calcul initialisé. Le programme de traitement de l'exception peut alors être signé normalement. A la fin de ce traitement, la signature du programme interrompu est dépilée; le calcul peut alors reprendre.

De même que pour les sous-programmes, il est nécessaire que le programme de traitement d'exception ne modifie pas l'adresse de retour mémorisée au moment de son occurrence.

Il faut donc insister sur le fait qu'une programmation structurée est absolument nécessaire pour prétendre utiliser le contrôle par analyse de signature. Mise à part cette contrainte, on remarque que les sous-programmes s'inscrivent dans le cas général, et que seules les exceptions demandent un traitement bien particulier. Par ailleurs, l'insertion des ajustements étant entièrement régie par un ensemble de règles, il est possible de modifier l'assembleur afin que celui-ci se charge de ce travail. L'un des paramètres d'assemblage pourrait être, par exemple, le nombre maximum d'instructions entre deux contrôles de signature permettant ainsi de fixer un niveau de sécurité. L'utilisation d'un langage

de haut niveau reste en accord avec les principes énoncés ci-dessus puisque la plupart des compilateurs génèrent un code assembleur intermédiaire. De plus, ce code est généralement structuré.

Pour les applications à haute sécurité, les signatures obtenues en certains points du programme peuvent être écrites vers l'extérieur et comparées avec les valeurs de référence stockées dans une mémoire externe. Pour les applications à moyenne sécurité, le composant peut effectuer une vérification autonome grâce à l'instruction AJST. Cette dernière instruction effectue la même opération que AJS, mais teste de plus la valeur de la signature obtenue après ajustement. Si le résultat est non nul (signature calculée différente de la valeur de référence donnée dans AJST), un signal d'alarme est généré.

Il faut remarquer que le polynôme utilisé pour la division est programmable par l'utilisateur. Par ailleurs, le type des données signées est également programmable. L'utilisateur peut ainsi signer, s'il le désire, des données ou des adresses à la place des codes d'instructions. Une fois indiqués le polynôme et le type des données à signer, la signature s'effectue automatiquement chaque fois que des données du type voulu transitent sur les bus internes du microprocesseur. Le choix du type des données signées doit être effectué en fonction de la stratégie de test en ligne choisie. Dans la stratégie présentée ci-dessus, seule la compaction des codes instructions est utilisée. Les données peuvent alors être protégées par codage. En effet, dans beaucoup d'applications, les 16 bits d'un mot sont suffisants pour consacrer quelques bits au codage de l'information. Toutefois, le microprocesseur a été conçu pour s'adapter facilement à toute stratégie définie par l'utilisateur.

Ces vérifications de propriétés invariantes peuvent être complétées par un test périodique des ressources du microprocesseur. En effet, dans le

domaine d'application visé, il arrive assez fréquemment que le processeur ne travaille pas (attente d'un signal externe, par exemple). Il est possible de profiter de ces temps morts pour s'assurer du bon fonctionnement des divers éléments fonctionnels. Ce test en oisiveté est favorisé dans notre processeur par les instructions LRS et SRS qui autorisent l'accès par programmation en lecture ou en écriture à l'ensemble des registres du microprocesseur, y compris les registres tampons internes auquels il n'est pas possible d'accéder avec les instructions classiques. Un tel test en oisiveté réduit encore la période de latence pour la détection des erreurs.

L'évaluation d'efficacité du test en ligne ne sera pas discutée dans ce chapitre, car elle dépend fortement de l'application considérée et de l'implantation du test proprement dit. En fait, l'évaluation d'efficacité d'un test en ligne ne peut en terme de sécurité être faite que par rapport à un objectif fixé, dépendant uniquement de l'application et des configurations potentiellement dangereuses de celle-ci. La stratégie suivie doit alors pouvoir s'adapter précisément au problème à résoudre ; c'est pourquoi une importante souplesse d'utilisation a été recherchée pour ce microprocesseur.

3. Architecture interne du microprocesseur

3.1. Une structure régulière

L'architecture générale du microprocesseur, donnée en figure 1.5, a été choisie la plus régulière possible, afin d'assurer un test hors ligne particulièrement efficace. Le circuit peut être décomposé, classiquement, en une partie opérative et un contrôleur. L'interface entre ces deux parties est réalisée à l'aide de registres utilisables en "scan-path". Nous

reviendrons sur leur utilisation par la suite.

En ce qui concerne la partie opérative, la régularité a été assurée par l'utilisation d'une structure en tranches de bits "bit-slice". Le chemin de données est organisé autour de deux bus selon le schéma de la figure 1.6. Il est composé de 16 registres accessibles avec la plupart des instructions, dont 9 registres généraux R0 à R8, un registre de masque RMA et un registre de pas RPA pour les modes d'adressage auto-modifiés. Les registres à vocation spécifique peuvent être utilisés comme des registres généraux lorsque leur spécificité n'est pas employée. A ces éléments viennent s'ajouter, outre l'Unité Arithmétique et Logique, le dispositif de masquage DM, le dispositif d'autorisation de signature AS, le LFSR (Linear Feedback Shift Register) à entrées parallèles SIGN servant à la division polynômiale et les deux registres associés POL et RTIS, le registre d'état RE, le compteur ordinal CO et 5 registres internes. L'ensemble des registres est accessible avec les instructions LRS et SRS.

En ce qui concerne le contrôleur, une première étude avait été menée à partir d'une architecture microprogrammée. En effet, si les divers éléments utilisés (micro-compteur ordinal, ROM, décodeurs ...) sont en eux-même réguliers, cette régularité disparaît pour le contrôleur considéré globalement. La partie contrôle du microprocesseur est donc actuellement composée de deux étages de PLAs, le second correspondant à des décodeurs. Une description plus précise du contrôleur et de la méthode de synthèse mise au point pour sa génération, peut être trouvée dans [LEV88b].

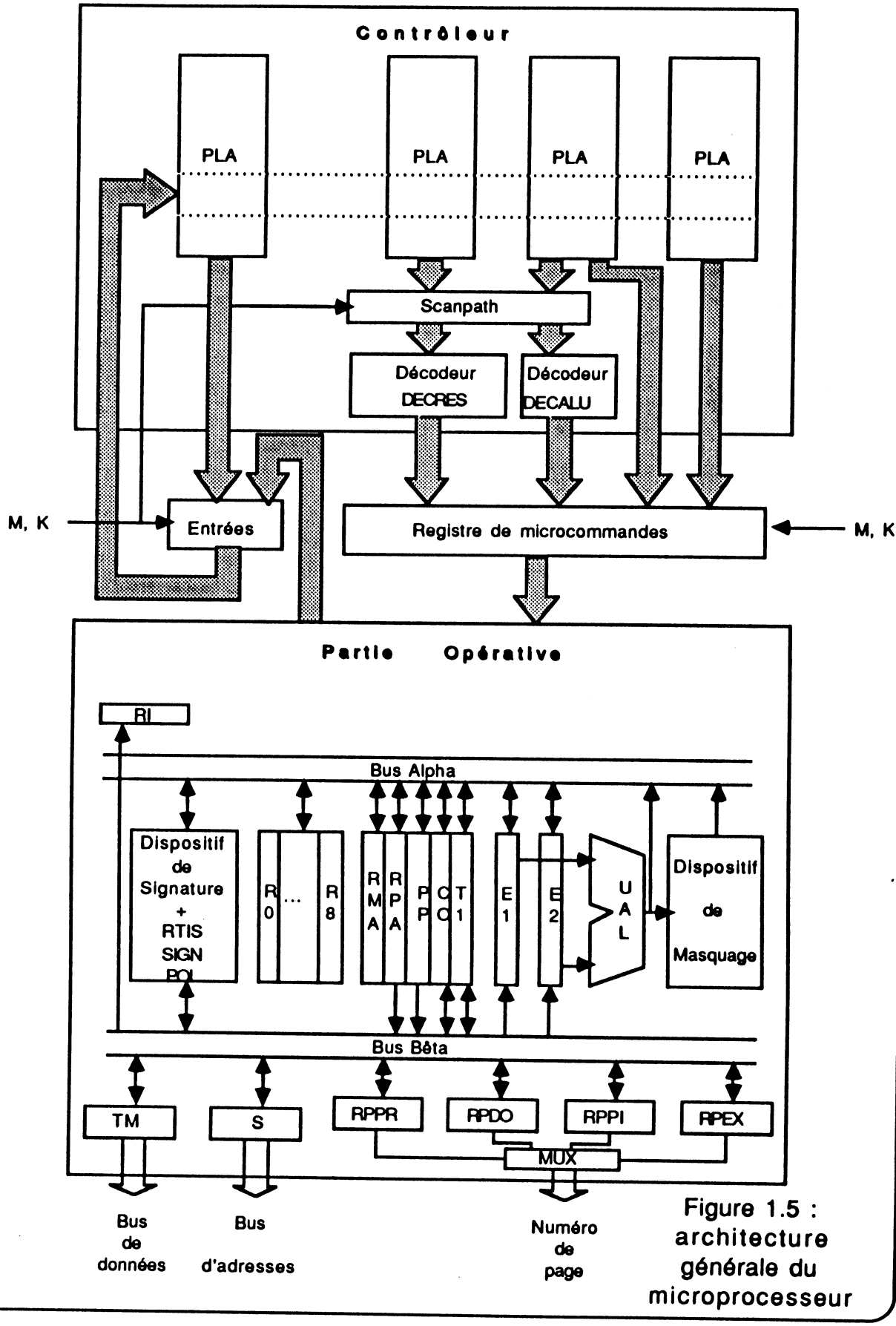


Figure 1.5 : architecture générale du microprocesseur

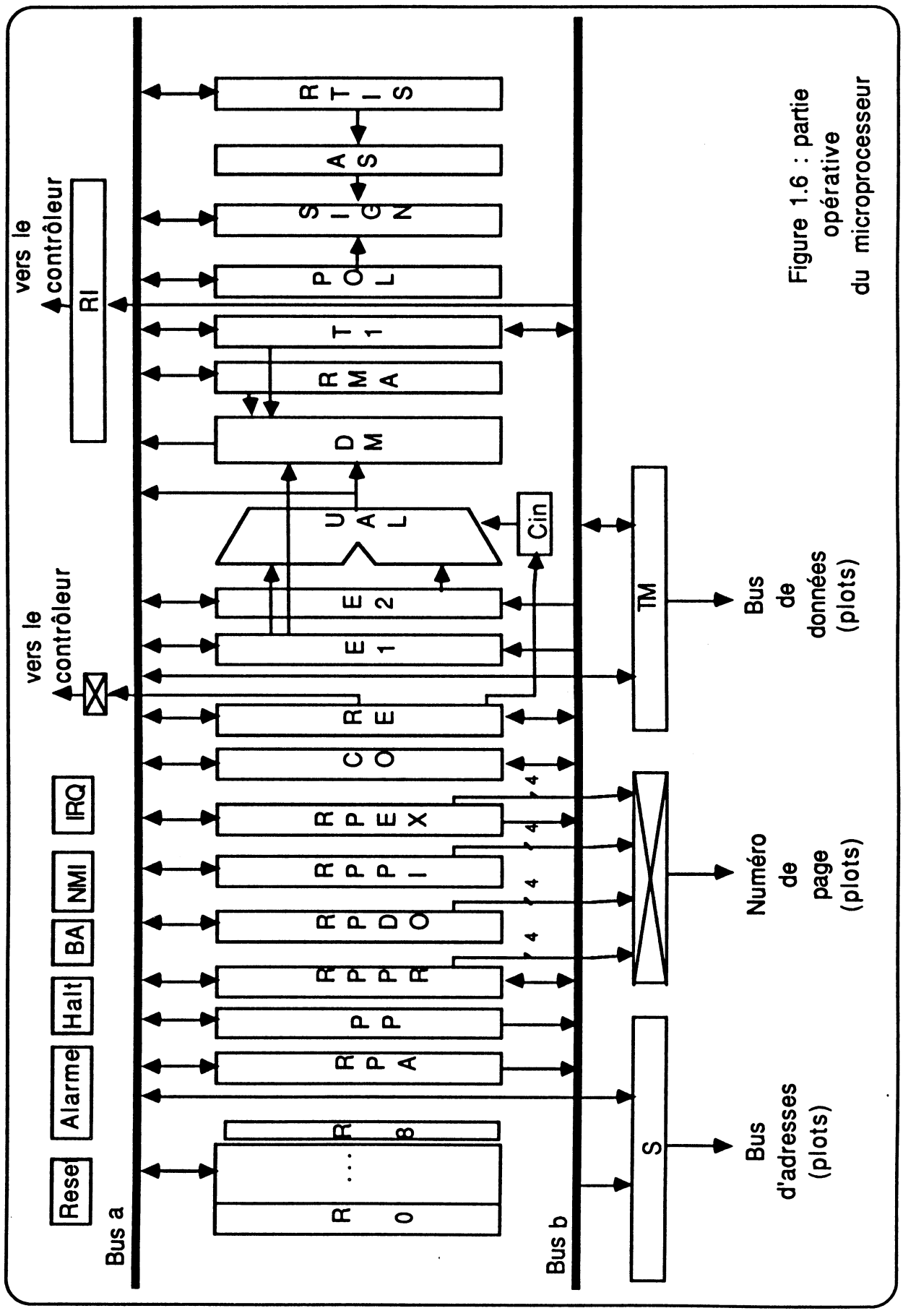


Figure 1.6 : partie opérative du microprocesseur

3.2. Observabilité et testabilité des divers éléments du microprocesseur

Comme indiqué précédemment, la structure du circuit a été conçue pour assurer une observabilité maximale. Tout d'abord, l'interface entre le contrôleur et la partie opérative est constituée de deux registres pouvant être utilisés comme "scan-path" : le registre d'entrée du contrôleur et le registre de microcommandes. En fonctionnement normal, le premier assure la connexion directe des signaux d'entrée du contrôleur aux entrées des PLAs du premier étage, tandis que le second fonctionne comme un registre maître/esclave. Lorsque le circuit est sous test, chacun de ces registres peut être chargé en parallèle ou sériellement, et leur contenu peut être lu sériellement. Les différents modes sont obtenus par des signaux de commande externes M et K. Ceci permet de tester indépendamment et précisément le contrôleur et la partie opérative.

Grâce à ces registres, il est possible d'accéder directement dans le contrôleur aux entrées des PLAs du premier étage et aux sorties des décodeurs. Afin de pouvoir effectuer un test exhaustif, un "scan-path" a de plus été rajouté à l'intérieur de la partie contrôle entre les deux étages. Ce "scan-path", qui est transparent en fonctionnement normal, permet un accès direct aux sorties des PLAs du premier étage et aux entrées des décodeurs. Une observabilité complète des divers éléments du contrôleur est ainsi assurée.

Par ailleurs, la conception des blocs de la partie opérative a pris en compte dès le début les problèmes liés à la testabilité. Ainsi, l'Unité Arithmétique et Logique implantée a une structure I.L.A. C-testable (Iterative Logic Array), la cellule élémentaire de la tranche de 1 bit ayant été conçue spécialement dans ce but. Ceci permet d'effectuer un test

rapide et exhaustif de l'U.A.L.

L'ensemble de ces caractéristiques fait du microprocesseur un composant pouvant être testé avec une couverture de pannes exceptionnelle, tout en conservant une durée de test hors ligne raisonnable (de l'ordre de la seconde).

4.Exemples d'emploi du microprocesseur

Plusieurs exemples d'application ont été étudiés au cours de la définition des caractéristiques du microprocesseur. Ces applications se situent dans des domaines où la sécurité est plus ou moins sévère.

Un exemple d'utilisation nécessitant une sécurité moyenne peut être trouvé dans le domaine automobile. Des automatismes de plus en plus nombreux, bâtis autour d'une carte à microprocesseur, sont proposés pour assister le conducteur. De manière générale, ces dispositifs ne sont pas critiques et un mauvais fonctionnement ne se répercute pas directement sur la sécurité des passagers du véhicule. Toutefois, il est important que le conducteur puisse être averti le plus rapidement possible d'une panne éventuelle afin de prendre les dispositions qui s'imposent. La détection d'un mauvais fonctionnement doit donc être assurée, mais son coût doit être minimisé. L'emploi d'un composant comme ce microprocesseur permet alors d'assurer la détection de certains problèmes à moindre coût que la réplication.

Lorsque l'automatisme contrôle directement une application mettant en jeu des vies humaines, les problèmes de certification sont beaucoup plus complexes. Une architecture "simplex" [SAU84], même munie de tests en ligne puissants, ne peut alors être retenue. Il devient nécessaire d'utiliser plusieurs systèmes indépendants dont on compare les résultats. Le test en

ligne permet alors de compléter l'efficacité de la détection.

Le poste d'aiguillage informatisé (PAI) présenté dans [SEV84] et [SAU84] est ainsi constitué de deux unités identiques isolées (système "duplex") dont les sorties sont en permanence comparées. Par ailleurs, chaque unité contient des dispositifs internes de détection d'erreurs. Il s'agit donc d'une stratégie de tolérance aux pannes à deux niveaux : le niveau système et le niveau carte. Le microprocesseur utilisé dans chaque unité est un 68000. Il est partiellement testé par l'utilisation d'un détecteur de codes invalides, par l'activation de "chiens de garde" et par l'exécution régulière de séquences de test (test en oisiveté).

L'utilisation de ce microprocesseur dans un tel système en remplacement du microprocesseur à usage général peut apporter diverses améliorations. Tout d'abord, une partie des vérifications effectuées au niveau carte, telles la détection de codes invalides, peut être ramenée au niveau du composant ; les circuits sur la carte peuvent alors être simplifiés et plus facilement testés. Le jeu d'instructions étant bien adapté, l'écriture des programmes est facilitée ainsi que leur validation. Les possibilités offertes pour le test en oisiveté permettent des vérifications, beaucoup plus complètes, du bon fonctionnement du microprocesseur. Enfin, la latence des pannes étant fortement réduite, le risque de double panne (erreur simultanée en sortie des deux unités indépendantes faisant échouer la détection) devient très faible.

La présence d'un signal d'alarme en cas de détection d'erreur permet par ailleurs de localiser l'origine de certaines erreurs. Il est ainsi possible, dans une architecture redondante à vote majoritaire, de ne plus tenir compte des sorties d'un processeur ayant généré une alarme (fonctionnement dégradé).

5. Méthode et outils CAO employés pour la conception du microprocesseur

5.1. Simulation du graphe de contrôle avec le système CADOC

Une fois le jeu d'instructions et les modes d'adressage définis, la partie contrôle a été décrite sous la forme d'un organigramme au niveau "transfert de registres". Ce graphe, qui comporte une centaine d'états, correspond à un automate de Moore avec des commandes paramétrées sur certains états.

Les éléments à implanter dans la partie opérative ont été définis simultanément. Connaissant ainsi la structure du chemin de données, le contrôleur a été décrit précisément en langage CADOC [BEL85], une liste de commandes à générer étant associée à chaque état.

Cette description permet de vérifier le bon fonctionnement du graphe de contrôle dessiné grâce à la génération de chronogrammes indiquant la variation des différents signaux, compte tenu de la temporisation globale du microprocesseur.

5.2. Synthèse du contrôleur avec le système ASYL

Une fois le graphe simulé, la description du contrôleur est utilisée comme entrée du système ASYL [SAU87] [POI88]. Ce système, qui fonctionne à base de règles, permet de générer automatiquement des contrôleurs sous différentes formes (logique aléatoire, portes complexes, PLA, ...).

Dans le cas de la synthèse à base de PLAs, ASYL détermine, à partir de la description de haut niveau fournie par le concepteur, les équations

logiques des sorties et des variables internes de séquencement. Ces dernières sont codées de façon quasi optimale par le système, en fonction de la structure du graphe traité, par recherche des adjacences souhaitables entre états.

Dans le cas où un PLA unique est trop gros pour pouvoir être implanté en respectant les contraintes électriques, un partitionnement peut être effectué en plusieurs PLAs plus petits. Ce partitionnement, qui peut être déterminé par le concepteur, peut également être pris en charge par le système qui cherche alors à dupliquer un nombre minimal de monômes. Il faut noter que plusieurs essais de partitionnement peuvent être faits de manière à trouver celui conduisant au meilleur plan de masse pour le contrôleur. Une première évaluation (grossière) du placement et du routage est alors à prendre en compte.

Une minimisation globale des équations est ensuite effectuée, de manière à réduire le nombre de monômes de chaque PLA, et donc la surface nécessitée par le contrôleur. Cette dernière étape conduit aux matrices de personnalisation des PLAs à implanter dans le premier étage, matrices qui peuvent être transmises directement à un générateur de masques.

Les décodeurs sont réalisés également sous forme de PLA. Toutefois, un décodeur peut nécessiter uniquement une matrice ET ; c'est le cas pour le décodeur DECRES (figure 1.5) du microprocesseur. La génération des matrices des PLAs peut être effectuée à partir d'une description de haut niveau en utilisant le système ASYL. Le générateur de masques doit quant à lui être capable de générer des décodeurs avec le seul étage ET, afin d'éviter la perte de surface occasionnée par l'implantation d'un étage OU n'ayant qu'un transistor par sortie. Par ailleurs, il est utile de disposer de lignes de transparence parallèles aux monômes dans un tel décodeur. Dans le cas du microprocesseur, ces lignes permettent de faire traverser le

second étage de PLAs par les commandes générées directement en sortie du premier étage. Ceci évite de grandes nappes de connexions contournant les décodeurs et permet d'obtenir directement les commandes dans l'ordre souhaité sur l'interface sans croisement coûteux en surface. De plus, la longueur des interconnexions est ainsi réduite. Les capacités à charger étant plus faibles, la taille des amplificateurs de sortie des PLAs peut être diminuée.

5.3. Dessin de la partie opérative avec le système VTI "VLSI Technology Incorporation"

Les diverses cellules de la partie opérative ont été dessinées en utilisant le système VTI [VTI88]. Ces cellules ont été obtenues par l'intermédiaire d'un éditeur "stick-diagram" et d'un compacteur associé. Ce compacteur permet d'obtenir le dessin au micron des masques de la cellule à partir du schéma bâton, décrivant les interconnexions et les positions respectives des éléments, et d'un fichier regroupant les règles de dessin de la technologie utilisée. Le concepteur peut par ailleurs définir des contraintes entre certains éléments pour contrôler le processus de compaction. La visualisation des chemins critiques pour la compaction en X et en Y est également possible et permet souvent au concepteur de réduire la surface de la cellule par modification des positions respectives de certains éléments. Les interventions du concepteur sur le dessin au micron devenant ainsi minimes, cette méthode permet un gain de temps appréciable lors de la phase d'implantation.

Le système VTI a également été utilisé pour l'assemblage hiérarchique des cellules, les simulations au niveau "switch" et les vérifications (extraction, ERC, DRC, comparaison de réseaux avec des schémas

électriques ou logiques). Une interface aisée avec SPICE a permis également des simulations électriques précises.

5.4. Génération des programmes de test

Les programmes de test en fin de fabrication sont obtenus par l'utilisation conjuguée du système CADOC et du système VTI.

Le système CADOC permet d'obtenir les chronogrammes souhaités pour les divers signaux, chronogrammes qui peuvent servir de référence pour la comparaison des résultats obtenus lors du test. Remarquons que l'ensemble du circuit peut être décrit en CADOC, et non seulement le contrôleur. Ainsi, la simulation CADOC peut aider le concepteur pour l'écriture des programmes de test.

Le système VTI possède un outil de génération automatique de programmes de test pour toute une gamme de testeurs. Dans notre cas, le testeur utilisé est un Sentry 7. L'outil VTItest utilise deux types d'entrées : un programme décrivant, dans un langage facilement compréhensible, les opérations à effectuer au cours du test et un fichier de vecteurs de test définis par le concepteur. Le système génère alors automatiquement le programme de contrôle du testeur et le fichier de vecteurs associé, dans le format adapté au testeur utilisé.

6. Conclusion

Les différents problèmes posés par l'utilisation des microprocesseurs du commerce dans les applications sécuritaires ont été étudiés. Les solutions proposées pour faciliter le test hors ligne et permettre un test en ligne puissant font du microprocesseur HSURF un composant bien adapté

à de telles applications.

L'utilisation d'outils CAO puissants a permis une conception sûre. De plus, une modification du jeu d'instructions de ce microprocesseur pourrait s'effectuer facilement et rapidement grâce au système ASYL en particulier. Il est donc possible d'adapter ce microprocesseur à une application bien précise et de l'utiliser, par exemple, comme cœur d'un microcontrôleur.

Chapitre 2 :
conception d'un microprocesseur reconfigurable

1.Stratégie de base [GEN87]

Le but de la stratégie de reconfiguration est la tolérance à un grand nombre de défauts, afin d'accroître le rendement de fabrication et d'utiliser le microproceseur, appelé HYETI, dans un système WSI. Toutefois, tous les défauts ne sont pas tolérables. Par exemple, les lignes d'interconnexion entre le contrôleur et le chemin de données ne sont pas reconfigurables. Quelques éléments du chemin de données ne peuvent être reconfigurés, car cela réclamerait une trop grande augmentation de surface. La surface des éléments redondants est limitée à 25%, et le but est d'obtenir un bon compromis entre l'augmentation de surface et l'amélioration du rendement de fabrication. L'accroissement espéré du rendement est théoriquement estimé de 15% à 20%.

L'analyse des rendements distingue les éléments de circuit qui sont reconfigurables de ceux qui ne le sont pas (pas de redondance). Si Aun représente la surface où un défaut est fatal et Arec la surface où un défaut peut être normalement réparé (quelques lignes de commande n'ont pas de redondance), la probabilité d'obtenir un circuit correct après reconfiguration est :

$Pw = P(X = 0; Aun) \cdot P(X \leq d; Arec) \cdot Prec,$

où X est le nombre de défauts,

$P(X = i, A)$ est la probabilité d'avoir i défauts dans la surface A,

$P(X \leq i, A)$ est la probabilité d'avoir au plus i défauts dans la surface A,

d est le nombre de défauts réparables,

Prec est la probabilité de n'avoir aucun défaut fatal dans les zones de reconfiguration.

Les probabilités sont calculées en utilisant la statistique binômiale négative et les paramètres de la technologie [WHE88].

Dans notre cas, le microprocesseur est divisé en blocs où au moins un défaut peut être corrigé, exceptés les défauts fatals sur les lignes de commande. Le partitionnement respecte la hiérarchie : contrôleur et chemin de données sont considérés séparément.

L'expression du rendement s'écrit alors :

$Pw = P(X = 0, Aun) . \prod P(X \leq 1, Abloci)\ Preci$

puisque chaque bloc i de surface Abloci peut tolérer au moins un défaut avec la probabilité Preci.

Cette expression permet la détermination de la meilleure dimension pour un bloc reconfigurable, compte tenu de la surface estimée du composant et des valeurs estimées de Aun et Arec.

Une estimation de la taille optimale des blocs est donnée dans [GEN87] : 7 à 8 mm^2.

Toutefois, la définition des blocs doit être étroitement associée à l'architecture du circuit. Il est par exemple impossible de considérer deux PLA dans un même bloc. Ils doivent à l'évidence être reconfigurés séparément.

Ainsi, les aspects topologiques et architecturaux doivent être pris en compte lors du partitionnement.

2. Chemin de données

2.1. Reconfiguration

D'une manière générale, le chemin de données d'un microprocesseur est constitué de plusieurs unités fonctionnelles (registres, UAL, logique de précharge des bus, opérateurs spécialisés) qui sont réalisées traditionnellement dans une structure en tranches de bits.

Le partitionnement du chemin de données peut être soit fonctionnel, soit structurel (la tranche est le bloc de base). Le partitionnement fonctionnel permet de reconfigurer séparément les différentes unités fonctionnelles.

Ce type de partitionnement peut être nécessaire quand la structure n'est pas régulière. Le principal inconvénient de cette stratégie réside dans le réseau complexe d'interconnexions entre blocs nécessaire à la reconfiguration. La reconfiguration du chemin de données du microprocesseur utilise dans notre cas la structure en tranches. Une tranche de réserve est disponible et peut remplacer une défectueuse.

Inclure plusieurs tranches de réserve aurait amené un accroissement de surface supérieur à la limite imposée par le projet ESPRIT. Par ailleurs, la surface du chemin de données est d'environ 7 mm^2. C'est-à-dire de l'ordre de la surface optimale déterminée par [GEN87]. Ainsi cette partie du circuit peut être considérée comme un bloc unique. L'accroissement de surface dû à la reconfiguration est de 20 à 25%. La portion non reconfigurable du chemin de données représente environ 20% de la surface totale. Le rendement espéré sans reconfiguration de la partie de traitement de données du circuit est :

$P(X = 0{,}6\ mm^2) \approx 0,67$

Le rendement espéré avec les dispositifs de reconfiguration (pour Preci = 95% d'après la surface des lignes non reconfigurables de la partie reconfigurable) est :

$P(X = 0{,}2\ mm^2).\ P(X \leq 1{,}6\ mm^2).\ 0{,}95 = 0{,}77$

Notez que ce gain de rendement correspond à des hypothèses restrictives. En fait, plusieurs défauts peuvent être corrigés à l'intérieur du chemin de données, s'ils apparaissent dans une même tranche.

2.2. Modifications

Quand un partitionnement structurel est choisi, deux problèmes doivent être résolus : la connexion des bus aux bonnes tranches, et la possibilité pour les signaux se propageant entre les tranches de contourner les tranches défectueuses.

Le premier problème se pose parce que certains éléments ne sont pas reconfigurables. Par exemple, chaque bit du registre d'état a une signification spécifique et est associé à une logique aléatoire donc trop de surface pour reconfigurer.

Ainsi le chemin de données doit être composé de 3 parties comme montré dans la figure 2.1.

Selon les tranches utilisées dans les parties reconfigurables, les bus doivent être reconfigurés de manière à connecter correctement les blocs reconfigurables. Les connexions initiales du bus sont réalisées avec des fusibles laser. Après test, la reconfiguration consiste à couper quelques fusibles et à connecter de nouvelles lignes. Les connexions peuvent utiliser des antifusibles. Cependant, d'un point de vue économique, il est préférable de minimiser le nombre de types de machines de reconfiguration, et par suite le nombre de types d'interrupteurs.

C'est pourquoi les connexions sont réalisées au moyen de portes de transfert. Un signal de reconfiguration Crec est généré pour chaque tranche par le circuit de la figure 2.2.

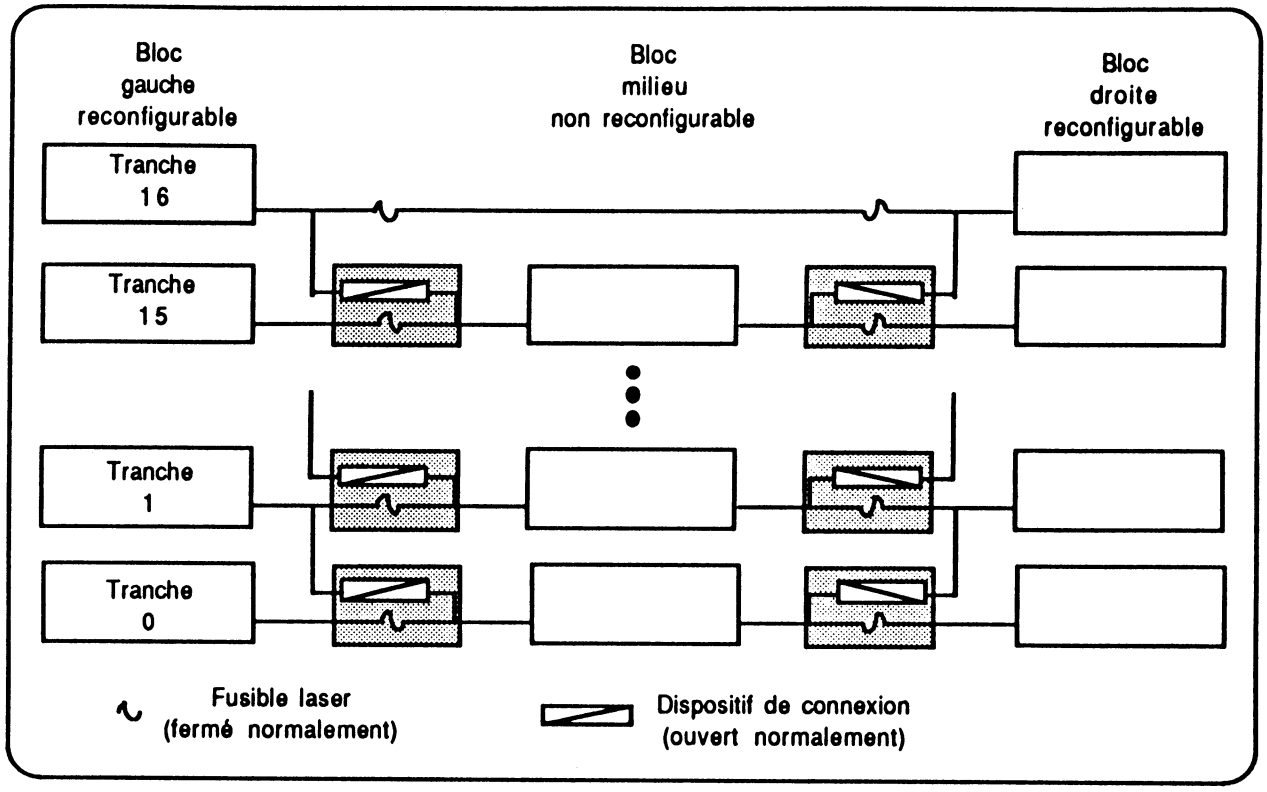


Figure 2.1 : plan général de la partie opérative.

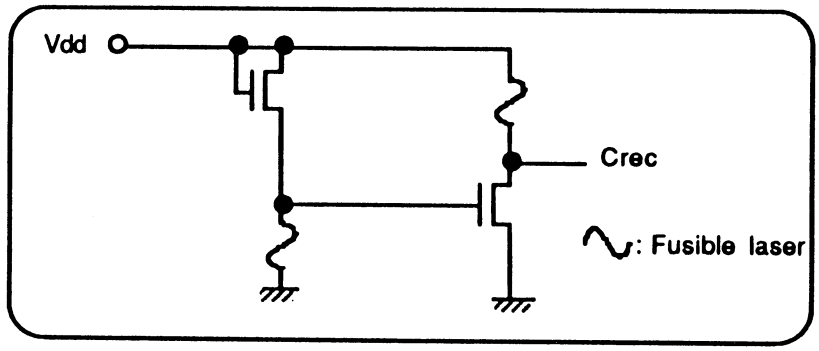


Figure 2.2 : circuit utilisé pour générer le signal Crec.

Ce signal indique si une tranche doit être contournée. Dans la première tranche, Crec commande directement la porte de transfert ; dans les autres tranches (1 à 15), la porte est commandée par une combinaison logique des signaux Crec:

$C_i = Crec_i \cdot C_{i-1}$, $1 \leq i \leq 15$ avec $C_0 = Crec_0$

Le circuit de connexion pour chaque bus est montré en figure 2.3.

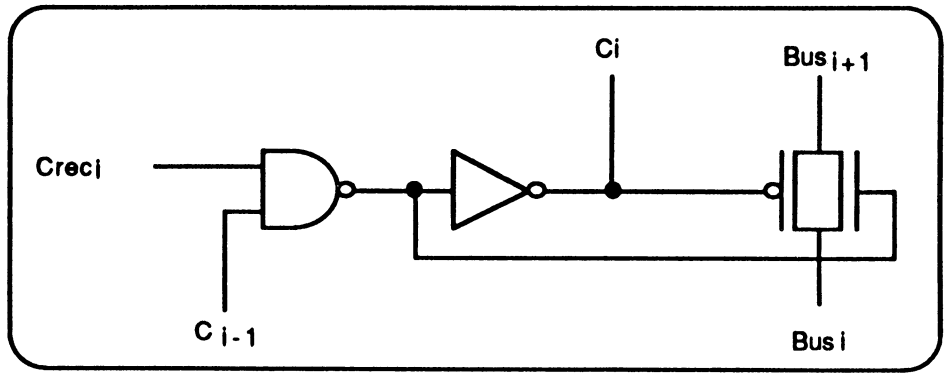


Figure 2.3 : circuit de connexion pour les tranches $1 \leq i \leq 15$.

Les signaux Crec sont aussi utilisés pour propager correctement l'information entre les tranches, selon la figure 2.4.

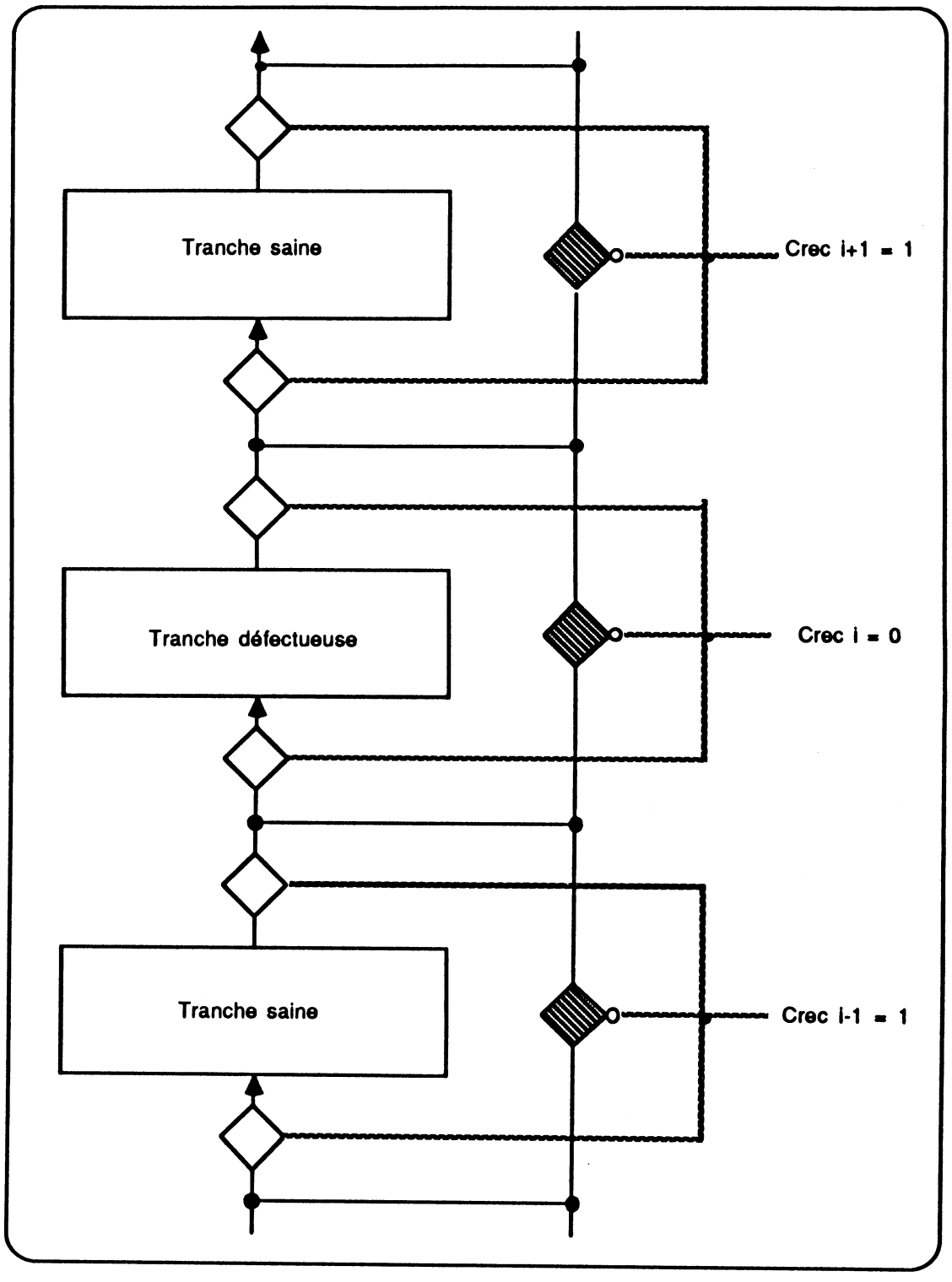


Figure 2.4 : reconfiguration des signaux propagés entre les tranches.

3. Contrôleur

3.1. Reconfiguration

On note une absence globale de régularité dans la plupart des structures de contrôleurs. Plusieurs types d'éléments sont nécessaires et la reconfiguration demande une grande redondance, car chaque élément est particulier. Dans le contrôleur du microprocesseur proposé, un seul type d'élément est utilisé : le PLA. Un PLA virtuel initial est partitionné en plusieurs autres, dont certains sont des décodeurs. Une organisation à deux étages permet de réduire la surface de silicium.

Le microprocesseur utilise des PLA dynamiques synchrones : le plan OU utilise des transistors directement connectés à la masse alors que ceux du plan ET le sont via des interrupteurs. Les matrices PLA initiales, obtenues après un processus complexe de synthèse (codage des états, minimisation,...) par le système ASYL, sont modifiées pour le test afin de pouvoir localiser un défaut. La stratégie de test des PLA est basée sur une méthode d'abord présentée par BOZORGUI-NESBAT et McCLUSKEY [BOZ84] puis améliorée ensuite par BOZORGUI-NESBAT et KHAKBAZ dans [BOZ85], et M.M. LIGTHART et al. dans [LIG86]. L'idée de base de cette méthode est d'activer un monôme individuel grâce à l'ajout de quelques entrées à la matrice ET du PLA. Pour cette méthode la procédure de test se résume à l'application de quelques vecteurs ; aucun registre n'est nécessaire pour commander l'activation des lignes d'entrée ou de produit. Une présentation plus détaillée de cette méthode apparaît dans [WEH88].

Chaque PLA est considéré comme un bloc du point de vue de la reconfiguration. La redondance est constituée de monômes programmables, sauf pour les entrées de test. De fait, en prenant en compte

l'accroissement des interconnexions externes dû aux entrées et sorties redondantes, il devient évident que les monômes redondants soient un bon compromis entre l'accroissement de surface et l'apparition de fautes. Une étude de rendement et d'accroissement de surface est résumée dans le tableau 2.1 pour un PLA à 30 entrées, 30 sorties, 322 monômes, et 3 entrées supplémentaires [WEH88]. Ce tableau montre un accroissement évident du rendement pour l'ajout d'un seul terme produit.

Nombre de termes redondants	Surface mm^2	Rendement
0	2,75	0.81
1	2,77	0.93
2	2,78	0.94
3	2,79	0.95

Tableau 2.1 : analyse du rendement et de la surface en fonction du nombre de termes redondants [WHE88]

Les lignes d'entrée et de sortie sont conçues sur deux niveaux à connexions multiples. Cette technique assure une tolérance aux défauts de coupure. En contrepartie il faut accepter une diminution de la vitesse du fait de l'accroissement de capacité. Dans le contrôleur du microprocesseur, l'accroissement de surface dû à la redondance est de l'ordre de 20%. En outre, la taille de chaque PLA est inférieure à celle de l'exemple du tableau 2.1. Un rendement supérieur à 95% est donc espéré pour chaque PLA.

3.2. Modifications

Comme cela a déjà été écrit, la reconfiguration des PLA est réalisée

grâce à des monômes programmables de réserve qui peuvent remplacer les termes défectueux. La programmation peut utiliser soit des fusibles laser, soit des transistors N déplétés à grille flottante. Seuls les fusibles laser ont été retenus ici, pour trois raisons :

- la réversibilité n'est pas essentielle,
- les fusibles sont déjà utilisés pour le chemin de données,
- les fusibles sont plus fiables que les transistors à grille flottante.

Un monôme normal est réalisé comme montré en figure 2.5.

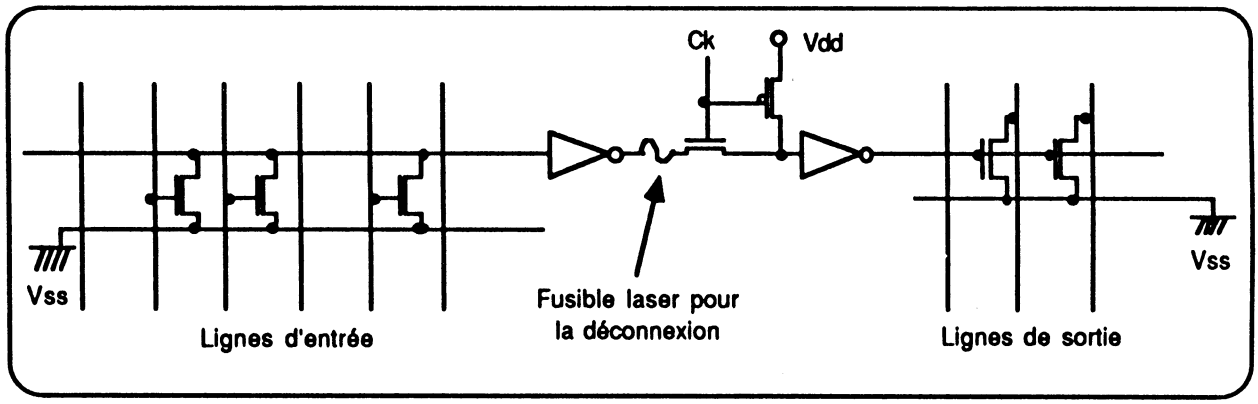


Figure 2.5 : monôme normal d'un PLA reconfigurable.

Un fusible laser permet sa déconnexion de la matrice OU quand il est prouvé qu'il est défectueux. Un monôme programmable est réalisé comme montré en figure 2.6. Un tel monôme ayant dans la matrice ET un transistor sur toutes les entrées et les entrées complémentées a toujours une valeur logique égale à 0. Il n'a donc pas d'influence sur le calcul des sorties. Il ne devient logiquement actif qu'après sa programmation (coupure de certains fusibles laser).

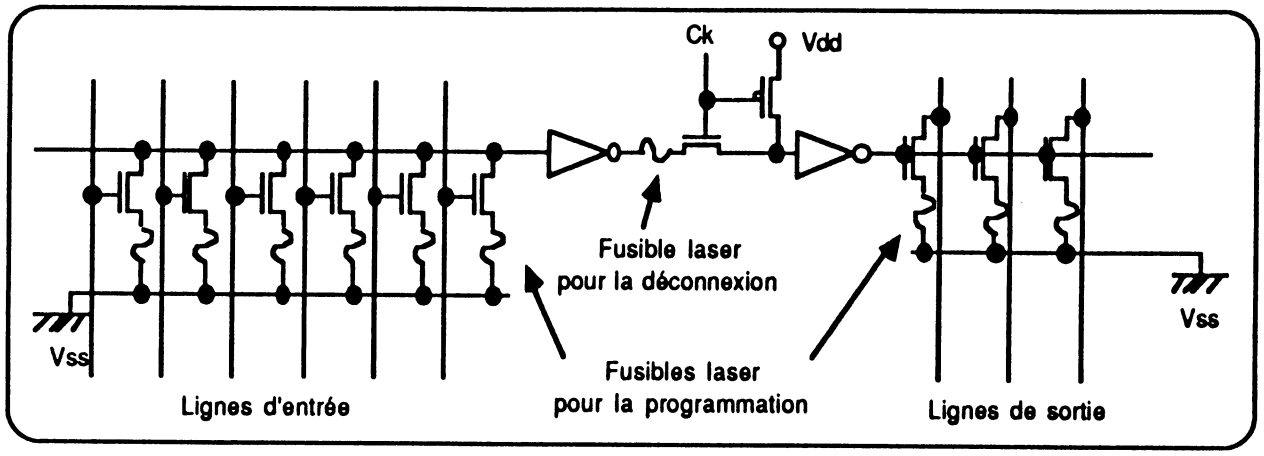


Figure 2.6 : monôme programmable d'un PLA reconfigurable.

Un exemple de reconfiguration est donné en figure 2.7.

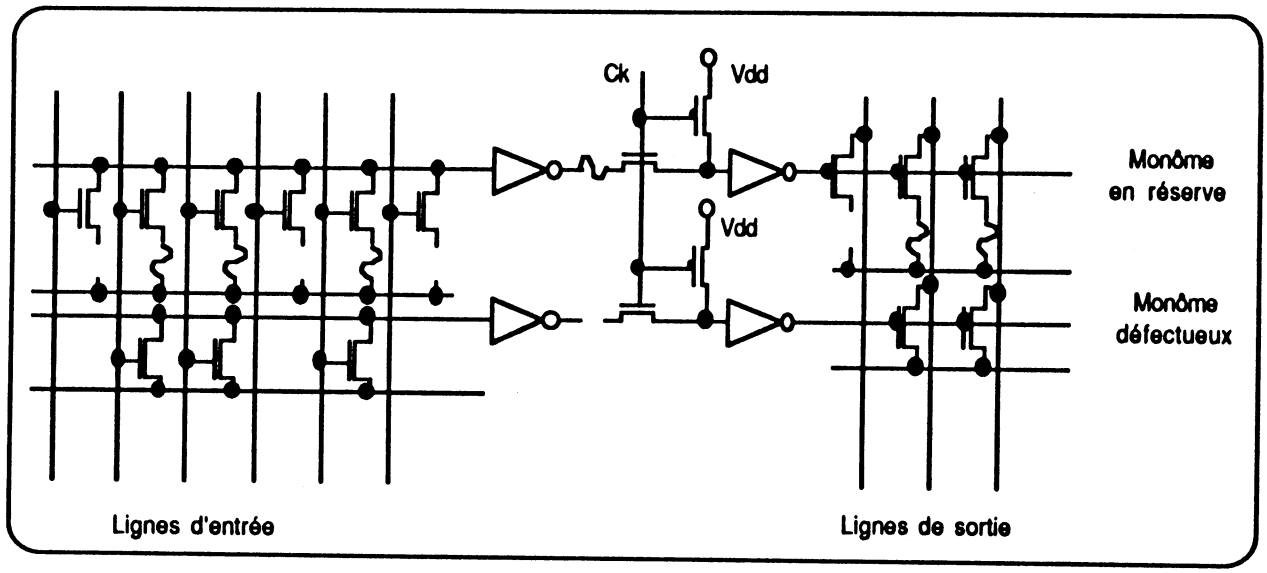


Figure 2.7 : reconfiguration dans un PLA.

Les monômes programmables peuvent aussi être déconnectés de la matrice OU si nécessaire.

4.Test hors ligne et reconfiguration

En fin de fabrication, on peut effectuer un test hors ligne très efficace. Cela est dû aux modifications introduites dans les PLA, à la conception particulière de l'UAL, qui a une structure ILA C-testable, et à l'implantation de registres "scan-path". De ce fait, le registre de microcommandes peut être chargé ou lu sériellement pendant le test.

En outre, deux registres "scan-path" sont ajoutés aux entrées du contrôleur et entre les deux étages de PLA dans le contrôleur. Deux signaux M et K sont disponibles pour commander les divers "scan-path".

Le test de fin de fabrication se déroule selon les étapes suivantes :

- test paramétrique (validation générale de la puce),
- test des trois registres "scan-path",
- test de chaque PLA du contrôleur, grâce aux registres "scan-path",
- test des éléments du chemin de données grâce aux registres "scan-path",
- test du microprocesseur entier en fonctionnement si tous les autres tests sont un succès.

Pendant cette phase, le test du chemin de données est réalisé sur les 17 tranches au même moment selon une stratégie "start small".

Quand des défauts sont détectés dans une partie reconfigurable du microprocesseur, une phase de reconfiguration débute après ce premier test. Dans les différents PLA, les monômes défectueux sont déconnectés et remplacés par ceux de réserve, programmés par fusibles laser pour réaliser la fonction correcte. Dans le chemin de données, le signal Crec de la tranche défectueuse est mis à 0 et tous les fusibles des bus sont coupés de cette tranche à la $17^{ème}$. Quand aucun défaut est détecté ou si la $17^{ème}$ tranche est défectueuse, tous les fusibles de celle-ci doivent être coupés.

Quand la reconfiguration est achevée, un second test est réalisé pour vérifier son efficacité. Il n'est pas prévu d'effectuer plusieurs tests et reconfigurations successifs. Un défaut ne peut être réparé que s'il est localisé pendant le premier test. Cette restriction est due à des motifs économiques, car le test et la reconfiguration d'une puce sont des processus gourmands en temps.

5.Gain de rendement de la puce du microprocesseur

Dans sa version présente, le prototype du composant est composé du seul microprocesseur. Des plots sont connectés de manière adéquate pour tester la puce. Cependant, ces plots ne sont pas pris en considération dans l'étude du rendement, car le but est d'obtenir un microprocesseur reconfigurable utilisable en WSI.

La surface utile de la puce est actuellement de 20 mm^2, dont 4 sont dédiés à la reconfiguration et 4 aux dispositifs non reconfigurables. Le rendement sans reconfiguration devrait être donc d'environ 35%. Après reconfiguration, le gain espéré est de 50%. Le prototype développé devrait permettre de valider ces résultats théoriques.

6.Le composant complet du microprocesseur

Le composant complet du microprocesseur envoyé à la fabrication est montré en figure 2.8. Il contient environ 50.000 transistors sur une surface de 36 mm^2 (incluant la place pour les motifs de positionnement du laser).

Parmi les 60 plots, 3 plots sont redondants (un par bus externe : adresses, pages et données). 6 plots seulement sont utilisés pour le test (2

plots pour les signaux de contrôle des registres "scan-path", 1 plot pour l'entrée série de ces registres et 3 plots pour les sorties séries).

La surface requise pour les deux registres "scan-path" entre le chemin de données et le contrôleur est d'environ 2 mm^2. Un registre

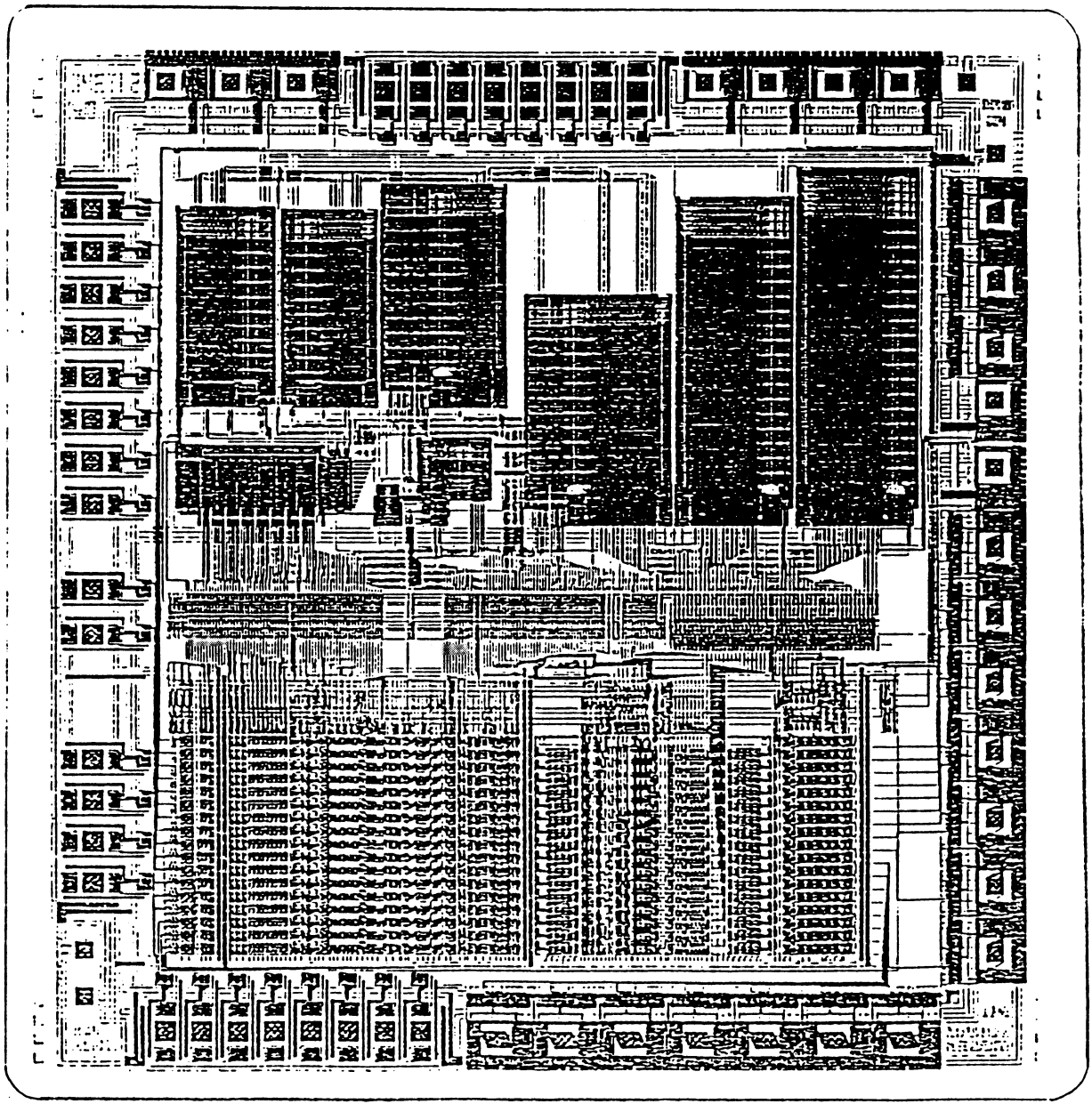


Figure 2.8 : dessin des masques du composant complet.

maître-esclave est par ailleurs indispensable pour la synchronisation des microcommandes. De même, des amplificateurs sont nécessaires pour les signaux d'entrée du contrôleur. La surface effectivement occupée par des facilités de test peut donc être évaluée à environ 0,5 mm^2. Si l'on considère, comme dans le paragraphe 5, uniquement la surface utile de la puce (sans la couronne de plots) la surface de silicium due à la présence des dispositifs de test et de reconfiguration peut être évaluée entre 4 et 5 mm^2 (la surface rajoutée pour la reconfiguration est environ 20%-25%).

M. SOUEIDAN et R. LEVEUGLE donnent des informations supplémentaires concernant ce microprocesseur dans [LEV88a...f] De plus les rapports techniques [D33] [D34] [D35] décrivent les différents blocs constituant le microprocesseur et leurs schémas.

7.Conclusion

La méthode de reconfiguration présentée ici peut être appliquée à tout type de circuit intégrant un contrôleur et un chemin de données en tranches. Cependant, quelques modifications sont nécessaires durant la conception, spécialement pour les opérateurs spécifiques complexes.

Ces techniques peuvent être utilisées pour améliorer la productivité des circuits VLSI et peuvent amener dans le futur à l'intégration de systèmes complets sur une seule puce.

Chapitre 3 :
Unités Arithmétiques et Logiques tolérant les défauts de fabrication

Dans le cadre d'une étude générale concernant les processeurs tolérant les défauts de fabrication, nous nous intéressons à la conception d'Unités Arithmétiques et Logiques (UAL) destinées à être intégrées dans ces processeurs. Des critères d'évaluation tels que la complexité, la performance, la testabilité et la reconfigurabilité sont définis.

Une UAL de type "Ripple Carry" utilisant un additionneur statique sera brièvement décrite. Cette UAL sert de référence pour l'évaluation des autres types d'UAL et a été réellement implantée sur silicium.

Huit types d'UAL sont ensuite analysés. Leur intérêt pour notre application est évalué en fonction des différents critères définis précédemment.

Parmi les huit types analysés, deux seront sélectionnés pour être implantés avec les mêmes contraintes topologiques que l'UAL de référence. Il est aussi possible, selon les contraintes de vitesse, de substituer à l'UAL statique une UAL utilisant la chaîne de retenue de Manchester ou une retenue bondissante ("Carry-Skip").

1. Critères d'évaluation

Quatre critères d'évaluation (complexité, performance, testabilité et reconfigurabilité) sont définis et servent à comparer les différents types d'UAL analysés.

1.1.Complexité

Pour la complexité, nous donnerons deux évaluations. La première, en nombre de transistors réels, est proportionnelle à la surface occupée par les interconnexions entre transistors ; la deuxième, en nombre de transistors dits équivalents, correspond à la surface des zones actives.

Le transistor unité a les dimensions W_{min}/L_{min}. Un inverseur minimal a un transistor N de dimensions W_{min}/L_{min} et un transistor P de dimensions $2.W_{min}/L_{min}$, soit un total de trois transistors équivalents. Selon le principe de dimensionnement utilisé, une porte NAND à n entrées a des transistors N de dimensions $n.W_{min}/L_{min}$ et des transistors P de dimensions $2.W_{min}/L_{min}$, soit un total de $n^2 + 2n$ transistors équivalents pour 2n transistors réels. De même, une porte NOR à n entrées a des transistors N de dimensions W_{min}/L_{min} et des transistors P de dimensions $2n.W_{min}/L_{min}$, soit un total de $n + 2n^2$ transistors équivalents pour 2n transistors réels.

1.2. Vitesse

En ce qui concerne la vitesse, nous évaluerons le temps de propagation le plus long (chemin critique) qui correspond au temps de calcul de l'UAL. Les simulations des différentes portes dans notre technologie avec une charge de 0,05 pF (charge moyenne due aux portes suivantes) ont donné les temps de propagation suivants :

τ_{INV} = 2,5 ns (pour un inverseur)

$\tau_{POR\ 2\ à\ 3}$ = 2,5 ns (portes NAND ou NOR à 2 ou 3 entrées)

$\tau_{POR\ 4\ à\ 5}$ = 3 ns (portes NAND ou NOR à 4 ou 5 entrées)

τ_{PT} = 3 à 4 ns (porte de transfert)

τ_{EOR} = 5 ns (porte OU Exclusif)

Il faut noter que l'estimation du temps de propagation est grossière et correspond à la technologie HCMOS3C 1,2 µm et à la stratégie de dimensionnement définie en 1.1.

1.3. Testabilité

Une testabilité aisée est nécessaire pour pouvoir tester les UAL avec une couverture élevée et pour pouvoir utiliser la logique redondante d'une façon correcte et précise (localisation des défauts).

L'évaluation de la testabilité est faite indépendamment des hypothèses de panne dans une technologie donnée. La structure itérative de l'UAL est prise en compte et le critère de testabilité est assuré par l'existence d'un test court pseudo-exhaustif (auquel on se réfère comme la propriété de C-testabilité).

Les travaux dans le domaine du test des circuits itératifs sont nombreux [FRI73] [DIA76] [PAR81] [SRI81] [ABO84] [SHE84] [CHE87] [ENS88], [VAR88], [HUA88]. Il est utile de rappeler les points principaux concernant le test de ces circuits afin de bien définir le critère de testabilité associé.

Définition 1 : les réseaux logiques itératifs à une dimension [FRI73 [ENS88]

Un réseau logique itératif à une dimension est un réseau linéaire de n cellules identiques (cf. figure 3.1). Chaque cellule est un circuit combinatoire, recevant une entrée X_i de la cellule gauche voisine et une entrée externe contrôlable (notée Y_i).

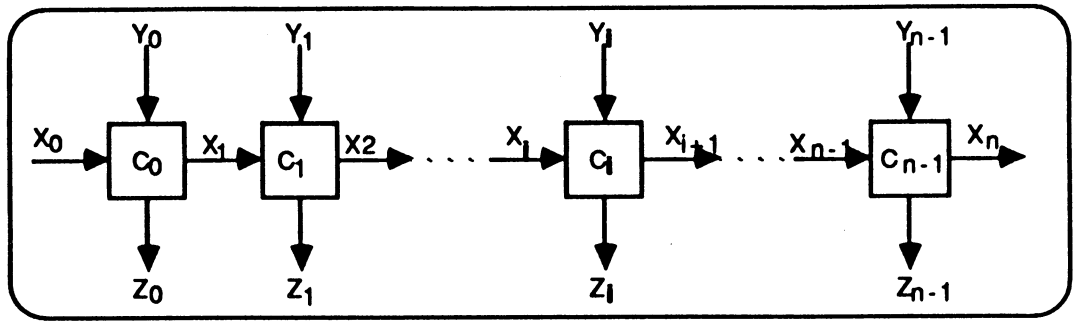


Figure 3.1 : réseau logique itératif

Elle génère une sortie observable Z_i et transmet une sortie X_{i+1} à la cellule droite voisine. A l'exception de l'entrée initiale X_0 et de la sortie finale X_n, les valeurs des X_i ($1 \geq i \geq n-1$) ne sont ni contrôlables ni observables.

Définition 2 : Modèle d'erreur [SRI81]

Soit M un module dans un réseau itératif sous test. Soit F une fonction réalisée par M et soit e le nombre d'états internes de M ; e=1 si M est combinatoire. Un mauvais fonctionnement P de M est appelé erreur simple de M si P change de façon permanente M en un module M^P qui réalise la fonction F^P avec $F \neq F^P$ et tel que le nombre d'états e^P de M^P n'est pas plus grand que e. Ainsi, une erreur simple dans un module combinatoire peut affecter d'une façon arbitraire la table de vérité relative au module, mais ne peut pas le transformer en un module séquentiel.

Pour détecter une telle erreur, il est nécessaire et suffisant d'appliquer au module de NE entrées les 2^{NE} vecteurs d'entrée possibles. Ce modèle d'erreur simple recouvre en fait l'ensemble des types d'erreur assurés d'être décelés par un ensemble de test exhaustif.

Il est aussi supposé que seul un module du réseau est fautif à un instant donné (erreur non multiple). L'élaboration de séquences de test couvrant les erreurs multiples (plusieurs cellules défaillantes en même temps) est

cependant envisageable [DIA76] [ABO84] [CHE87].

Définition 3 : conditions de testabilité des réseaux itératifs

Pour pouvoir tester un réseau logique itératif suivant le modèle d'erreur précédent, il faut pouvoir satisfaire les deux conditions suivantes [SRI81] :

i) la possibilité d'appliquer une séquence de test exhaustif à chacune des cellules du réseau (soit 2^N valeurs possibles si la cellule a N entrées).

ii) la possibilité de propager vers des sorties primaires du réseau (Z_i ou X_n) une valeur fautive en sortie de n'importe quelle cellule.

Définition 4 : C-testabilité [PAR81] [SHE84] [ENS88] [CER88]

D'une façon générale, le nombre de vecteurs de test exhaustif (NVTE) nécessaire pour tester le réseau de la figure 3.1 selon les hypothèses précédentes est donné par la formule suivante :

$NVTE = 2^{NE}$, où NE est le nombre d'entrées du réseau.

Ceci signifie que NVTE est une fonction du nombre d'entrées du réseau, donc du nombre de cellules le constituant.

La C-testabilité est la propriété d'un réseau que l'on peut tester avec un NVTE constant et indépendant du nombre de cellules (n) du réseau.

Définition 5 : RT-graphe [SRI81]

Il est montré dans [SAU85] qu'un réseau combinatoire itératif est équivalent à un circuit séquentiel. La cellule de base, dans ce cas, au lieu d'être repétée n fois, est rebouclée et le fonctionnement du circuit séquentiel se fera en n passages dans la boucle de réaction (compromis spatio-temporel). La table de vérité de la cellule de base est alors remplacée par un tableau d'états. Dans ce tableau X_i, X_{i+1}, Y_i, Z_i correspondent respectivement à l'état, l'état suivant, l'entrée et la sortie.

Pour qu'un circuit itératif soit C-testable, il suffit que le graphe d'états

associé à l'automate équivalent n'ait que des transitions répétables. En effet le test exhaustif d'une cellule correspond au passage par toutes les transitions du graphe d'états équivalent. Il s'agit de rechercher un chemin faisant passer par tous les arcs du graphe d'états qui donnera un vecteur d'entrée faisant passer chaque cellule par une transition différente.

Tout réseau dont la cellule de base a un graphe d'états de type RT-graphe est C-testable.

Le but recherché est de vérifier le bon fonctionnement de chaque cellule par l'application d'un nombre constant (indépendant de la longueur du réseau) et réduit de vecteurs de test.

Le critère de C-testabilité, associé au modèle d'erreur simple évoqué précédemment, est donc retenu pour l'évaluation de la testabilité des UAL étudiées.

1.4. Reconfigurabilité

La reconfiguration utilisée consiste à remplacer des éléments défectueux par des éléments de réserve supposés sains. Ceci nécessite l'utilisation de dispositifs de connexion et de déconnexion, tels que les fusibles et les antifusibles [MET83], les transistors à grilles flottantes [SHA84] [GIR86], ou les dispositifs programmés par logique [CHE85] [KAT86]. Dans le cas de l'UAL, la reconfiguration se fait par l'ajout d'une ou de deux tranche(s) supplémentaire(s) en utilisant des dispositifs de commutation programmés par logique. Le principe de cette reconfiguration est illustré dans la figure 3.2 [GEN87].

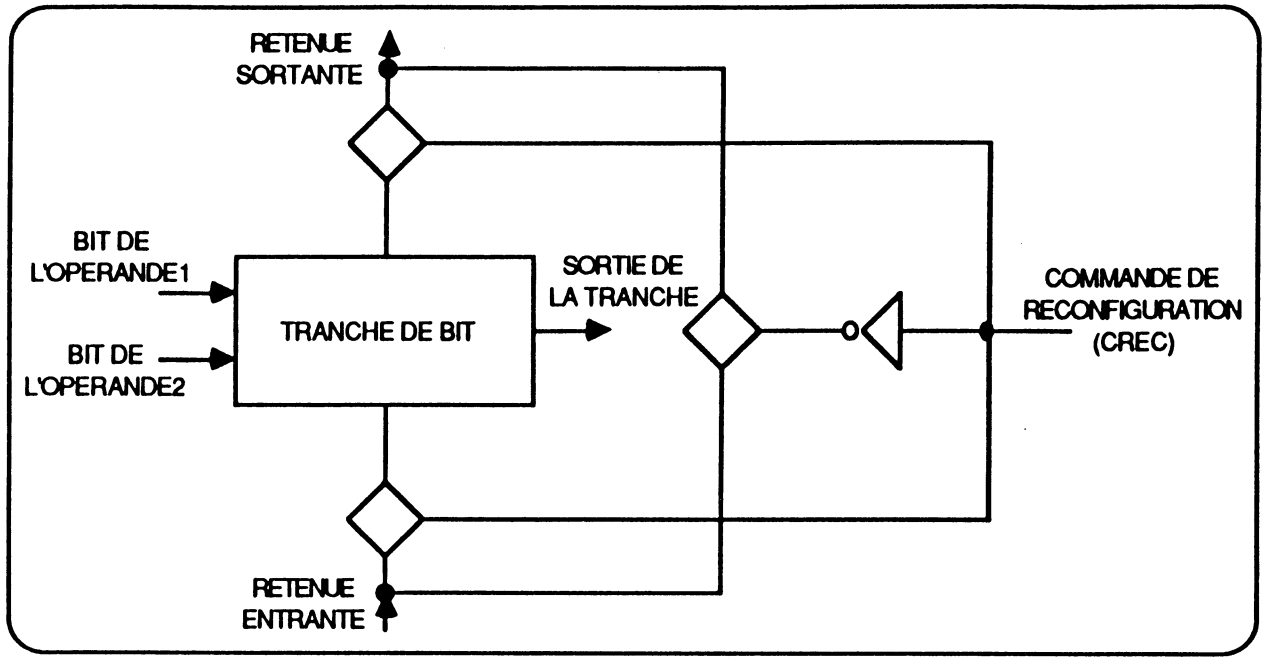


Figure 3.2 : principe de la reconfiguration.

Le but recherché est l'augmentation du rendement de fabrication. Une contrainte du cahier des charges du projet impose par ailleurs de ne pas dépasser 25 à 30% de la surface initiale à des fins de reconfiguration.

2. UAL de type "Ripple Carry Adder" implantée dans le microprocesseur (HYETI2)

Il s'agit d'une unité arithmétique et logique à chemin de retenue simple utilisant comme additionneur le "ripple carry adder". Le choix de cet additionneur a été dicté par les raisons suivantes :

- sa simplicité,
- un temps de propagation suffisant pour l'utilisation de cette UAL au sein du microprocesseur HYETI2,
- sa facilité de reconfiguration et de test.

2.1. Description générale [JAY86] [GEN87]

L'UAL est définie comme suit (cf. figure 3.3) :

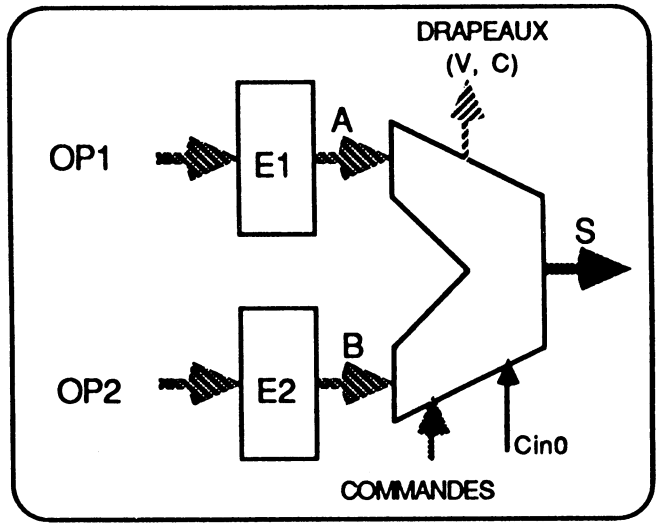


Figure 3.3 : schéma général de l'UAL.

A,B : opérandes sur 16 bits codés en complément à 2,

S : résultat de l'opération sur 16 bits codé en complément à 2,

commandes : commandes de sélection de l'opération,

Cin0 : retenue entrante,

V (débordement) : V = 1 débordement,
V = 0 pas de débordement,

C (retenue) : C = 1 retenue sortante,
C = 0 pas de retenue sortante.

N (signe) : N = 1 résultat négatif,
N = 0 résultat positif,

Z (zéro) : Z = 1 résultat nul,
Z = 0 résultat différent de zéro,

N et Z sont générés par deux circuits spéciaux dans la partie opérative.

Les deux registres E1 et E2 sont placés en tampon à l'entrée de l'UAL.

E1 est un registre à décalage bidirectionnel. Les opérations possibles sont décrites dans le tableau 3.1. Les opérations de décalage, réalisées par E1, ne seront pas considérées lors de l'étude des différentes UAL.

OPERATION	ENTREE OP1	ENTREE OP2	RESULTAT S
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		DECALAGE LOGIQUE A DROITE
LSL	A		DECALAGE LOGIQUE A GAUCHE
ASR	A		DECALAGE ARITHMETIQUE A DROITE
ASL	A		DECALAGE ARITHMETIQUE A GAUCHE

Tableau 3.1 : Opérations de l'UAL.

2.2. UAL statique

Une tranche de bit de l'UAL est constituée, d'une part, d'un bloc combinatoire sélectionnant l'opération désirée en fonction des commandes, et d'autre part, d'un opérateur effectuant cette opération (cf. figure 3.4). Les deux blocs sont réalisés en CMOS statique.

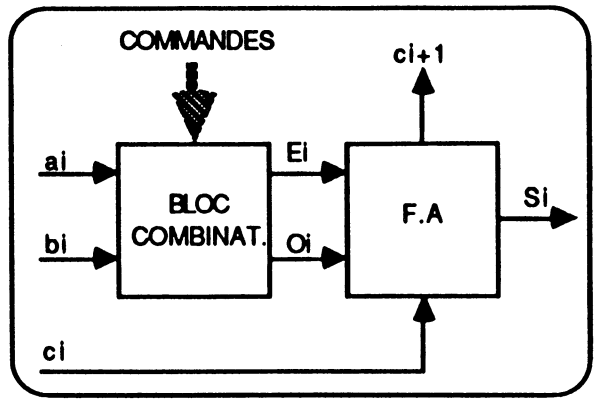


Figure 3. 4 : schéma général d'1 bit de l'UAL.

Des équations logiques correspondant aux opérations désignées dans le tableau 3.1 ont été établies. Les valeurs des commandes suivant les opérations à effectuer ont été déterminées et sont résumées dans le tableau 3.2.

Le schéma logique d'une tranche de 1 bit, donné dans la figure 3.5, en a été déduit.

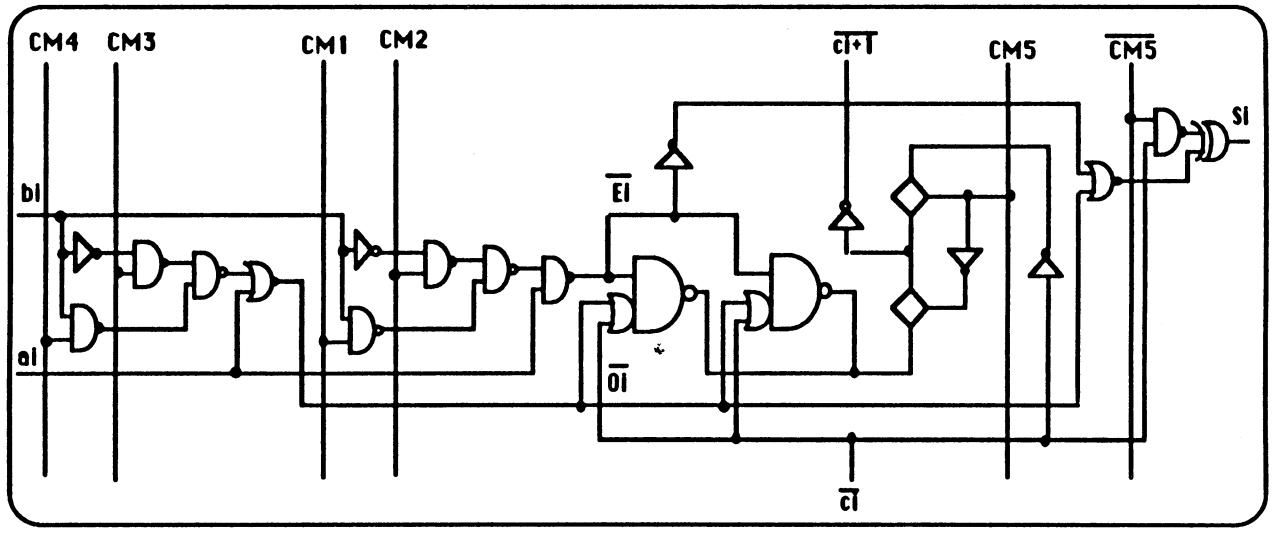


Figure 3.5 : schéma logique d'une tranche de l'UAL.

Dans ce schéma, un multiplexeur et une commande (CM5) supplémentaires ont été ajoutés dans la partie calculant la retenue sortante de l'additionneur. Cet ajout a pour but de rendre les graphes des états associés aux opérations logiques de type RT-graphe

(cf. paragraphe 2.2.3).

OPERATION	EQUATIONS LOGIQUES	CONDITION		COMMANDES DE L'UAL				
		c0	c0b	CM1	CM2	CM3	CM4	CM5
ADD	Oi = ai + bi Ei = ai . bi Si = ai ⊕ bi ⊕ ci ci+1 = ai . bi + (ai + bi) . ci	0	1	1	0	0	1	0
ADDC	Oi = ai + bi Ei = ai . bi Si = ai ⊕ bi ⊕ ci ci+1 = ai . bi + (ai + bi) . ci	C	$\overline{C}$	1	0	0	1	0
SUB	Oi = ai + $\overline{bi}$ Ei = ai . $\overline{bi}$ Si = ai ⊕ bi ⊕ ci ci+1 = ai . bi + (ai + bi) . ci	1	0	0	1	1	0	0
SUBC	Oi = ai + $\overline{bi}$ Ei = ai . bi Si = ai ⊕ $\overline{bi}$ ⊕ ci ci+1 = ai . $\overline{bi}$ + (ai + $\overline{bi}$) . ci	$\overline{C}$	C	0	1	1	0	0
INC	Oi = ai Ei = 0 Si = (Oi . Ei) ⊕ ci ci+1 = Ei + Oi . ci	1	0	0	0	0	0	0
DEC	Oi = 1 Ei = ai Si = (Oi . Ei) ⊕ ci ci+1 = Ei + Oi . ci	0	1	1	1	1	1	0
NOT	Oi = ai Ei = 0 Si = (Oi . $\overline{Ei}$) ⊕ ci ci+1 = ci = c0 = 1	1	0	0	0	0	0	1
AND	Oi = 1 Ei = ai . bi Si = (Oi . Ei) ⊕ ci ci+1 = ci = c0 = 1	1	0	1	0	1	1	1
OR	Oi = ai + $\overline{bi}$ Ei = ai Si = (Oi . Ei) ⊕ ci ci+1 = ci = c0 = 1	1	0	1	1	1	0	1
NAND	Oi = ai Ei = ai . $\overline{bi}$ Si = (Oi . Ei) ⊕ ci ci+1 = ci = c0 = 1	1	0	0	1	0	0	1
NOR	Oi = ai + bi Ei = 0 Si = (Oi . Ei) ⊕ ci ci+1 = ci = c0 = 1	1	0	0	0	0	1	1
EOR	Oi = ai + $\overline{bi}$ Ei = ai . $\overline{bi}$ Si = (Oi . $\overline{Ei}$) ⊕ ci ci+1 = ci = c0 = 1	1	0	0	1	1	0	1

Tableau 3.2 : équations logiques et valeurs des commandes.

2.2.1. Complexité

Dans cette partie, nous ne tiendrons pas compte de la place réservée à la reconfiguration. Le bloc combinatoire est commun aux différentes UAL qui seront examinées. La deuxième partie qui est en fait un simple additionneur sera modifiée pour les différents types d'UAL.

Les complexités du bloc combinatoire et de l'additionneur sont données dans le tableau 3.3.

BLOC COMBINATOIRE		
PORTES	NOMBRE DE TRANSISTORS	NOMBRE DE TRANSISTORS EQUIVALENTS
2 × 	4	6
7 × 	28	56
1 × 	4	10
1 tranche	36	72
16 tranches	576	1152

ADDITIONNEUR (FA)		
PORTES	NOMBRE DE TRANSISTORS	NOMBRE DE TRANSISTORS EQUIVALENTS
2 × 	12	18
2 × 	4	6
4 × 	8	12
1 × 	4	10
1 × 	4	8
1 × 	6	9
1 tranche	38	63
16 tranches	608	1008

Tableau 3.3 : complexités du bloc combinatoire et de l'additionneur.

soit au total 1184 transistors réels et 2160 transistors équivalents.

Ces chiffres serviront de référence pour la suite de notre étude.

2.2.2. Vitesse

La vitesse est estimée en considérant le schéma de la figure 3.5 et

les remarques suivantes :

- la lettre b représente la complementation logique d'une variable (Eib se prononce Ei barre),

- on note PCOM3 une porte complexe à 3 entrées ayant un temps de propagation équivalent à celui d'une porte NAND à 3 entrées.

Seule la vitesse de l'additionneur (temps de calcul de la retenue) a une influence sur la comparaison :

τ_{BC} (b_i ---> E_ib ou b_i ---> O_ib) $= \tau_{INV} + 3\,\tau_{NAND2}$

τ_{BCR} (C_ib ---> $C_{i+1}b$ ou E_ib ---> $C_{i+1}b$) $= \tau_{PCOM3} + \tau_{PT} + \tau_{INV}$

τ_{BS} (C_ib ---> S_i) $= \tau_{NAND2} + \tau_{EOR}$

D'après la simulation SPICE, une tranche de l'UAL a les temps de propagation suivants :

$\tau_{BCR} = 9$ ns, $\tau_{BS} = 7{,}5$ ns, $\tau_{BC} = 10$ ns

Le temps de calcul de l'UAL (figure 3.5) sur 16 bits est donné par la relation :

$\tau\,UAL = \tau_{BC} + \tau\,(E_ib$ ---> $C_{i+1}b) + 15\,\tau_{BCR} = \tau_{BC} + 16\,\tau_{BCR}$

Ce temps vaut 154 ns, ce qui correspond à une fréquence maximale estimée de 6,5 MHZ.

2.2.3. Testabilité

Considérons l'opérateur UAL de 16 bits dans son ensemble

(cf. figure 3.6).

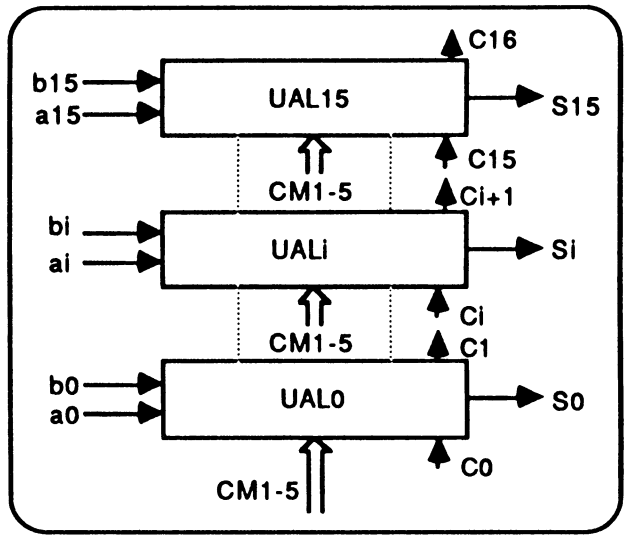


Figure 3.6 : UAL type "Ripple Carry Adder"

L'ensemble des points accessibles en entrée est:

E = {COM1-5, a0-15, b0-15, C0}

et en sortie

S = {S0-15, C16}

Le test exhaustif d'un tel opérateur exige 2^{38} vecteurs de test (38 entrées). Ce nombre énorme de vecteurs de test nécessite un temps de test trop élevé, donc inacceptable. L'organisation en tranche de bit de cet opérateur permet de le considérer comme un circuit itératif. Cet opérateur est en fait considéré comme 12 circuits itératifs, chacun réalisant une des 12 fonctions existantes. L'étude de chacune des fonctions réalisables par l'UAL fait apparaître que pour plusieurs opérations, les graphes d'états associés ne sont pas de type RT-graphe. Ce problème se pose pour 2 opérations arithmétiques (INC et DEC), et pour de nombreuses opérations logiques étant donné que la retenue entrante Ci n'a pas d'influence sur le résultat du calcul des fonctions logiques. La modification de la cellule de base de telle façon que Ci = Ci+1 rend le graphe d'états associé à cette cellule de type RT-graphe pour toutes les fonctions logiques .

L'étude de la testabilité de cet opérateur selon le principe qui vient d'être énoncé donne trois types de graphes ; deux de type RT-graphe (§a, b) et un n'est pas de type RT-graphe (§c).

a- Les graphes des états pour les opérations logiques (NOT, AND, OR, NAND, NOR et EOR).

Prenons à titre d'exemple le cas de l'opération OR.
La table des états et son graphe sont présentés à la figure 3.7.

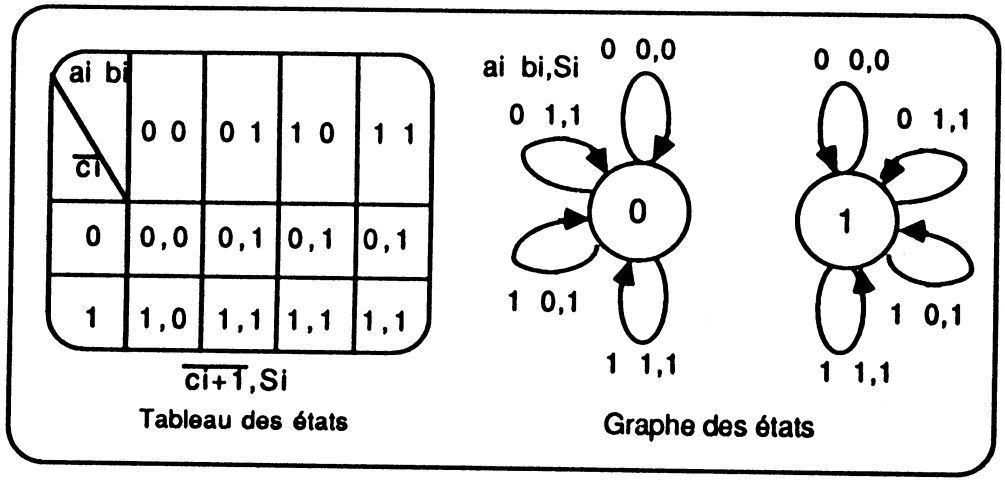
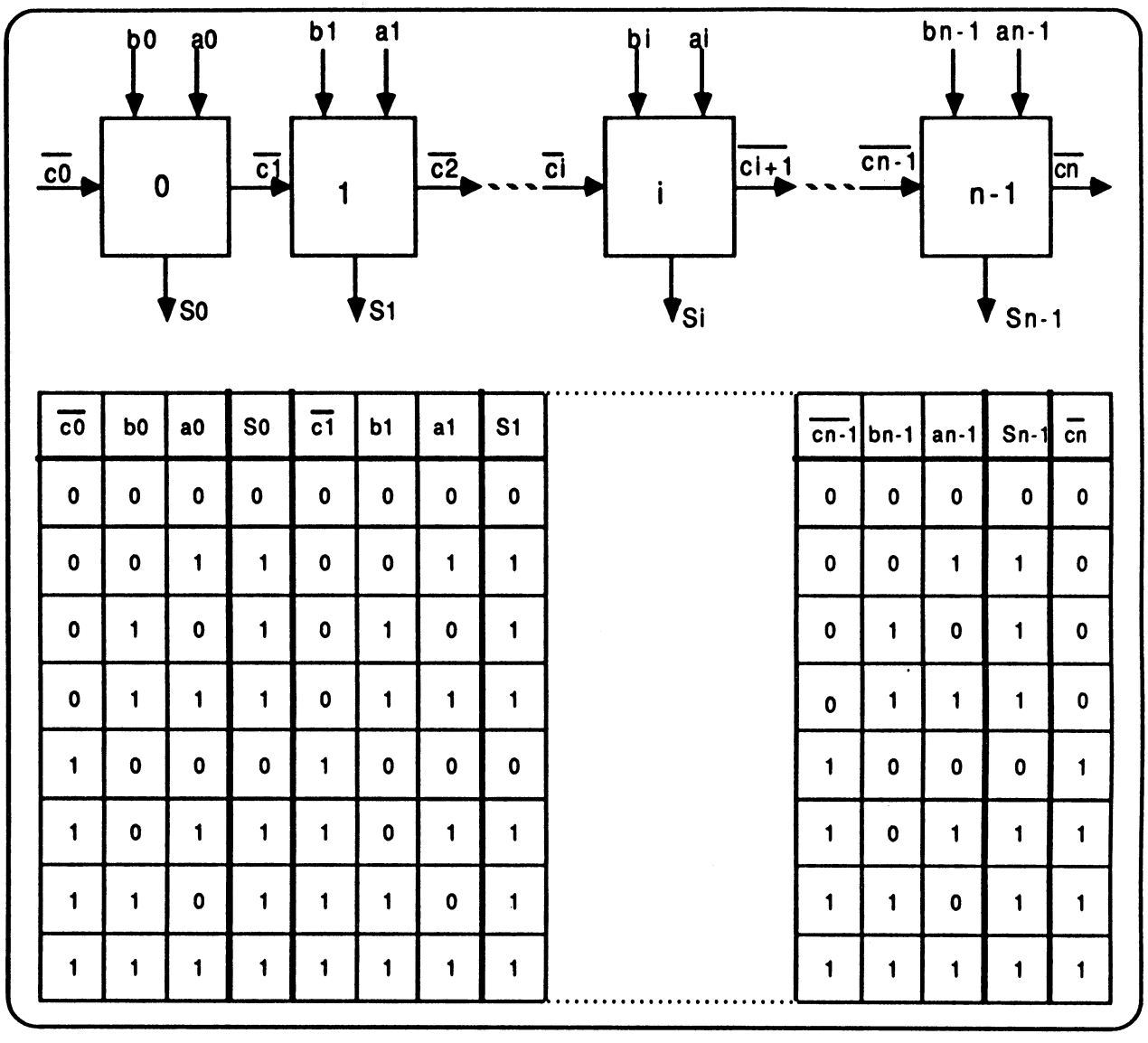


Figure 3.7 : Tableau et graphe des états de l'opération OR.

Ceci donne, pour une UAL de n bits, les vecteurs de test exhaustif montrés dans le tableau 3.4.



$\overline{c0}$	b0	a0	S0	$\overline{c1}$	b1	a1	S1
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

$\overline{cn-1}$	bn-1	an-1	Sn-1	$\overline{cn}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Tableau 3.4 : vecteurs de test de l'opération OR sur n bits.

b- Les graphes des états pour les opérations SUB, ADD, SUBC et ADDC.

Prenons le cas de l'opération ADD pour laquelle la table et le graphe associé à cette table sont donnés à la figure 3.8.

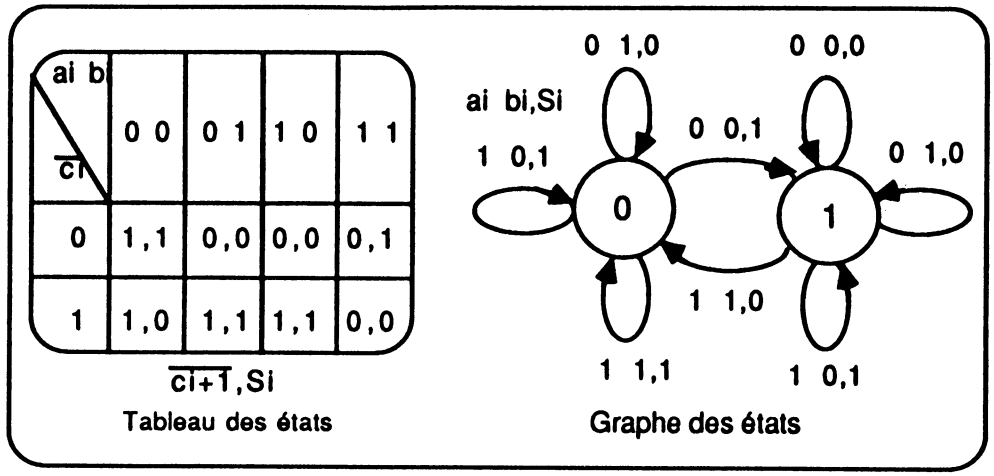


Figure 3.8 : tableau et graphe des états de l'opération ADD.

Ceci donne, pour l'UAL décrite au paragraphe a, les vecteurs de test exhaustif donnés dans le tableau 3.5.

$\overline{c0}$	b0	a0	S0	$\overline{c1}$	b1	a1	S1
0	0	1	0	0	1	0	0
0	1	0	0	1	1	1	0
0	1	1	1	0	0	0	1
0	0	0	1	1	0	0	0
1	0	0	0	1	0	1	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	0	0	1	0

$\overline{cn-1}$	bn-1	an-1	Sn-1	$\overline{cn}$
1	1	1	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
0	0	0	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1

Tableau 3.5 : vecteur de test de l'opération ADD.

c- Les deux graphes d'état correspondant aux opérations INC et DEC.

Considérons le cas de l'opération INC : le tableau d'états et le graphe associé sont donnés à la figure 3.9.

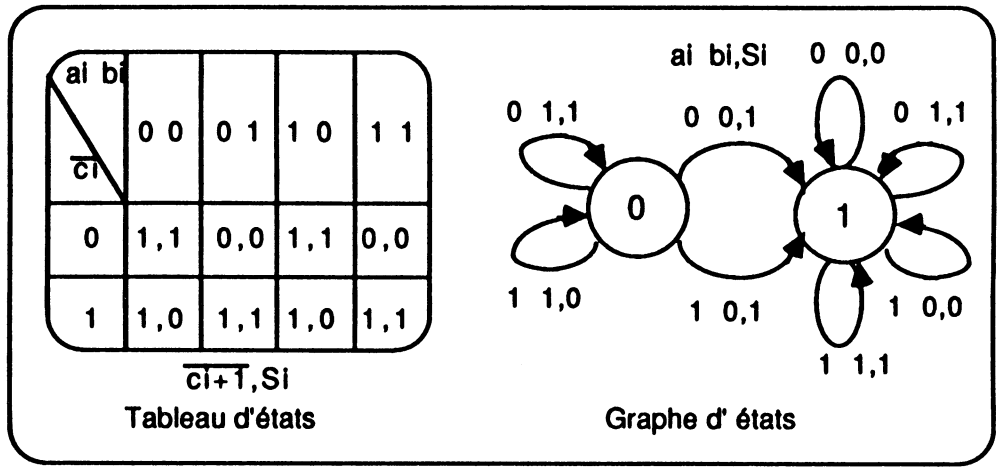


Figure 3.9 : tableau et graphe des états de l'opération INC.

Remarquons que ce graphe d'états n'est pas fortement connexe (manque de transition de l'état 1 vers l'état 0), il n'est pas de type RT-graphe.

d) Détermination des vecteurs de test de l'UAL

_ La formule générale donnant le nombre de vecteurs de test exhaustif de l'UAL étudiée est pour n bits :

$$N_{vtt} = N_{rt}\, 2^N + (N_{to}-N_{rt})\,[2^N + 2\,(n-1)]$$

- N_{rt} est le nombre d'opérations ayant un graphe d'états de type RT-graphe,
- N est le nombre d'entrées de la cellule de base,
- N_{to} est le nombre total des opérations,
- n est le nombre de cellules dans le circuit.

Le principe de base pour déterminer les vecteurs de test exhaustif pour les graphes des opérations INC ou DEC consiste à :

- appliquer toutes les combinaisons possibles sur les entrées contrôlables de la première cellule.
- ajouter deux vecteurs de test supplémentaires pour chaque cellule de 1 à n - 1. Ceci permet de propager toutes les valeurs possibles de la retenue entrante vers la retenue sortante de manière à effectuer un test exhaustif.

Le nombre de vecteurs de test est donné par la formule suivante :

$N_{vt} = 2^N + 2 (n - 1)$

Dans le cas de l'UAL 16 bits, on a :

$N_{rt} = 8$ (les séquences de test des opérations ADD et SUB sont les mêmes pour ADDC et SUBC), $N = 3$, $N_{to} = 10$, $n = 16$,

$N_{vtt} = 140$.

L'UAL implantée dans le microprocesseur est sur 17 bits (16 bits + une tranche de réserve). Le nombre de vecteurs de test de cette UAL est de 146. Ces derniers ont été donnés en annexe 3.1 .

2.2.4. Reconfigurabilité

La deuxième contrainte de cette UAL est sa reconfigurabilité. Deux possibilités ont été prévues : soit l'ajout d'une tranche redondante (figure 3.10), soit de deux (figure 3.11).

Examinons la place supplémentaire nécessaire pour réaliser ceci.

Le nombre de transistors dû à la reconfiguration pour chaque tranche est donné dans le tableau 3.6.

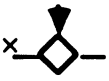
PORTES	NB TR. R.	NB. TR. E.
3 × 	6	9
1× 	2	3
Σ	8	12

Tableau 3.6 : nombre de transistors dû à la reconfiguration pour chaque tranche

Dans le cas d'une tranche supplémentaire, il faut 212 transistors réels et 342 transistors équivalents, c'est-à-dire 16 % de la surface de l'UAL. Pour deux tranches, il faut 374 transistors et 503 transistors équivalents. Cela donne un supplément de 23 % de surface.

2.2.5. Conclusion

L'UAL de référence a été implantée sur silicium dans le microprocesseur HYETI2 sur une surface de 0,694 mm^2. Cette surface comporte 1184 + 374 transistors et 2160 + 503 transistors équivalents.

La vitesse maximale estimée est de 6,5 MHZ.

Le nombre de vecteurs de test exhaustif est de 140 pour 16 tranches et de 146 pour 17 tranches.

Une tranche de réserve a été utilisée (16% de surface supplémentaire) pour satisfaire les contraintes du cahier des charges de la partie opérative (PO). En fait, deux tranches supplémentaires de la PO donnent une surface supplémentaire supérieure à 30% de la surface initiale.

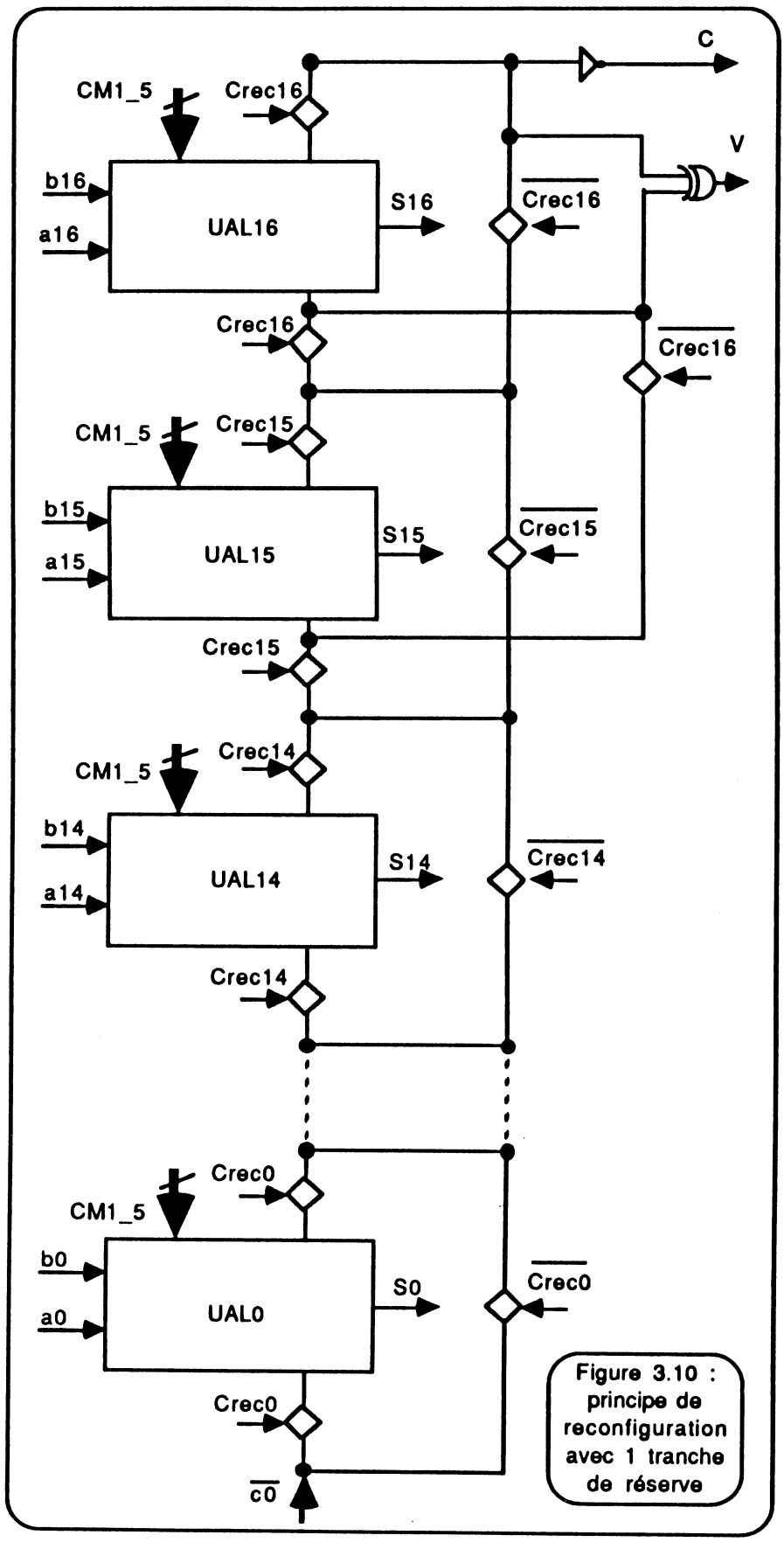


Figure 3.10 : principe de reconfiguration avec 1 tranche de réserve

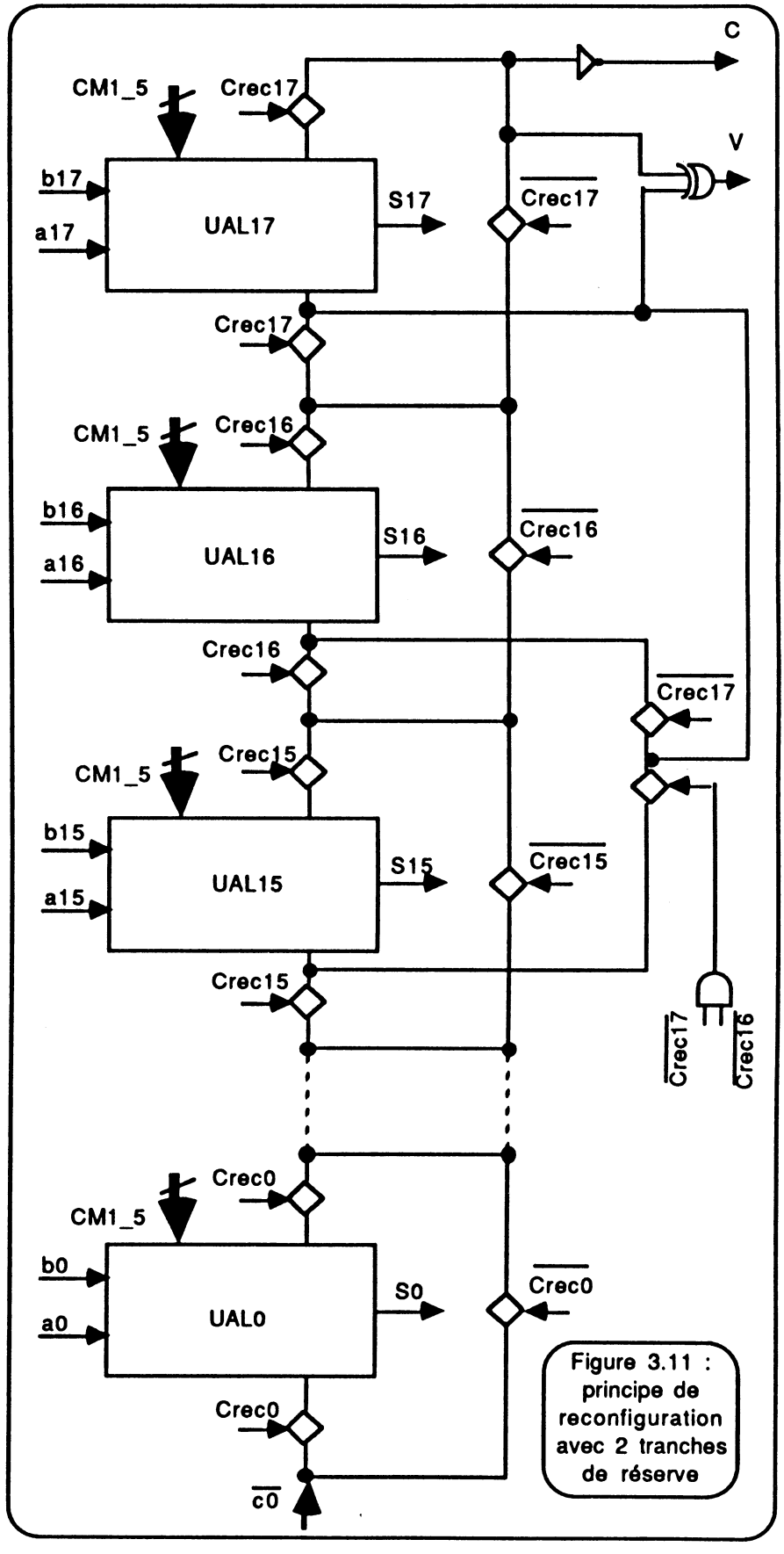


Figure 3.11 : principe de reconfiguration avec 2 tranches de réserve

2.3. UAL de type domino

Généralement, une porte logique de type CMOS domino est constituée d'une porte dynamique suivie par un buffer CMOS. Pour des opérations statiques ou à faibles fréquences, un transistor de maintien du niveau est à ajouter en parallèle avec le transistor de précharge P (cf. figure 3.12).

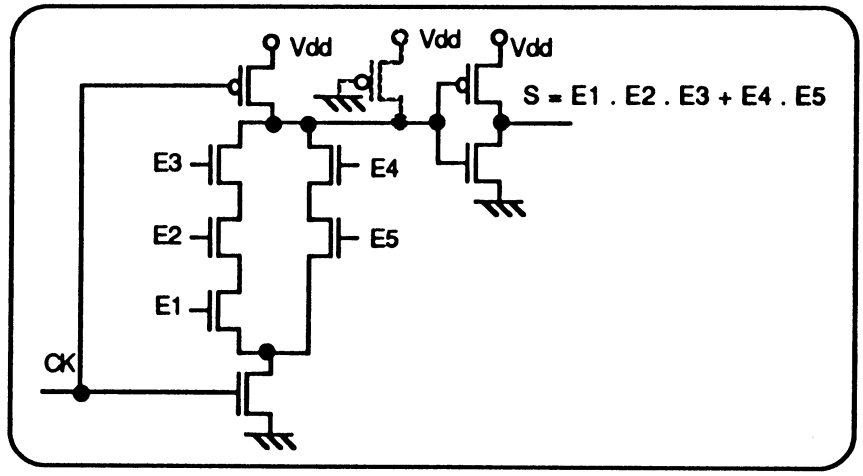


Figure 3.12 : structure CMOS domino.

Ce type de circuit combine les avantages des circuits dynamiques, par rapport aux circuits NMOS, (faible consommation, surface comparable avec la structure NMOS, rapidité) et celles des circuits CMOS (stabilité, facilité d'utilisation des circuits statiques).

R.H. KRAMBECK et al. dans [KRA 82] ont montré que la technique CMOS domino donne des circuits comparables en surface avec les circuits de types NMOS ou pseudo-NMOS, mais avec une vitesse de 1,5 à 2 fois supérieure. A.S. SHUBAT et al., dans [SHU 87], ont étudié la réalisation d'une UAL de type CMOS domino. Ceci donne un gain en vitesse de 30 % par rapport à une UAL de type CMOS statique.

2.3.1. Schéma de principe

L'UAL de référence peut être transformée en UAL de type domino selon le principe donné en figure 3.12.

Pour le bloc combinatoire, nous avons :

$Oi = ai + (CM3 . \overline{bi} + CM4 . bi)$

$Ei = ai . (CM1 . bi + CM2 . \overline{bi})$

Ces deux fonctions peuvent être traduites en schéma domino selon la figure 3.13.

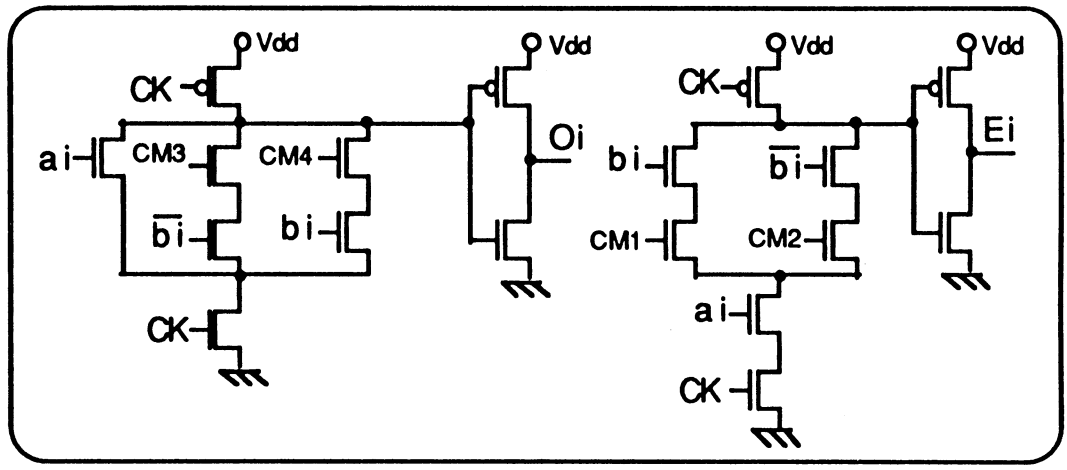


Figure 3.13 : schéma électrique réalisant les deux fonctions Oi et Ei.

Ce bloc est composé de 18 transistors et 52 transistors équivalents.

On a également pour le bloc calculant la retenue sortante :

$$Ci+1 = CM5 . Ci + \overline{CM5} (Ei + Oi . Ci)$$

Cette équation est traduite par le schéma électrique donné en figure 3.14.

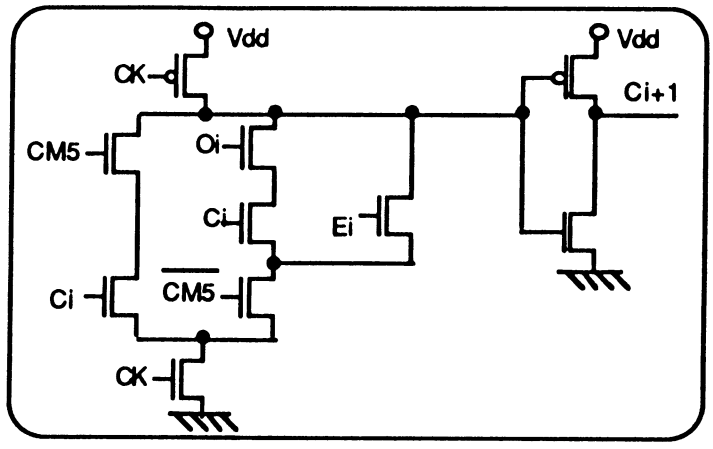


Figure 3.14 : schéma électrique du bloc de la retenue.

Ce bloc contient 10 transistors et 33 transistors équivalents.

Le bloc de sortie est réalisé en CMOS pour deux raisons :

- son influence sur la rapidité de cette UAL est négligeable,
- les fonctions complémentaires sont utilisées.

La fonction logique de la sortie d'une tranche est donnée par la formule suivante :

$$Si = \overline{Oi \cdot \overline{Ei} \oplus (Ci + CM5)}$$

Le schéma logique du bloc de sortie est donné en figure 3.15.

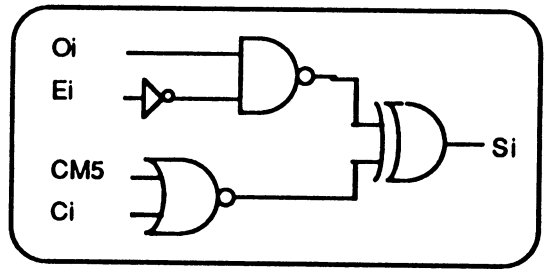


Figure 3.15 : schéma logique du bloc de sortie.

Ce bloc contient 16 transistors et 30 transistors équivalents.

2.3.2. Complexité

D'après les évaluations précédentes, on obtient pour une tranche de l'UAL, 44 transistors et 115 transistors équivalents.

Pour l'UAL complète, on a 704 transistors et 1840 transistors équivalents. Ceci donne un gain de surface d'environ 15 % par rapport à l'UAL de référence.

2.3.3. Vitesse

La structure de cette UAL est de type "Ripple Carry Adder", le gain de vitesse est obtenu au niveau de la cellule de base. Le temps de propagation estimé est donné par la formule :

$$\tau_{max} = \tau_{BC} + 15\,\tau_{BCR} + \tau_{BS} = 4 + 15 \times 4 + 10 = 74 \text{ ns}$$

La fréquence de calcul est d'environ 13 MHZ, soit 2 fois plus rapide que la fréquence de l'UAL de référence (le temps de précharge est compté à part).

2.3.4. Testabilité et reconfigurabilité

Cette méthode d'accélération de la retenue ne modifie pas les fonctions employées. Donc cette UAL est testable et reconfigurable comme l'UAL initiale.

2.3.5. Conclusion

La méthode "DOMINO" appliquée à l'UAL de référence, avec une diminution

de surface de 15% a un gain de vitesse de 2 fois par rapport à l'UAL de référence (le temps de précharge n'est pas pris en compte). Elle est par contre aisément testable et reconfigurable. Une horloge supplémentaire est indispensable.

2.4. UAL de type "MANCHESTER"

Pour améliorer la performance de l'UAL initiale, la chaîne de retenue de Manchester peut être utilisée [PAN88].

2.4.1. Présentation des principes de génération et de propagation de la retenue

Le principe de génération est basé sur une constatation très simple : lors de l'addition de deux nombres A = an-1 an-2...ai...a1a0 et B = bn-1 bn-2...bi...b1b0, si ai = bi, il n'est pas nécessaire de connaître la retenue entrante ci pour calculer la retenue sortante ci+1. En effet :

si ai = bi = 0, alors ci+1 = 0,

si ai = bi = 1, alors ci+1 = 1.

Le principe de propagation complète celui de génération. Dans le cas où ai ≠ bi, nous avons ci+1 = ci. La retenue est alors propagée.

D'après les deux notions précédentes, deux variables Pi et Gi peuvent être définies afin d'accélérer la retenue :

Pi = ai ⊕ bi (aptitude du ième rang à propager la retenue)

et Gi = ai bi (aptitude du ième rang à générer la retenue).

2.4.2. Principe de fonctionnement

L'UAL "Manchester" utilise l'additionneur de type Manchester pour effectuer les opérations énoncées au début de ce chapitre.

Cette méthode d'accélération de retenue est basée sur le principe de génération et de propagation de la retenue.

Le schéma de principe de la chaîne de retenue Manchester est donné en figure 3.16 [MEA 80] [WES 85].

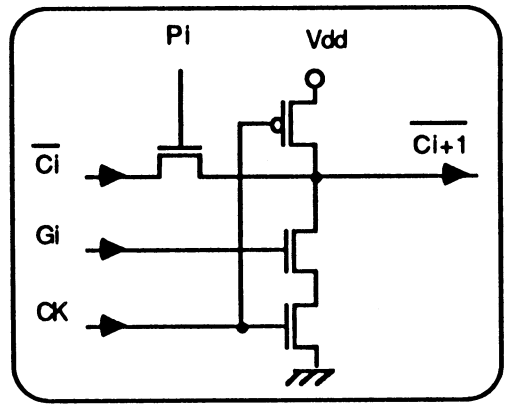


Figure 3.16 : chaîne de retenue Manchester.

Lorsque CK passe à 0, la chaîne de retenue est préchargée à 1. Lorsque CK passe à 1, la phase d'évaluation commence. Un schéma simplifié de l'UAL de type Manchester est donné en figure 3.17.

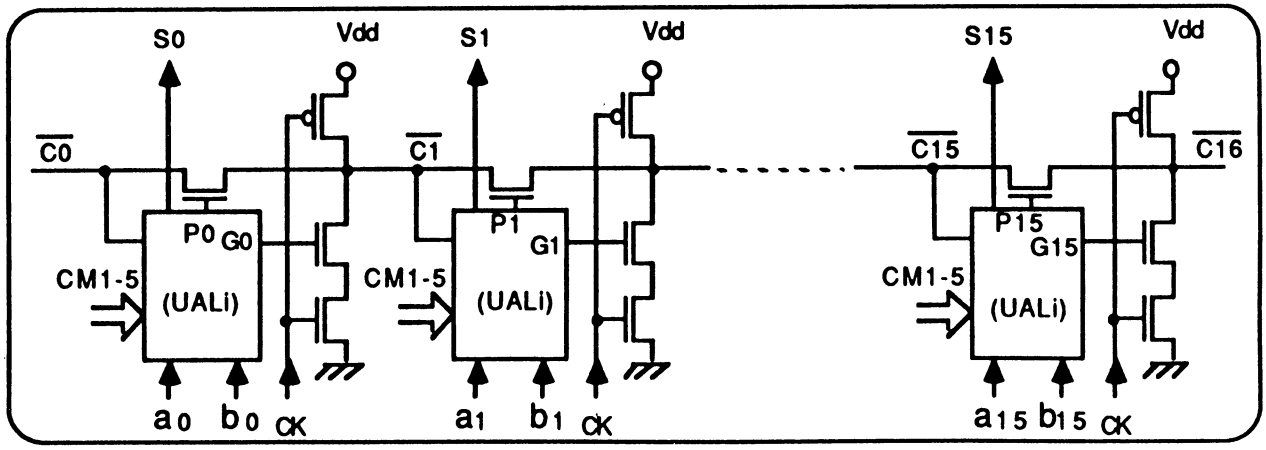


Figure 3.17 : schéma simplifié de l'UAL Manchester.

Il faut noter que la décharge complète de la ligne de retenue correspondant à $P_i = 1$ pour tout i avec Co = 1 revient au schéma de la figure 3.18.

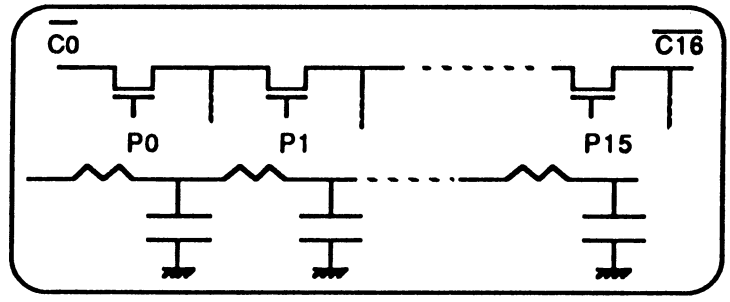


Figure 3.18 : correspondance RC de la chaîne de transistors .

La vitesse de l'UAL dépendra principalement de la facilité à décharger la ligne de retenue. Dans le but de faciliter la décharge, deux techniques peuvent être utilisées : la première consiste à calculer $\prod Pi$ pour 4 tranches. Lorsque $\prod Pi = 1$, la retenue entrante saute ces dernières selon le schéma de principe donné en figure 3.19.

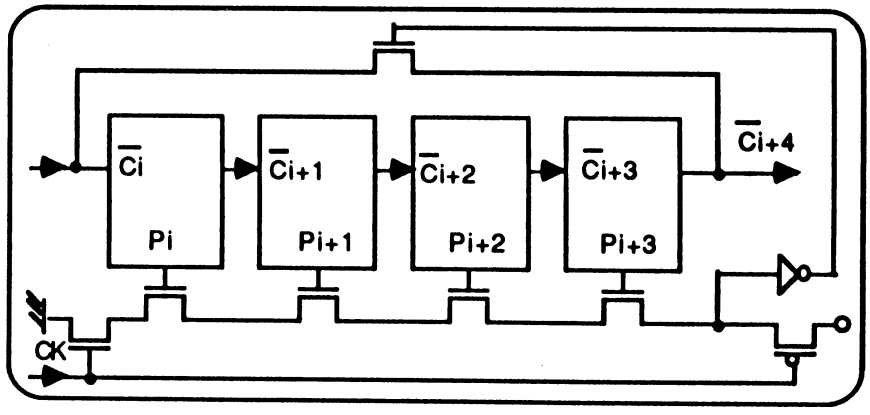


Figure 3.19 : technique d'accélération de la retenue.

La seconde consiste à utiliser un circuit de décharge inclus dans chaque tranche de l'UAL (cf. figure 3.20).

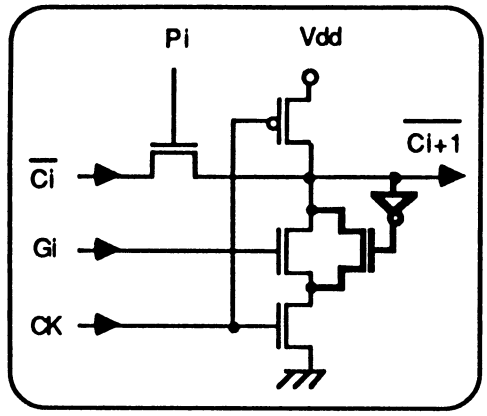


Figure 3.20 : circuit de décharge de la ligne de retenue.

Ce circuit détecte la décharge de la ligne, l'amplifie et regénère un niveau zéro pendant la phase d'évaluation (CK = 1).

Pour évaluer la performance de cette UAL, on donne le schéma d'une tranche (Figure 3.21).

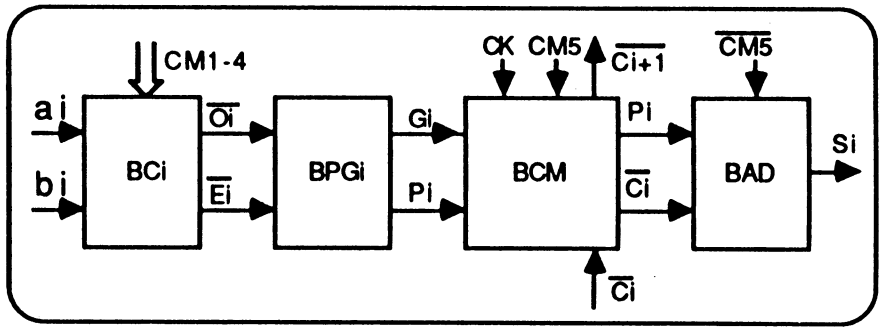


Figure 3.21 : schéma d'une tranche de l'UAL du type Manchester.

Cette UAL est composée de quatre blocs (BC, BPG, BCM, BAD). Le bloc BCi sélectionne l'opération à effectuer au moyen des commandes CM1-4, fournissant les deux variables Oi et Ei vues au début de ce chapitre. Le bloc BPGi génère les deux termes Pi et Gi à partir des deux variables Oi et Ei selon les formules :

$Pi = \overline{Ei} \cdot Oi$ avec $Ei = ai\ bi$, $Oi = ai + bi$

$Gi = Ei$

Le schéma logique de ce bloc, déduit des équations précédentes, est donné en figure 3.22.

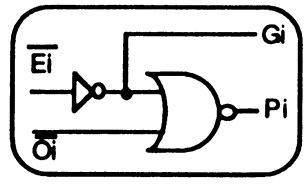


Figure 3.22 : schéma logique du bloc BPG.

Le bloc BCMi génère la retenue sortante Ci + 1 en fonction des deux termes Pi, Gi et de la retenue entrante Ci. Le schéma électrique de ce bloc est donné en figure 3.23.

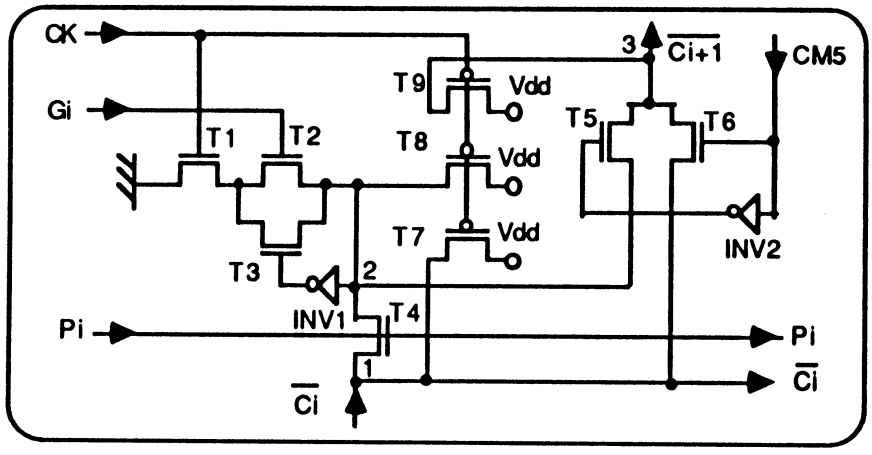


Figure 3.23 : schéma électrique du bloc BCM.

Ce bloc détermine la vitesse de l'UAL. Les transistors T7, T8 et T9 servent à précharger les nœuds 1, 2 et 3 pendant la phase de précharge (CK=0). Les transistors T1, T2, T3, T4 et l'inverseur INV1 constituent la chaîne de retenue Manchester avec le circuit de décharge.

Il reste 2 transistors (T5, T6) formant le multiplexeur qui sélectionne soit la retenue entrante Ci si la commande CM5 = 1 (opération logique) quels que soient Gi et Pi, soit la retenue calculée pour une opération

arithmétique (CM5 = 0).

Le dernier bloc de l'UAL est BAD, son schéma logique est présenté en figure 3.24.

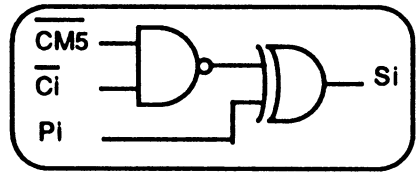


Figure 3.24 : schéma logique du bloc BAD.

2.4.3. Complexité

Cette UAL est constituée de 16 tranches, chacune de ces tranches contient 4 blocs comportant le nombre de transistors et le nombre de transistors équivalents indiqués ci-dessous.

bloc	transistors	transistors équivalents
BCi	36	72
BPGi	6	13
BCMi	13	21
BADi	10	17
Σ	65	123
16 Σ	1040	1968

Cette méthode entraîne une diminution de surface d'environ 10 % par rapport à l'UAL de référence.

2.4.4. Vitesse

Le temps maximum de calcul se décompose selon la formule suivante :

$$\tau_m = \tau_{BCi} + \tau_{BPGi} + 15\,\tau_{BCi} + \tau_{BAD}$$

$$\tau_m = 10 + 5 + 30 + 7{,}5 = 52{,}5$$

Ce temps de calcul estimé ne tient pas compte de la précharge. Maintenant, la fréquence de calcul est d'environ 19 MHZ, soit 2,9 fois plus rapide que l'UAL de référence.

2.4.5. Testabilité

Cette UAL possède une structure de type itératif comme l'UAL de référence. De plus, le tableau et le graphe des états sont identiques à ceux de l'UAL de référence. Elle est C-testable au même titre que l'UAL initiale.

2.4.6. Reconfigurabilité

La reconfiguration de cette UAL se fait de la même façon que pour l'UAL initiale. En effet, seule la structure interne d'une tranche a été modifiée. Le principe de la reconfiguration d'une tranche est donné en figure 3.25.

Les trois transistors de reconfiguration jouent le rôle des portes de transfert. En effet, seul le niveau "0" de la retenue entrante peut être propagé. Le transistor N ne dégrade que le niveau 1, donc ce transistor peut remplacer une porte de transfert. Ceci nécessite une augmentation de surface avec une seule tranche de réserve d'environ 15 %.

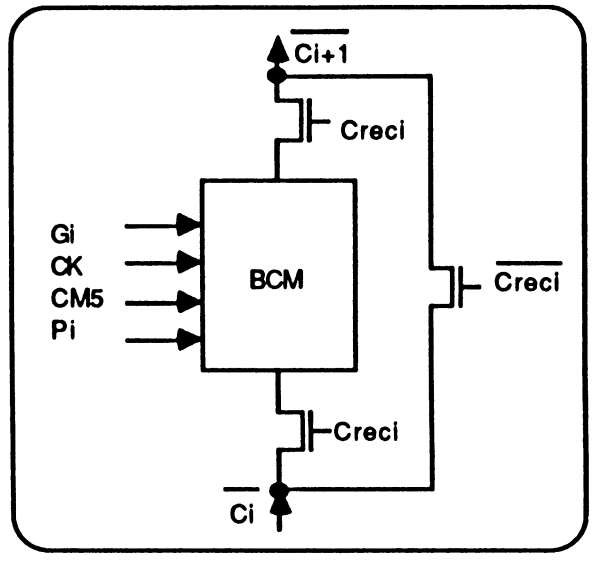


Figure 3.25 : schéma de principe de la reconfiguration.

2.4.7. Conclusion

La méthode "MANCHESTER" bien que nécessitant une horloge de précharge, offre une augmentation de vitesse satisfaisante (2,9 fois la vitesse de l'UAL initiale) pour une légère diminution de surface (10%). Elle est également facilement testable et reconfigurable.

3.UAL utilisant les techniques d'accélération de la retenue

Deux types d'UAL utilisant les techniques d'accélération de la retenue seront étudiés. Le premier est l'UAL "Carry Select" et le second est l'UAL "Carry-Skip"

3.1. UAL de type "Carry Select Adder"

3.1.1. Principe et schéma général

Cette méthode d'anticipation de la retenue a été étudiée par BEDRIJ [BED62] et HWANG [HWA79]. Dans le cas d'une UAL 16 bits, elle consiste à diviser les 16 tranches de bits en 4 groupes de 4 tranches (figure 3.26).

Chacun de ces groupes de 4 bits est doublé; l'un effectue l'opération avec une retenue entrante nulle, l'autre avec une retenue entrante égale à 1. Les huit blocs calculent ainsi parallèlement. A l'aide de multiplexeurs commandés par des blocs de sélection, on sélectionne les bons blocs et la somme exacte.

La commande de sélection du premier groupe est évidemment la retenue entrante de l'UAL (C0). La sélection du second groupe est effectuée par la retenue sortante exacte du premier (C4), donnée par l'équation logique suivante :

$C4 = C^0_4 + C0\ C^1_4$, avec :

- C^0_4 est la retenue sortante du premier groupe recevant une retenue entrante égale à 0,

- C^1_4 est la retenue sortante du premier groupe recevant une retenue entrante égale à 1.

La retenue exacte du $2^{ème}$ groupe C8 sélectionne le résultat du calcul du $3^{ème}$ groupe.

Nous avons alors :

$C8 = C^0_8 + C^0_4\ C^1_8 + C0\ C^1_4\ C^1_8$, avec :

- C^0_8 est la retenue sortante du deuxième groupe recevant une retenue entrante égale à 0,

- C^1_8 est la retenue sortante du deuxième groupe recevant une retenue

entrante égale à 1.

Le dernier groupe est sélectionné par :

$C12 = C^0_{12} + C^0_8 C^1_{12} + C^0_4 C^1_8 C^1_{12} + C0\, C^1_4 C^1_8 C^1_{12}$, avec :

- C^0_{12} est la retenue sortante du troisième groupe recevant une retenue entrante égale à 0,

- C^1_{12} est la retenue sortante du troisième groupe recevant une retenue entrante égale à 1.

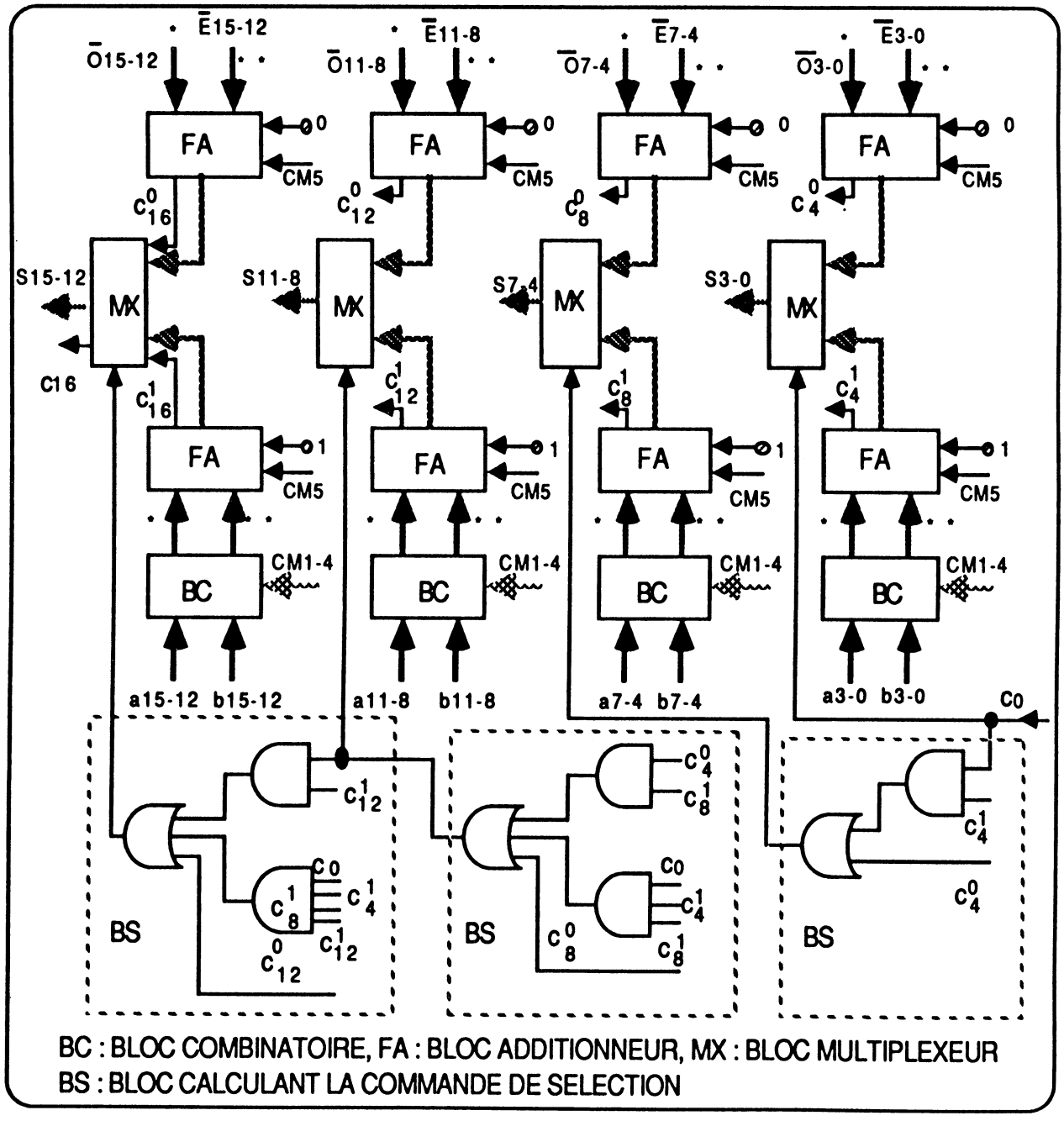


Figure 3.26 : UAL de type "Carry Select Adder"

Le schéma complet de l'UAL (figure 3.26) utilisant cette méthode est composé des blocs suivants :

- 4 groupes de blocs combinatoires (BC) calculant les fonctions Eib et Oib,
- 4 groupes de blocs additionneurs (FA), ayant chacun une retenue entrante à 0.
- 4 groupes de blocs additionneurs (FA), ayant chacun une retenue entrante à 1.
- 4 groupes de multiplexeurs (MX), chacun étant constitué par 4 multiplexeurs à 2 entrées.
- 3 blocs (BS) de calcul de la commande de sélection des multiplexeurs.

Cette UAL offre une possibilité d'implantation en tranche (bit-slice), sauf en ce qui concerne les blocs (BS) destinés à calculer les commandes de sélection des multiplexeurs.

3.1.2. Complexité

Cette UAL contient :

- 16 tranches de génération de Oib et Eib, soit 576 transistors et 1152 transistors équivalents,
- 32 tranches d'additionneur 1 bit, soit 1216 transistors et 2016 transistors équivalents,
- 46 transistors pour les multiplexeurs et 96 transistors équivalents,
- 56 transistors pour les blocs de sélection et 134 transistors équivalents,

soit au total : 1894 transistors et 3398 transistors équivalents.

L'augmentation de surface par rapport à l'UAL de référence est d'environ 60 %.

3.1.3. Vitesse

Le temps maximum de calcul d'une addition correspond au temps de calcul de S15, S14, S13, S12 et se décompose en :

- temps de retard dû au BC, soit 10 ns,
- temps de retard dû au FA, soit 36 ns,
- temps de retard dû au BS, soit 6 ns,
- temps de retard dû au MPX, soit 4 ns.

Ceci donne au total 56 ns, c'est-à-dire une fréquence de calcul de l'ordre de 18 MHZ, donc 2,8 fois plus rapide que la première solution.

3.1.4. Testabilité

Le test d'une telle UAL est partiellement possible par la mise en série, avec une logique appropriée, des blocs de 4 bits (figure 3. 27).

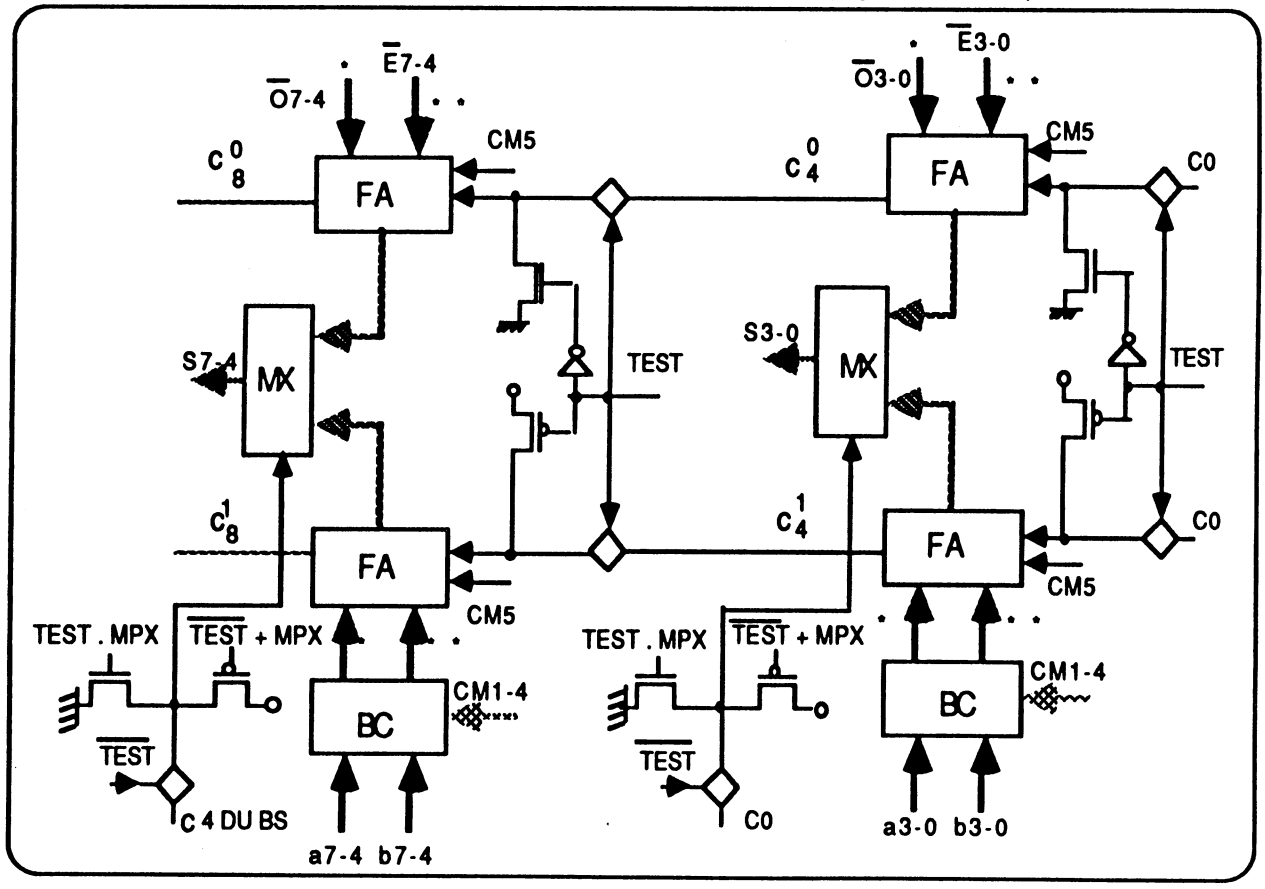


Figure 3. 27 : principe de test de l'UAL.

Le mode de fonctionnement utilisant deux signaux de test (TEST et MPX) peut être résumé dans le tableau 3.7.

TEST	MPX	MODE DE FONCTIONNEMENT	REMARQUES
0	X	NORMAL	
1	0	TEST	FA (RETENUE=1)
1	1	TEST	FA (RETENUE=0)

Tableau 3.7 : mode de fonctionnement.

La commande TEST mettrait alors en série les 4 blocs (FA) recevant, dans le mode de fonctionnement normal, des retenues entrantes à 0, ainsi que les 4 autres blocs (FA) recevant des retenues entrantes à 1. En commandant directement les multiplexeurs par MPX, on obtient deux structures de type ILA, similaires à l'UAL de référence, et donc facilement testables. 240 vecteurs de test sont nécessaires pour tester le circuit à l'exception de la logique de commande des multiplexeurs.

Le test des blocs générant les commandes de sélection nécessite des vecteurs de test supplémentaires.

Nous constatons que le test est possible, mais nécessite des dispositifs supplémentaires, ainsi que deux commandes spécifiques.

3.1.5. Reconfigurabilité

La reconfiguration nécessite l'adjonction d'un bloc opérant sur 4 bits. Cet ajout amène une augmentation de surface non négligeable d'environ 30%. De plus, le contournement d'un bloc défectueux et son remplacement par un bloc redondant est trop compliqué, car les blocs générant les commandes de sélection ne sont pas identiques.

3.1.6. Conclusion

L'UAL utilisant la méthode "Carry Select" est de 2,8 fois plus rapide que l'UAL de référence (la fréquence de calcul est de 20 MHZ) et a une surface supplémentaire de 60%.

Le test de l'UAL est possible et nécessite de la logique appropriée avec deux commandes supplémentaires.

La reconfiguration de l'UAL exige environ 30% de la surface supplémentaire sans parler de sa compléxité.

3.2. UAL "Carry-Skip"

3.2.1. Schéma de principe

L'UAL "Carry-Skip" utilise les principes de génération et de propagation de la retenue. L'idée générale de la méthode "Carry-Skip", proposée par M. LEHMAN et N. BURLA dans [LEH61], consiste à diviser l'additionneur en blocs. Chacun de ces blocs est constitué d'un additionneur séquentiel et d'un circuit spécial permettant de détecter s'il y a propagation de la retenue à travers le bloc ($\prod P_i = 1$). La figure 3.28 présente le schéma général d'une UAL "Carry-Skip" sur 16 bits, décomposée en 4 blocs de 4 bits chacun.

Si dans un quelconque des blocs on a $P_i = a_i \oplus b_i = 0$, alors, une retenue est générée et toute la partie "aval" est ignorée. On note P, le nombre de tranches de l'UAL dans un bloc, m le nombre de blocs, T1 le temps de propagation dans une tranche de l'UAL et T2 le temps nécessaire pour que la retenue saute un bloc.

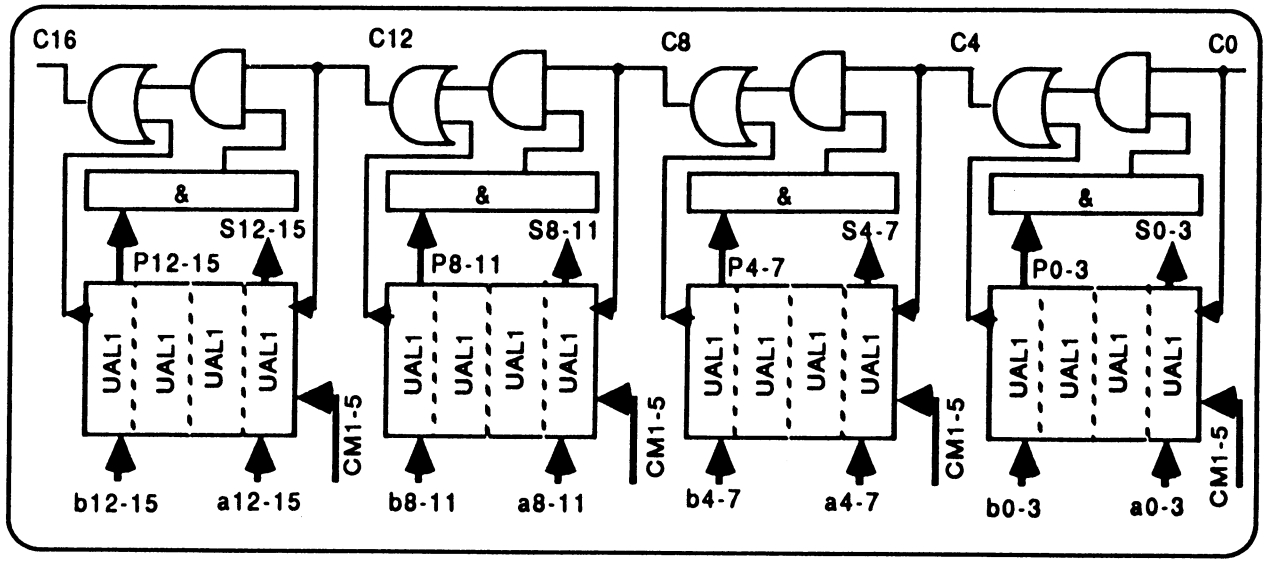


Figure 3.28 : schéma général de l'UAL "Carry-Skip".

Dans le cas des blocs de tailles égales, on cherche à minimiser le plus long temps de propagation de la retenue. Le plus long temps est atteint pour une retenue qui apparaît au début du premier bloc, qui le traverse, qui saute les m-2 blocs suivants et qui traverse le dernier. Le temps total est donné par la formule suivante :

$T(m) = 2\ T1\ P(m) + (m - 2)\ T2$, avec $p = n/m$; n est le nombre de bits traités.

Dans le but de trouver la valeur de m qui minimise T, on cherche la valeur de m qui annule la dérivée première de T(m).

$$\frac{d\ T(m)}{dm} = 0 \Longrightarrow m = \sqrt{2n\ \frac{T1}{T2}}$$

$$T_{min} = 2\ (\sqrt{2\ n\ T1\ T2} - T2\ )$$

Donc, le temps de propagation de la retenue est proportionnel à $\sqrt{n}$.

Dans [MAJ67], S. MAJERSKI a étudié le découpage en blocs de tailles distinctes, ayant pour but de minimiser les portes de saut ("Skip"). A. GUYOT, B. HOCHET et J.M. MULLER ont étudié le même problème dans [GUY87] avec l'idée de base consistant à transformer le problème de

découpage en un problème géométrique afin d'optimiser la surface globale pour un temps minimum de propagation.

Prenons l'exemple de l'UAL 16 bits avec 2 tranches de réserve. L'optimisation du temps de calcul revient à placer des carrés sous des droites de pentes ± T2/T1. Chaque carré représente un additionneur, chaque colonne représente un bloc. Le temps de calcul maximum est celui mis pour partir du bas d'une colonne, la remonter, parcourir les droites et redescendre la dernière colonne (cf. figure 3.29).

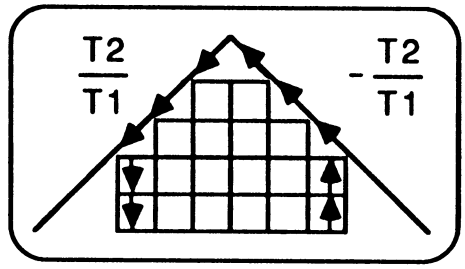


Figure 3.29 : découpage de l'UAL en blocs de tailles distinctes.

Le schéma logique de l'UAL "Carry-Skip" correspondant au découpage précédent est donné en figure 3.30.

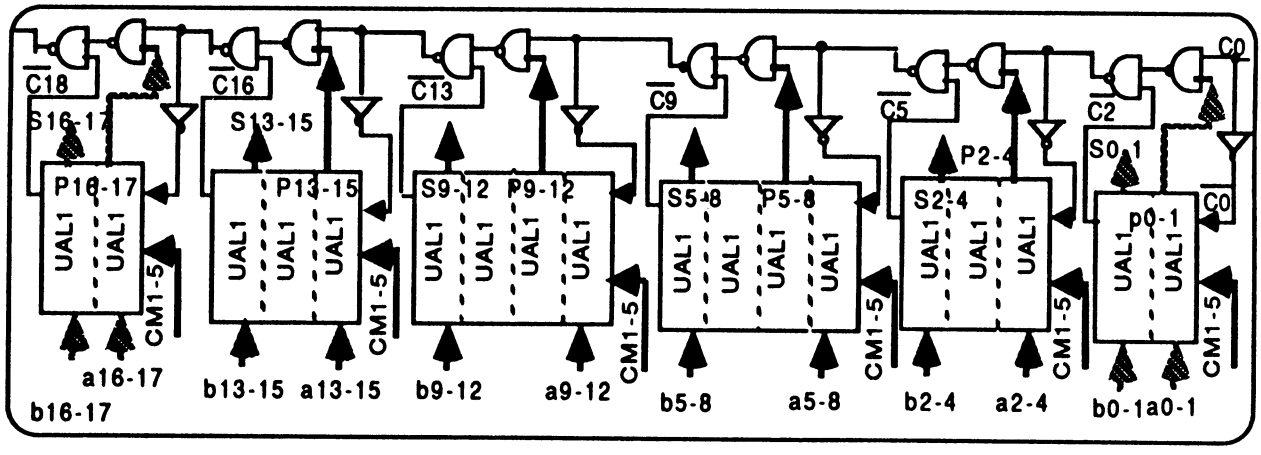


Figure 3.30 : schéma de l'UAL "Carry-Skip" en blocs de tailles distinctes.

La cellule de base UAL1 utilisée est celle de l'UAL de référence avec un peu de logique supplémentaire nécessaire pour générer les variables Pi .

Pour toutes les fonctions arithmétiques, Pi est généré par l'UAL1, Pi = Oi Eib. Par contre, pour les opérations logiques, on cherche à avoir une propagation systématique de la retenue donc il faut forcer Pi à "1". Ceci est réalisé en prenant :

$$Pi = CM5 + Oi \cdot \overline{Ei} = \overline{CM5 \cdot Oi \cdot Ei}$$

car CM5 = 1 pour toutes les opérations logiques.

Un inverseur et une porte NAND2 ont donc été rajoutés à chaque tranche pour obtenir UAL1.

3.2.2. Complexité

L'UAL "Carry-Skip" sur 16 bits est constituée :

- de 16 tranches de l'UAL1 contenant 1240 transistors et 2336 transistors équivalents,
- de la logique supplémentaire contenant 72 transistors et 168 transistors équivalents,

soit au total 1312 transistors réels et 2504 transistors équivalents. L'augmentation de surface par rapport à l'UAL de référence est d'environ 16%.

3.2.3. Vitesse

On estime la vitesse de l'UAL "Carry-Skip" pour les deux configurations possibles :

- l'UAL avec découpage en blocs de tailles égales, figure 3.28 ;

Le temps maximal de calcul de cette UAL est donné par la formule suivante :

$\tau_{max} = 2 \times 4\ T1 + 2\ T2 + \Delta T$

T1 = 9 ns et T2 = 5,5 ns, ΔT est le temps de propagation nécessaire pour calculer les variables (O_i, E_i) et vaut 10 ns.

τ_{max} = 93 ns .

La fréquence de calcul maximale est donc d'environ 10 MHZ, soit 1,5 fois plus rapide que celle de l'UAL de référence.

- l'UAL avec découpage en blocs de tailles distinctes :
  * le bloc1 contient 2 tranches,
  * le bloc2 contient 3 tranches,
  * le bloc3 contient 3 tranches,
  * le bloc4 contient 4 tranches,
  * le bloc5 contient 3 tranches,
  * le bloc6 contient 1 tranche.

Le temps maximal de calcul pour cette configuration est le suivant :

$\tau max = (3\ T1 + 4\ T2) + \Delta T$ = 59 ns.

La fréquence estimée est alors d'environ 17 MHZ, soit 2,6 fois plus rapide que l'UAL de référence.

3.2.4. Testabilité

Ce type d'UAL n'est pas C-testable sans modification. Cependant, on peut le rendre facilement testable en modifiant peu sa structure. Il s'agit de rajouter des multiplexeurs entre les blocs (cf. figure 3.31).

Ces multiplexeurs sont contrôlés par la commande TEST qui est aussi à rajouter. En effet, si la commande TEST est valide, tous les blocs "skip" sont ignorés.

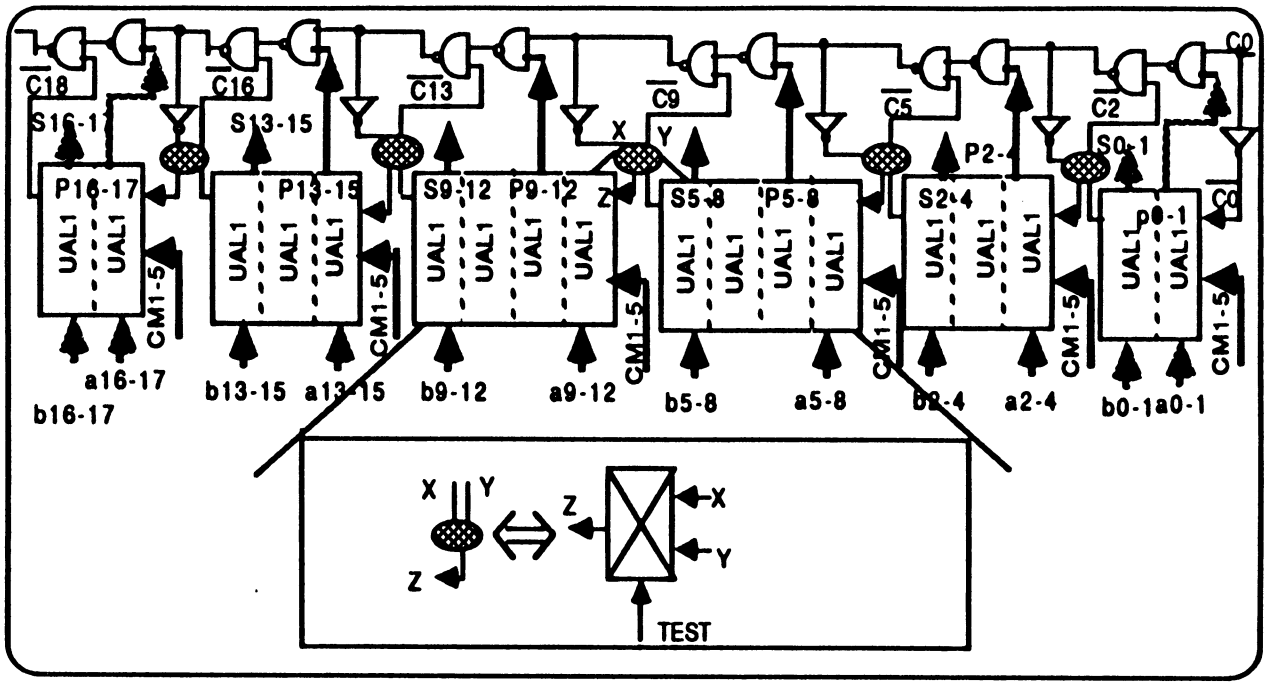


Figure 3.31 : UAL "Carry-Skip" testable.

L'UAL est alors identique à l'UAL initiale, on peut la tester avec les 146 vecteurs de test donnés en Annexe 3.1. La logique de saut rajoutée est testable avec 32 vecteurs de test donnés en annexe 3.2.

3.2.5. Reconfigurabilité

La reconfiguration de cette UAL est possible. Pour satisfaire les contraintes du cahier des charges, les additionneurs sont seulement reconfigurés selon le schéma de principe donné en figure 3.32. Cela exige l'ajout de 17 multiplexeurs pour imposer Pi =1 lorsque la tranche numéro i est défectueuse. Ceci donne une augmentation de surface pour la reconfiguration d'environ 20 %.

Il faut signaler que ni les multiplexeurs destinés au test, ni la logique de saut "Skip", ni la logique de commande ne sont reconfigurés.

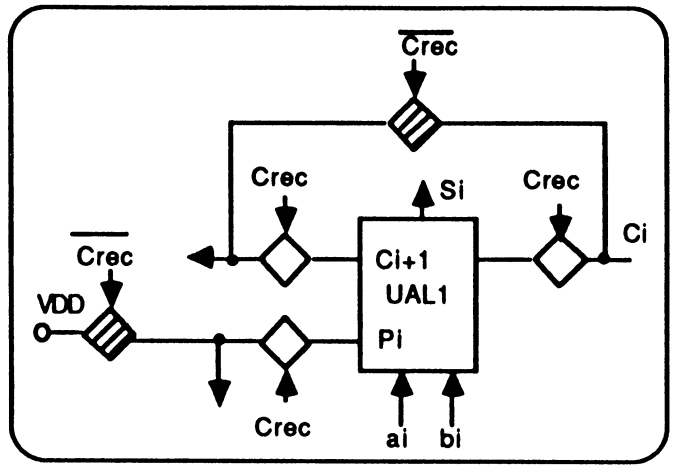


Figure 3.32 : schéma de principe de reconfiguration de l'UAL "Carry-Skip".

3.2.6. Conclusion

La méthode "Carry-Skip" pour une légère augmentation de surface (16%), offre une bonne rapidité (17 MHZ), mais par contre est moins aisément testable que l'UAL initiale et la surface non reconfigurée est assez élevée.

4.UAL anticipant la retenue

Les travaux de recherche concernant les techniques d'anticipation de la retenue et leurs applications sont nombreux [HWA79] [BRE82] [PAN88] [BEW88]. Après un rappel de ces principes, plusieurs techniques seront étudiées.

4.1. Principe d'anticipation de la retenue

Ce principe consiste à calculer à l'avance la retenue selon l'algorithme suivant plutôt que de la propager :

- l'étage i reçoit une retenue entrante égale à 1, si et seulement si :
- l'étage i - 1 a généré une retenue à 1 (Gi-1 = 1) ou
- l'étage i - 1 a propagé une retenue à 1 générée par l'étage i - 2

(Pi-1 = 1 et Gi-2 = 1) ou

- ..... ou

- les étages i - 1, i - 2, ..., 1, 0 ont propagé une retenue entrante dans l'additionneur (Pi-1 = Pi-2 = ... = P1 = P0 = 1 et C0 = 1)

On obtient donc :

Ci = Gi-1 + Gi-2 Pi-1 + Gi-3 Pi-2 Pi-1 + ... + Pi-1 Pi-2 ... P1 P0 C0

On conçoit facilement que cette méthode devienne très lourde dès que la longueur de l'additionneur dépasse 4 ou 5 étages. Nous pouvons toutefois regrouper les bits de l'additionneur par petits groupes.

Nous allons étudier quelques types d'additionneurs utilisant ces méthodes. Nous déduirons des schémas d'UAL issus de celles-ci, afin de les comparer à l'UAL de référence selon les critères énoncés au début de ce chapitre.

4.2. UAL de type "Full Carry Look Ahead Adder"

Le principe de base consiste à calculer parallèlement toutes les retenues à partir des entrées a_i, b_i, avec $0 \leq i \leq 15$, ce qui implique un temps de calcul indépendant de la longueur des opérandes (A, B).

4.2.1. Schéma de principe

A partir des principes que nous venons d'étudier, et du cahier des charges pour une UAL de 16 bits, un schéma général de type "Full Carry Look Ahead Adder" est proposé (cf. figure 3.33).

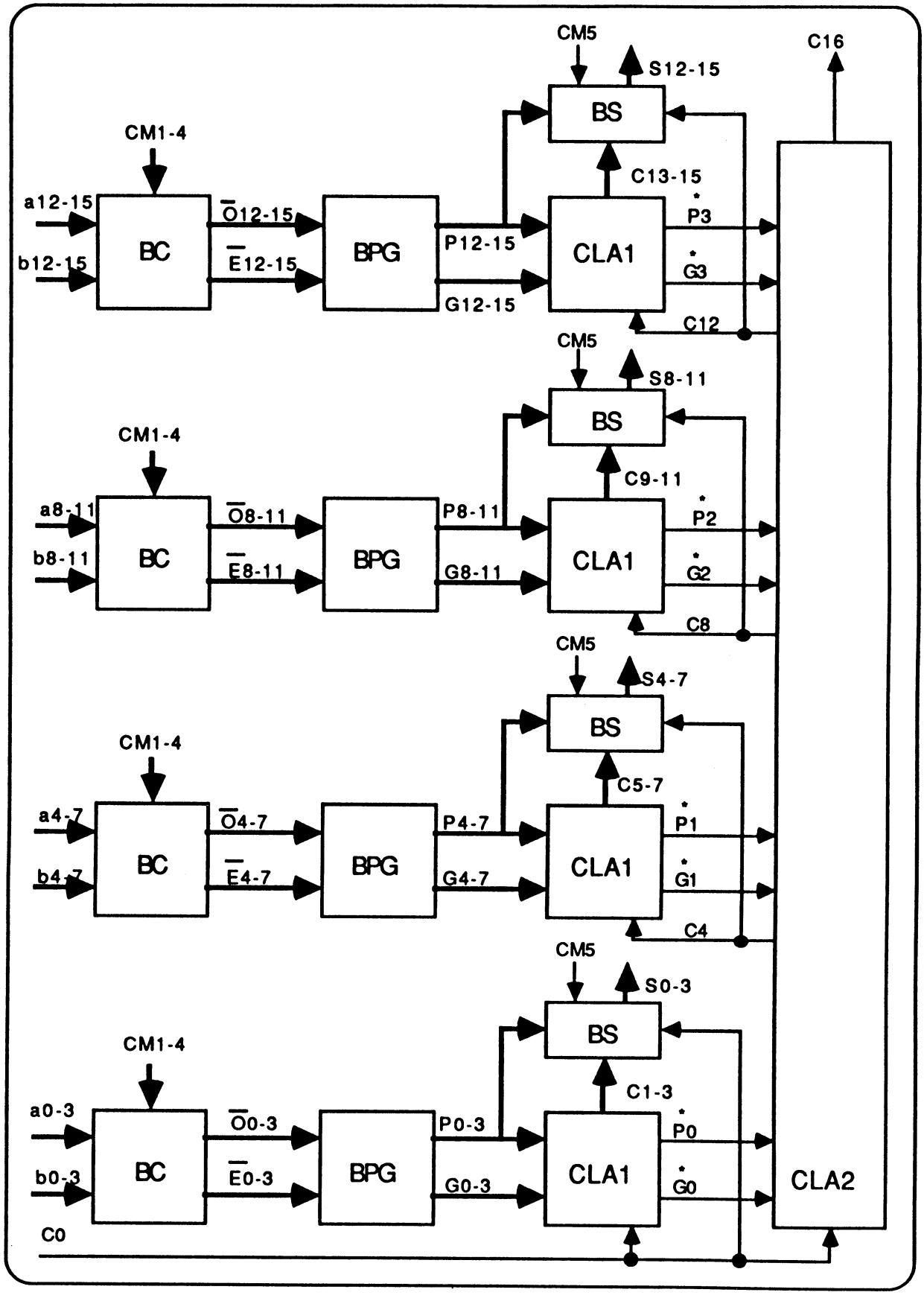


Figure 3.33 : UAL 16 bits de type "Full Carry Look Ahead Adder".

La partie additionneur est décomposée en 4 groupes de 4 bits. Chacun de ces groupes comporte 3 blocs :

- BPG calcule les variables Pi Gi ($0 \leq i \leq 3$)

$Pi = Oi \oplus Ei = ai \oplus bi$ (pour l'addition)

$Gi = Oi + Ei = ai\ bi$ (pour l'addition)

Le schéma logique déduit de ces équations est donné dans la figure 3.34.

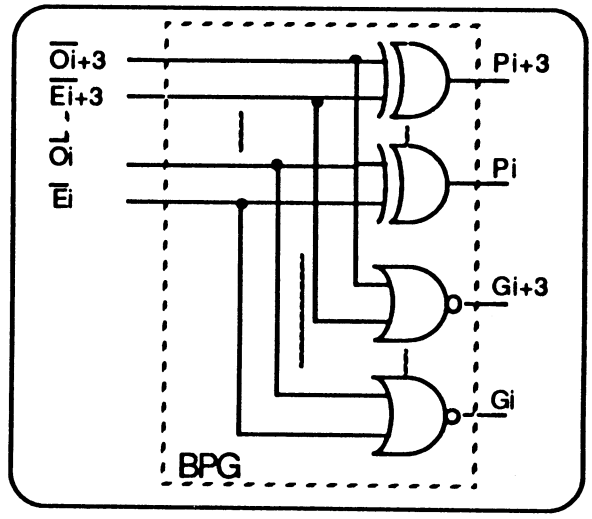


Figure 3.34 : schéma logique du bloc BPG.

- CLA1 calcule 3 retenues. Par exemple Ci+1 Ci+2 Ci+3 pour les 4 bits (ai ai+1 ai+2 ai+3, bi bi+1 bi+2 bi+3) et deux variables Pi^* et Gi^* alimentant le bloc CLA2, qui sont définies par :

$Pi^* = Pi\ Pi+1\ Pi+2\ Pi+3$

$Gi^* = Gi+3 + Gi+2\ Pi+3 + Gi+1\ Pi+2\ Pi+3 + Gi\ Pi+1\ Pi+2\ Pi+3$

Le schéma logique du bloc CLA1 est donné en figure 3.35.

- CLA2 calcule les retenues C4 C8 C12 C16 à partir des variables $P0^*$ $P1^*$ $P2^*$ $P3^*$, $G0^*$ $G1^*$ $G2^*$ $G3^*$ et C0. Son schéma logique est aussi donné en figure 3.36.

- BS calcule le résultat Si qui est défini par :

$S_i = P_i \oplus C_i$ pour les opérations arithmétiques

$S_i = P_i \oplus 1$ pour les opérations logiques

Le schéma logique est donné en figure 3.37.

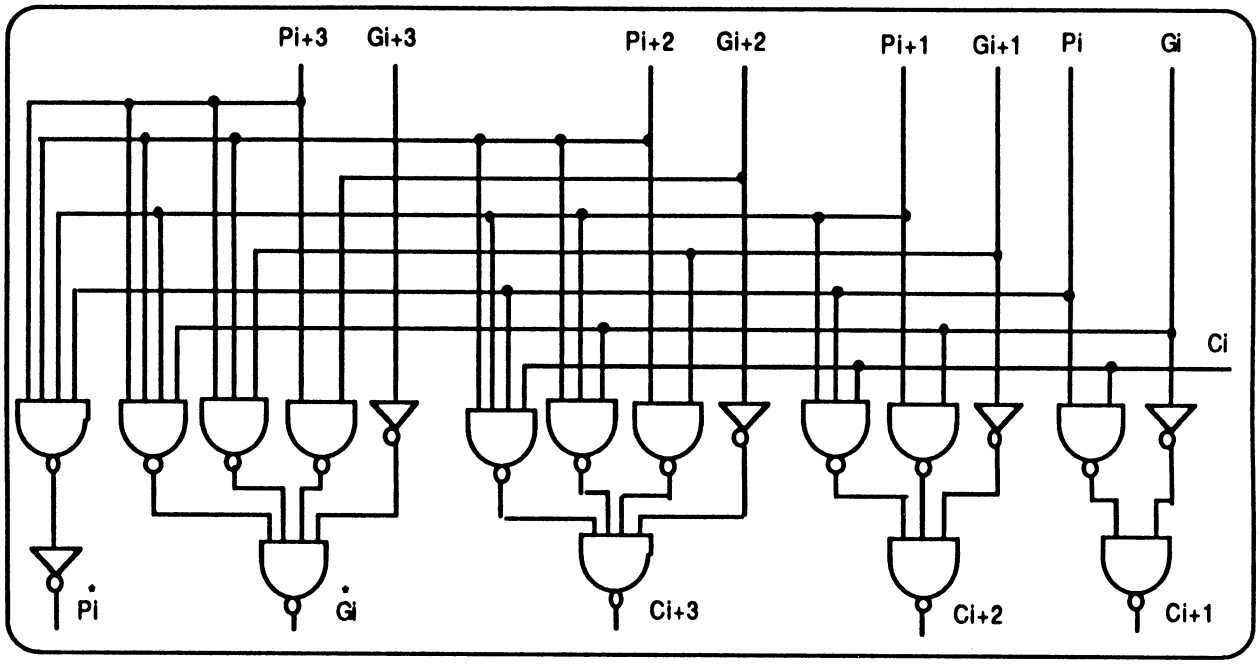


Figure 3.35 : schéma logique du bloc CLA1.

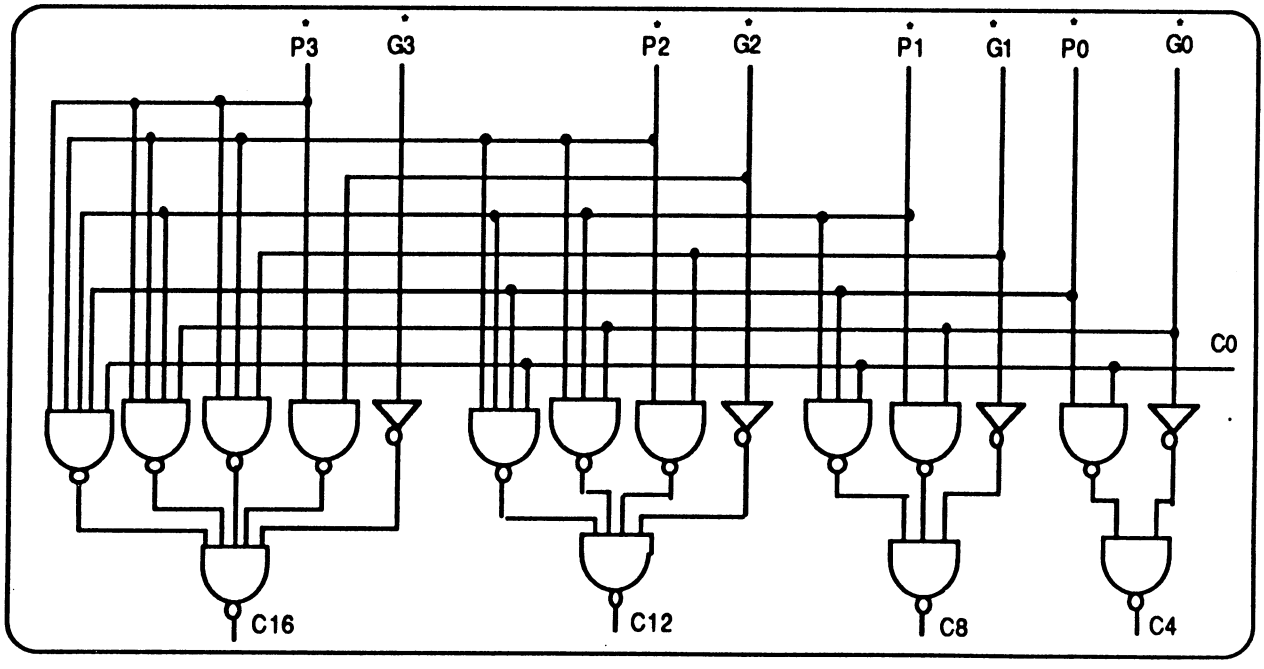


Figure 3.36 : schéma logique du bloc CLA2.

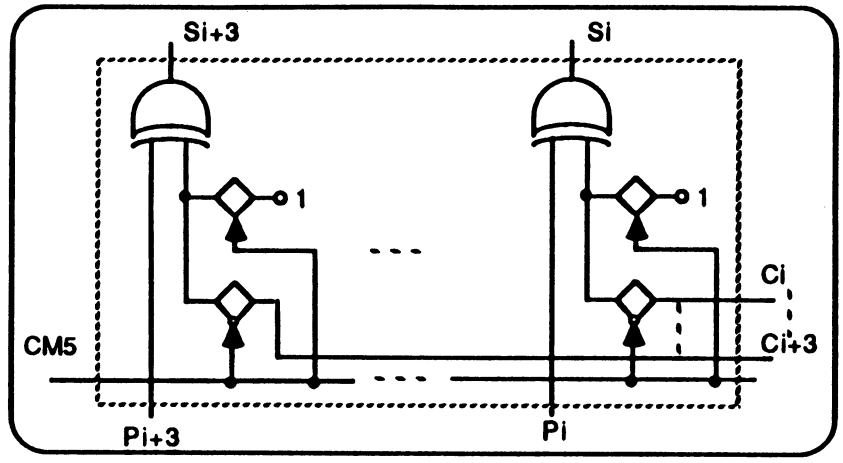


Figure 3.37 : schéma logique du bloc BS.

4.2.2. Complexité

D'après ces schémas, nous pouvons évaluer la complexité d'une telle UAL:

- les blocs BC contiennent 576 transistors et 1152 transistors équivalents

- les blocs BPG comportent :

Portes	Transistors	Transistors équivalents
4 X 4 EOR	16 X 6 = 96	16 X 9 =144
4 X 4 NOR2	16 X 4 = 64	16 X 10 = 160
Σ	160	304

- les blocs CLA1 contiennent :

portes	transistors	transistors équivalents
4 X 5 NAND4	20 X 8 = 160	20 X 24 = 480
4 X 4 NAND3	16 X 6 = 96	16 X 15 = 240
4 X 5 NAND2	20 X 4 = 80	20 X 8 = 160
4 X 5 INV	20 X 2 = 40	20 X 3 = 60
Σ	376	940

- le bloc CLA2 contient :

portes	transistors	transistors équivalents
2 NAND5	2 X 10 = 20	2 X 35 = 70

3 NAND4	3 X 8 = 24	3 X 24 = 72
4 NAND3	4 X 6 = 24	4 X 15 = 60
5 NAND2	5 X 4 = 20	5 X 8 = 40
4 INV	4 X 2 = 8	4 X 3 = 12
Σ	96	254

- les blocs BS contiennent :

portes	transistors	transistors équivalents
4 X 4 EOR2	16 X 6 = 96	16 X 9 = 144
4 X 8 PT	32 X 2 = 64	32 X 3 = 96
Σ	160	240

Au total nous avons pour l'UAL 1368 transistors et 2890 transistors équivalents.

Ceci donne un supplément de surface de 33,4 % par rapport à la surface de l'UAL de référence.

4.2.3. Vitesse

Le temps de calcul maximum est la somme de 5 termes :

$$\tau_{BC} + \tau_{BPG} + 2\,\tau_{BCLA1} + \tau_{BCLA2} + \tau_{BS} = 10\text{ ns} + 5\text{ ns} + 12\text{ ns} + 6\text{ ns} + 9\text{ns}$$

$$= 42\text{ ns}$$

La fréquence de calcul est alors 23,8 MHZ, ce qui revient à dire que la vitesse de cette UAL est 3,6 fois plus rapide que celle de l'UAL de référence.

4.2.4. Testabilité

Etant donnée la structure de cette UAL, il est clair qu'elle n'est pas de type C-testable, ni même ILA. Les blocs rajoutés ne permettent plus de

considérer l'UAL comme 16 cellules identiques.

Par conséquent, pour tester l'UAL, il est nécessaire de rentrer toutes les combinaisons d'entrées possibles. De plus, il n'est pas évident qu'une erreur détectée puisse être localisée et nous pouvons donc conclure que ce circuit n'est pas testable selon nos critères.

4.2.5. Reconfigurabilité

Du point de vue reconfiguration, il est difficile d'utiliser ce type d'additionneur. Ceci nécessiterait une logique supplémentaire énorme. Toutefois, en supposant qu'il soit possible de le tester et de localiser une erreur, nous pouvons le reconfigurer en rajoutant quatre tranches redondantes, ainsi qu'un bloc CLA1 et en modifiant le bloc CLA2 en le doublant.

Ceci demanderait environ 40 % de place supplémentaire sans parler de la complexité d'une telle reconfiguration.

4.2.6. Conclusion

La méthode "Full Carry Look Ahead " est plus rapide que celle de l'UAL initiale (23,8 MHZ). Elle nécessite par contre une surface supplémentaire de 33,4% par rapport à l'UAL initiale.

Cette UAL n'est pas facilement testable et est surtout gourmande en surface nécessaire pour la reconfiguration.

4.3. UAL de type "Ripple Within Groups, Look Ahead Between Groups Adder"

L'UAL de type "Ripple Within Groups, Look Ahead Between Groups Adder"

utilise la même approche que précédemment en modifiant son schéma de base [MAC 61].

4.3.1. Schéma de principe

Cette approche consiste à calculer les retenues (C4, C8, C12, C16) entre les groupes à l'aide des blocs CLA3 (remplaçant les blocs CLA1) et le bloc CLA2 puis laisser ces retenues se propager dans chacun des groupes.

Le schéma général de l'UAL utilisant ce principe est donné en figure 3.40. Le schéma logique des blocs CLA3 qui génèrent les variables P* et G* à partir des Pi et Gi est donné dans la figure 3.38.

Chacun des blocs (BSM) est constitué de 3 additionneurs (AD) calculant la somme et la retenue sortante et 1 additionneur (ADF) qui donne seulement la somme (figure 2.39).

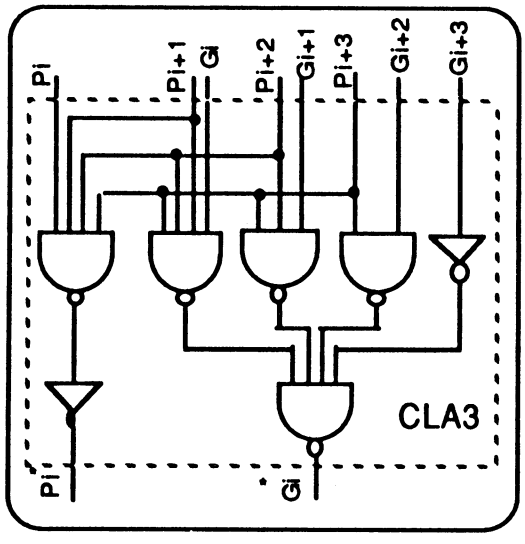


Figure 3.38 : schéma logique du bloc CLA3

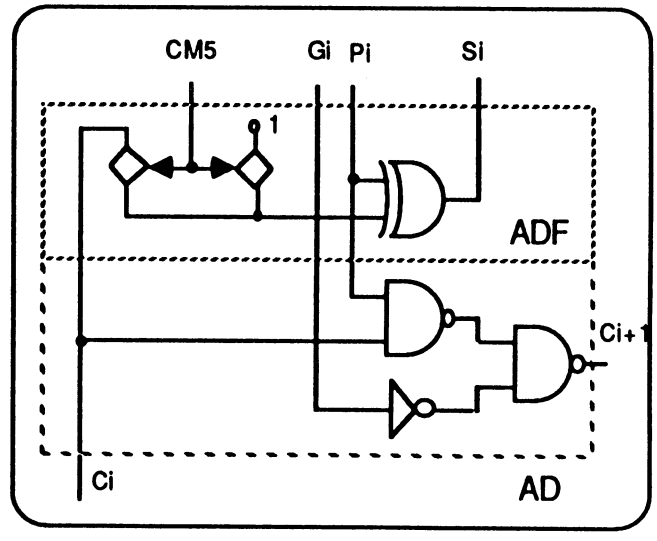


Figure 3.39 : schéma logique du AD et ADF

4.3.2. Complexité

Les blocs BC, BPG et CLA2 ont déjà été évalués. Les blocs CLA3

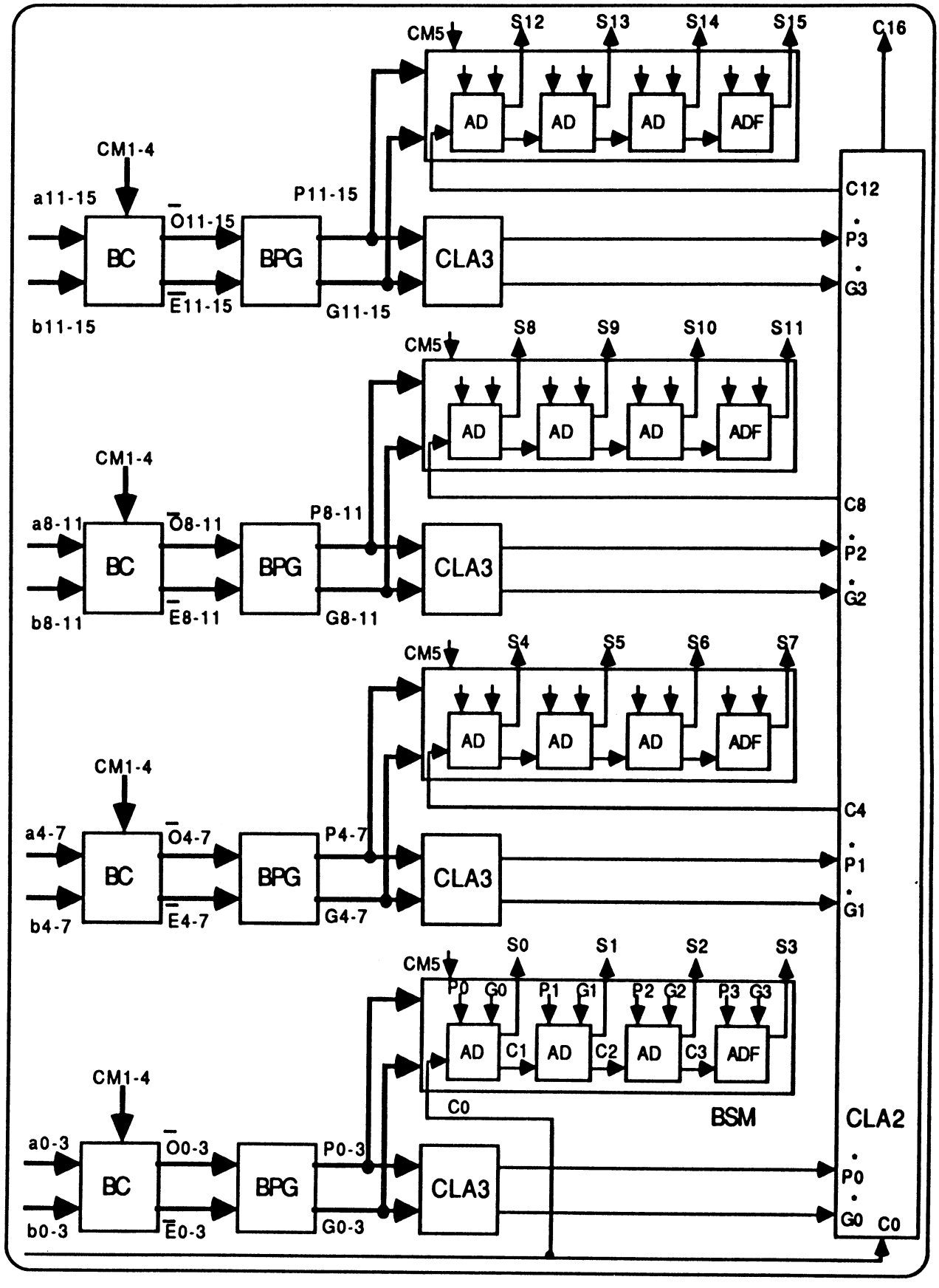


Figure 3.40 : UAL 16 bits à base "Ripple Within Groups, Look Ahead Between Groups Adder".

comprennent :

portes	transistors	transistors équivalents
4 X 3 NAND4	12 X 8 = 96	12 X 24 = 288
4 X 1 NAND3	4 X 6 = 24	4 X 15 = 60
4 X 1 NAND2	4 X 4 = 16	4 X 8 = 32
4 X 2 INV	8 X 2 = 16	8 X 3 = 24
Σ	152	404

Les blocs BSM contiennent :

portes	transistors	transistors équivalents
4 X 3 X 2 PT	24 X 2 = 48	24 X 3 = 72
4 X 3 X 1 EOR	12 X 6 = 72	12 X 9 = 108
4 X 3 X 2 NAND2	24 X 4 = 96	24 X 8 = 192
4 X 3 X 1 INV	12 X 2 = 24	12 X 3 = 36
4 X 1 X 2 PT	8 X 2 = 16	8 X 3 = 24
4 X 1 X 1 EOR	4 X 6 = 24	4 X 9 = 36
Σ	280	468

Pour l'UAL, on a 1264 transistors et 2582 transistors équivalents, soit 19,5 % de surface supplémentaire par rapport à l'UAL initiale.

4.3.3. Vitesse

Le temps maximum de calcul (τ) peut être estimé en utilisant la formule:

$$\tau = \tau_{BC} + \tau_{BPG} + \tau_{CLA3} + \tau_{CLA2} + \tau_{BSM}$$

τ = 10 ns + 5 ns + 6 ns + 6 ns + (15 + 9) ns = 51 ns.

La fréquence maximale de fonctionnement de l'UAL est d'environ 19,6 MHZ, donc 3 fois plus rapide que l'UAL classique.

4.3.4. Testabilité

La testabilité de cette UAL est identique à la testabilité de l'UAL de type "Full Carry Look Ahead". Elle n'est donc pas testable selon nos critères.

4.3.5. Reconfigurabilité

La reconfigurabilité de cette UAL est aussi identique à celle de l'UAL précédente, considérant la surface importante à ajouter et la complexité d'une telle reconfiguration.

Donc, la reconfiguration de cette UAL ne correspond pas à nos critères et n'est, par conséquent, pas envisageable.

4.3.6. Conclusion

La méthode "Ripple Within Groups, Look Ahead Between Groups" est plus rapide que celle de l'UAL initiale (19,6 MHZ). Elle nécessite par contre une surface supplémentaire de 19,5% par rapport à l'UAL initiale.

Cette UAL n'est pas facilement testable et est surtout gourmande en surface nécessaire à la reconfiguration.

4.4. UAL de type "Carry Look Ahead Within Groups, Ripple Between Groups Adder"

4.4.1. Principe

Ce principe est dérivé du "Full Carry Look Ahead Adder" et consiste à diviser l'UAL en quatre groupes où l'on réalise l'anticipation des retenues

que l'on propage ensuite entre ces blocs. Le schéma de principe de cette UAL est donné en figure 3.41. Un tel schéma nous permet d'évaluer les performances de l'UAL.

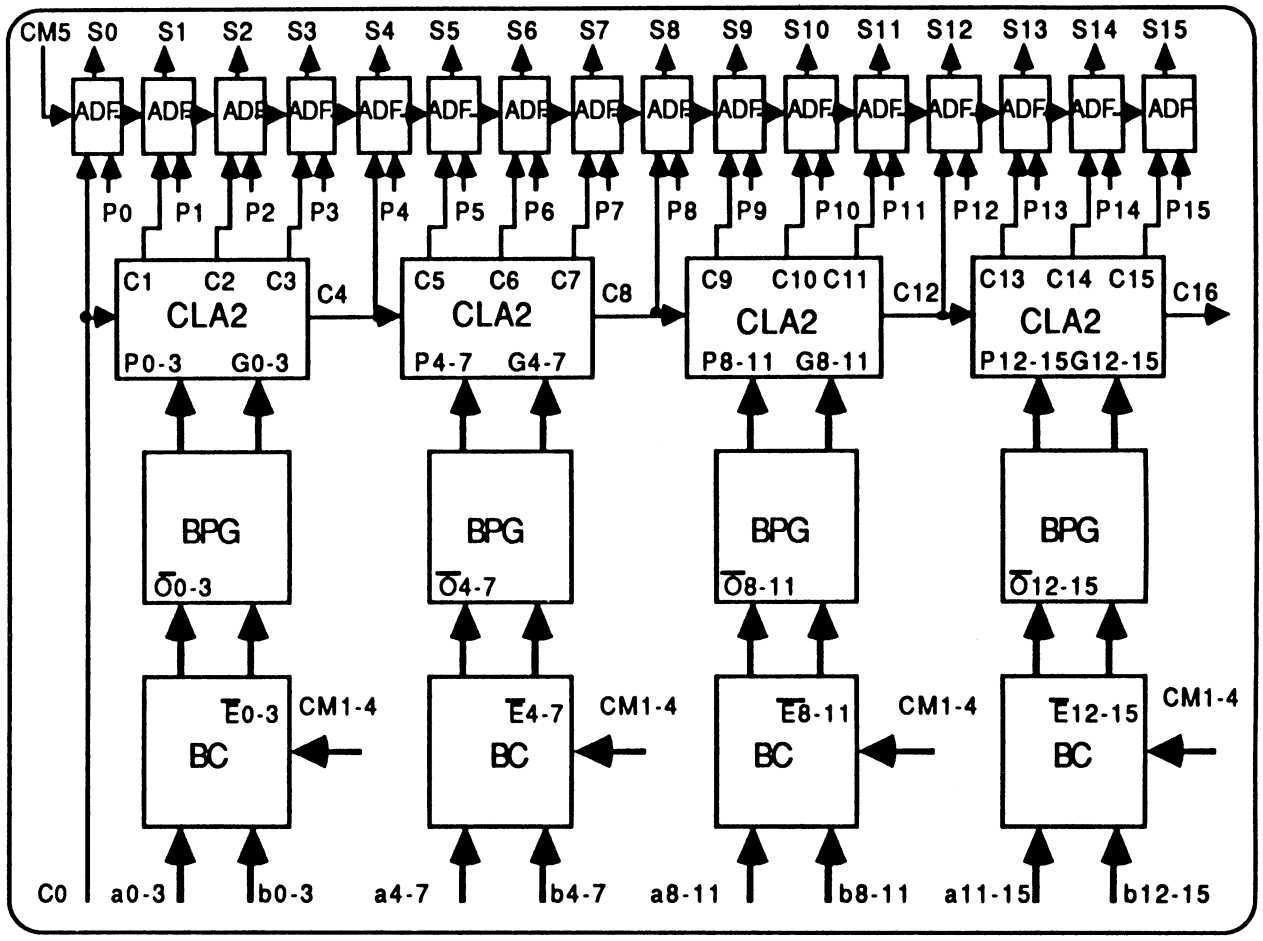


Figure 3.41 : schéma général de l'UAL de type "Carry Look Ahead Within Groups, Ripple Between Groups Adder".

4.4.2. Complexité

Nous avons :

- 576 transistors et 1152 transistors équivalents pour les blocs BC,
- 160 transistors et 304 transistors équivalents pour les blocs BPG,
- 384 transistors et 1016 transistors équivalents pour les blocs CLA2,
- 160 transistors et 240 transistors équivalents pour les additionneurs

ADF.

soit au total 1280 transistors et 2712 transistors équivalents qui donnent une augmentation de surface de 25,5 % par rapport à l'UAL de référence.

4.4.3. Vitesse

La vitesse de cette UAL est donnée pour le temps de traversée du plus long chemin (ai --> S15), soit :

$$\tau = \tau_{BC} + \tau_{BPG} + 4\,\tau_{CLA2} + \tau_{ADF}$$

$$\tau = 10 + 5 + 4 \times 6 + 9 = 48 \text{ ns}$$

Ceci correspond à une fréquence de calcul de 20,8 MHZ. Cette UAL est donc 3,2 fois plus rapide que celle de référence.

4.4.4. Testabilité

Dérivée du "Full Carry Look Ahead", cette UAL n'est pas facilement testable.

4.4.5. Reconfigurabilité

Tout comme précédemment, pour reconfigurer cette UAL, il faut ajouter un groupe supplémentaire et la logique nécessaire pour pouvoir remplacer n'importe quel groupe défectueux. Ceci implique un rajout supérieur à 30 % de la surface initiale. Donc, elle n'est pas reconfigurable compte tenu de nos critères.

4.4.6. Conclusion

La méthode "Carry Look Ahead Within Groups, Ripple Between Groups" est plus rapide que celle de l'UAL initiale (20,8 MHZ). Elle nécessite par contre une surface supplémentaire de 25,5% par rapport à l'UAL initiale.

Cette UAL n'est pas facilement testable et est surtout gourmande en surface nécessaire pour la reconfiguration.

4.5. UAL de type "Brent-Kung"

En 1982, dans [BRE82], R.P. BRENT et H.T. KUNG ont présenté un type d'additionneur utilisant les mêmes principes que les CLA (Carry Look Ahead), mais beaucoup plus réaliste pour une taille raisonnable des opérandes.

4.5.1. Principe

Le principe de cet additionneur consiste à calculer aussi les retenues Ci+1 en fonction des couples (Gi, Pi), mais au lieu de le faire en temps constant (cas des additionneurs à évaluation anticipée de la retenue), le calcul est fait en un temps proportionnel au logarithme binaire de la taille des opérandes.

Soit A et B les deux opérandes définis comme suit :

A = an - 1 an - 2 --- ai --- a1 a0

B = bn - 1 bn - 2 --- bi --- b1 b0

et

Gi = ai bi

Pi = ai $\oplus$ bi

Ci+1 = Gi + Pi Gi-1 + Pi Pi-1 Gi-2 + ... + Pi Pi-2 .. Pi G0 + Pi Pi-1...

P1P0 C0

Un opérateur O est défini par :

(X1, Y1) O (X2 , Y2) = (X1 + X2 Y1, Y1 Y2)

Il a été montré dans [BRE82] que cet opérateur est associatif (Propriété N°1).

Deux quantités γi et πi sont aussi définies par :

$$(\gamma_i, \pi_i) = \begin{cases} (G0, P0) \text{ si } i = 0 \\ (Gi, Pi)\ O\ (\gamma_{i-1}, \pi_{i-1}) \text{ si } 1 \leq i \leq n \end{cases}$$

avec la propriété N° 2 :

γi = Ci + 1 et πi = Pi Pi -1 .... Po

Suite à la démonstration des deux propriétés précédentes, un schéma de calcul des retenues (BBK) a été proposé. Nous allons emprunter le même schéma afin de donner une structure de l'UAL 16 bits à base de cet additionneur (Figure 3.44).

Signalons que le bloc BBK calcule les couples (γi, πi) en fonction des (Gi,Pi) avec C0 = 0. Ceci implique que :

γi = Ci +1 = Gi + Pi Gi-1 + Pi Pi-1 G i- 2 + ... + Pi Pi-1 ..... P2 P1 G0

πi = Pi Pi-1 ..... P2 P1 P0

Le schéma logique de l'opérateur O est également donné en figure 3.42.

Des blocs supplémentaires (CR) sont destinés à compléter le calcul des retenues pour une retenue entrante différente de 0. Ceci n'est pas prévu dans le schéma Brent-Kung. Chacun de ces blocs calcule l'expression :

Ci+1 = πi . C0 + γi.

La réalisation de cette expression pourrait être le circuit donné par la figure 3.43.

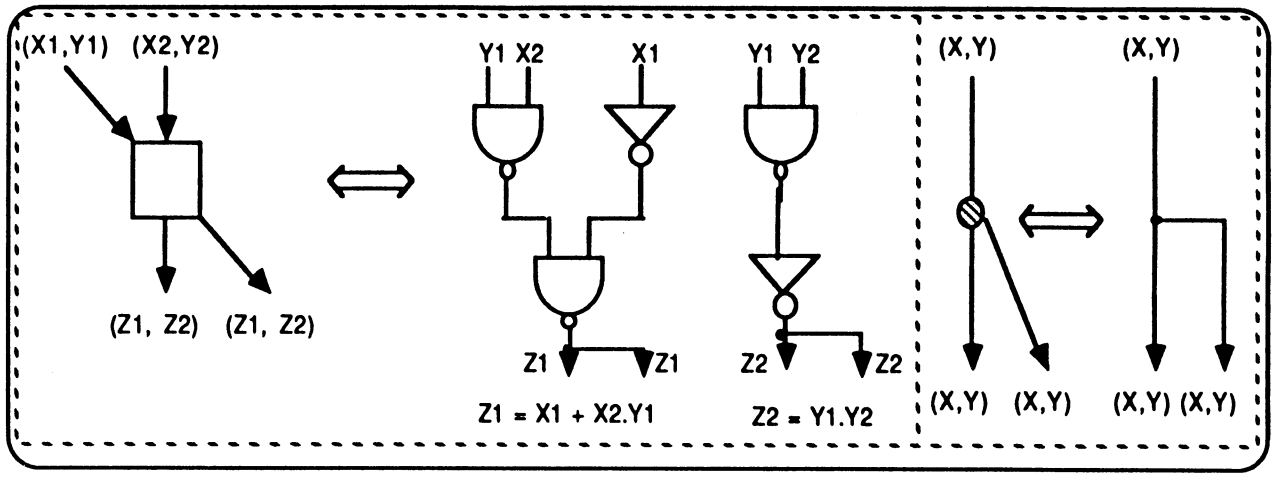


Figure 3.42 : schéma logique de l'opérateur O.

Les blocs restant sont déjà connus, cela permet d'évaluer la complexité et la vitesse de cette UAL.

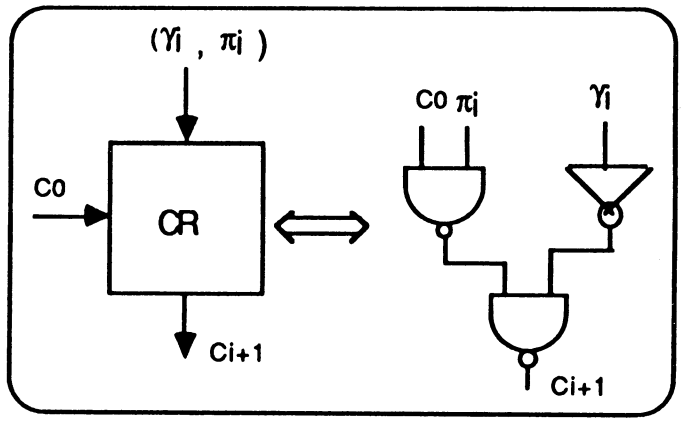


Figure 3.43 : circuit logique du bloc CR.

4.5.2. Complexité

- blocs BC : 576 transistors et 1152 transistors équivalents,
- blocs BPG : 160 transistors et 304 transistors équivalents,
- bloc BBK : 416 transistors et 780 transistors équivalents,
- blocs CR : 160 transistors et 304 transistors équivalents,
- blocs ADF: 160 transistors et 240 transistors équivalents.

Pour l'UAL entière on a 1472 transistors et 2780 transistors équivalents, soit 28,7% de plus que la surface de l'UAL de référence.

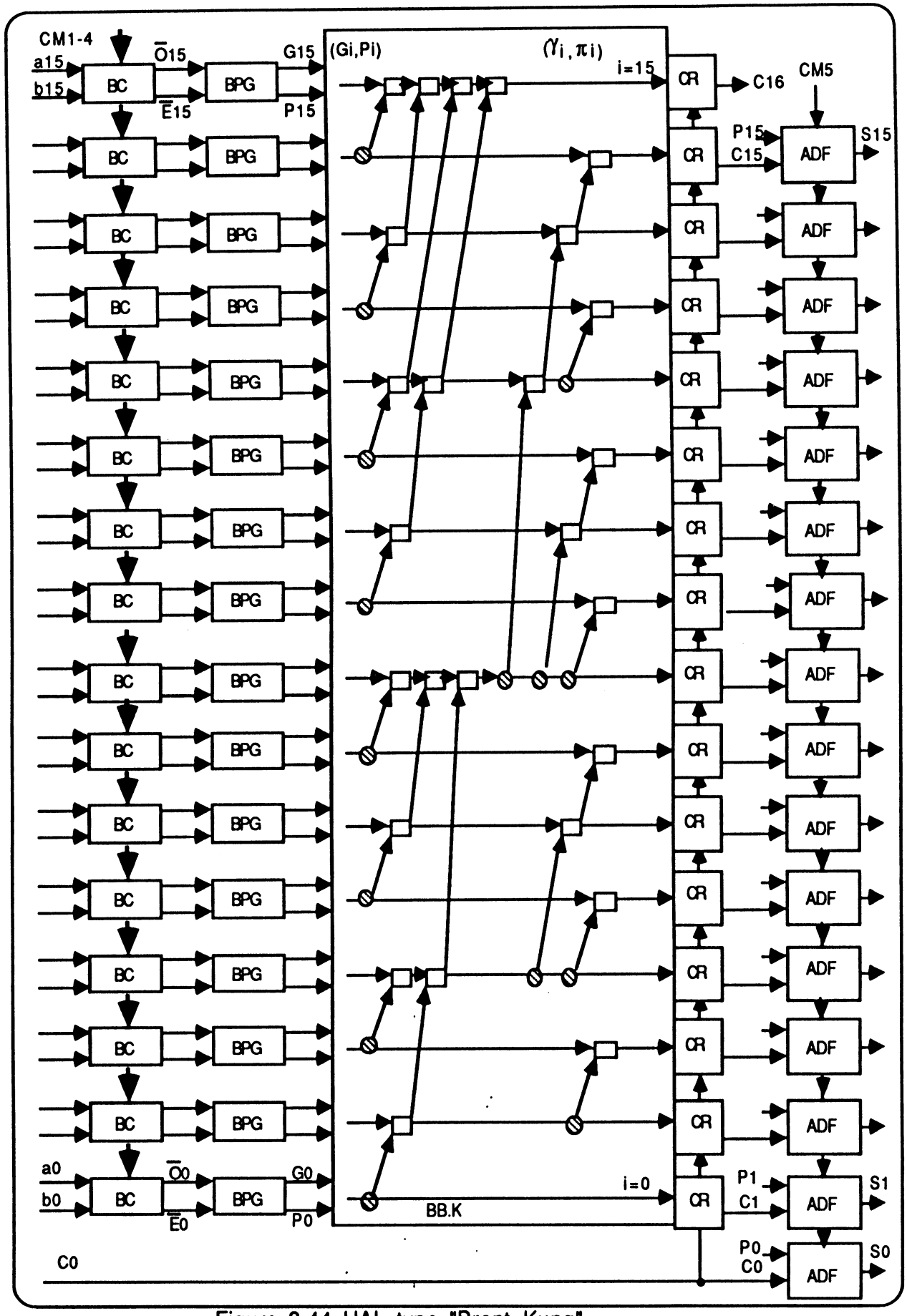


Figure 3.44 UAL type "Brent_Kung"

4.5.3. Vitesse

La vitesse de l'UAL de type "Brent-Kung" peut être estimée en calculant le temps de propagation de chaque bloc suivant la formule donnée ci-dessous:

$$\tau_{UAL} = \tau_{BC} + \tau_{BPG} + \tau_{BBK} + \tau_{CR} + \tau_{ADF}$$

$$\tau_{UAL}(ns) = 10 + 5 + 7 \times 5 + 5 + 9 = 64 \text{ ns.}$$

La fréquence déduite de la valeur précédente est d'environ 15,6 MHZ, c'est-à-dire 2,4 fois plus grande que celle de l'UAL de référence.

4.5.4. Testabilité

Les remarques faites sur le "Full Carry Look Ahead Adder" sont encore valables ici : la structure n'est plus en tranche et une panne créée par le circuit ne se propage pas obligatoirement vers une sortie observable.

De plus, une séquence de test complète ne permet pas nécessairement de localiser la position de la panne.

4.5.5. Reconfigurabilité

La reconfiguration est ici encore plus complexe; la seule possible consiste à doubler le bloc UAL. Elle n'est donc pas envisageable avec nos critères, car cela nécessiterait plus de 100 % de surface supplémentaire.

4.5.6. Conclusion

La méthode "Brent-Kung" est plus rapide que celle de l'UAL initiale (15,6

MHZ) et nécessite un tiers de surface supplémentaire, mais surtout, elle ne se prête pas à une reconfiguration acceptable et n'est pas facilement testable.

5.Comparaison des différentes méthodes

Le tableau 3.7 résume les différentes évaluations déjà effectuées. Pour chaque méthode sont donnés :

- le nombre de transistors réels (NB. TR. R),
- le nombre de transistors équivalents (NB. TR. E) et l'accroissement de surface par rapport à l'UAL initiale,
- la fréquence de calcul estimée d'après le délai maximum du calcul,
- la testabilité,
- la reconfigurabilité,
- des remarques éventielles.

TYPE D'UAL	NB. TR. R	NB. TR. E	FREQENCE (MHZ)	TESTABILITE	RECONFIGURABILITE	REMARQUES
UAL INITIALE	1184	2160	6,5	TYPE ILA 140 VE. TES.	RECONFIGURABLE SUR.SUP<25%	
DOMINO	704	1840, -15%	13	TYPE ILA 140 VE. TES.	RECONFIGURABLE SUR.SUP<25%	+HORLOGE
MANCHESTER	1040	1968, -9%	19	TYPE ILA 140 VE. TES.	RECONFIGURABLE SUR.SUP<25%	+HORLOGE
CARRY SELECT	1894	3398, +57%	18	POSSIBLE(*) 2X140 VE.TES.	SUR.SUP>30%	+COMMANDE +TEST C_i
CARRY SKIP	1312	2504, +16%	17	POSSIBLE(**) 140 VE. TES.	SUR.SUP<25% SUR.NON REC~40%.	+COMMANDE +32VE. TES.
FULL CLA	1368	2890, +34%	23,8	PAS DE TYPE ILA	SUR.SUP>30%	
RIPPLE WIT. GRO. CLA BET. GRO.	1264	2582, +20%	19,6	PAS DE TYPE ILA	SUR.SUP>30%	
CLA WIT. GRO. RIPPLE BET. GRO	1280	2712, +25%	20,8	PAS DE TYPE ILA	SUR.SUP>30%	
BRENT_KUNG	1472	2780, +29%	15,6	PAS DE TYPE ILA	SUR.SUP>30%	

Tableau 3.7 : résumé des différentes évaluations des UAL

D'après ce tableau, les méthodes les plus rapides sont le "Carry Select" et le "Carry Look Ahead" avec ses variantes. Mais ces deux types de méthodes nécessitent des surfaces supplémentaires par rapport à l'UAL initiale les rendant inintéressantes pour l'application recherchée : 60% pour le "Carry Select", entre 20 % et 34 % pour le "Carry Look Ahead". De plus, ces circuits ne sont pas facilement testables et sont surtout gourmands en surface nécessaire pour la reconfiguration.

La méthode de "BRENT-KUNG" est moins rapide que les précédentes et nécessite un tiers de surface supplémentaire, mais surtout, elle ne se prête pas à une reconfiguration acceptable. Cette méthode ne peut donc pas être retenue.

La méthode "DOMINO" appliquée à l'UAL initiale, avec une diminution de surface de 15 % a un gain de vitesse moins élevé que les autres méthodes. Elle est par contre aisément testable et reconfigurable tout comme l'UAL initiale.

La méthode "MANCHESTER", bien que nécessitant une horloge de précharge, offre une augmentation de vitesse satisfaisante pour une légère diminution de surface (-10 %). Elle est également facilement testable et reconfigurable.

La méthode "Carry-Skip" pour une légère augmentation de surface (+16%), offre une bonne rapidité, mais par contre est moins aisément testable que Manchester et la surface non reconfigurée est assez élevée.

Ces deux dernières méthodes se révèlent être les plus intéressantes pour notre application.

L'implantation et la simulation fine de ces deux méthodes nous permettront de préciser les caractéristiques évaluées du point de vue vitesse et surface. Les séquences de test nécessaires sont également déterminées.

6. Réalisation

6.1. UAL initiale (CMOS statique) implantée

L'UAL initiale a été implantée au sein du microprocesseur HYETI sur une surface de 0,694 mm^2. Un prototype de 5 tranches a aussi été implanté. Sa fabrication par la société SGS Thomson permet de valider la conception de cette UAL et d'examiner les performances attendues. Une présentation du dessin au niveau masque de l'UAL 5 bits est donnée en annexe 3.3.

6.2. UAL "Manchester" implantée

La surface de l'UAL Manchester est comparable à celle de l'UAL initiale (0,687 mm^2). Ceci démontre que l'estimation de surface effectuée est grossière. Par ailleurs, l'évaluation des performances est également peu précise. La simulation électrique SPICE après extraction du schéma implanté montre que le gain en vitesse n'est que de 50%. Ceci s'explique par l'influence des paramètres qui n'avaient pu être mis en évidence lors de l'estimation initiale (capacités parasites, ...). Le schéma d'implantation de l'UAL Manchester est donné en annexe 3.4.

6.3. UAL "Carry-Skip" implantée

L'UAL "Carry-Skip" a une surface de 0,929 mm^2, ce qui correspond à une augmentation de surface de l'ordre de 34% (16% estimés). Notons qu'avec cette méthode 40% de la surface n'est pas reconfigurée au lieu de 20% pour l'UAL initiale. C'est l'un des principaux inconvénients de cette UAL. La fréquence de calcul est de 20 MHZ. Le schéma d'implantation de cette UAL est donné en annexe 3.5.

7.Conclusion

L'implantation de deux types d'UAL (Manchester et Carry-Skip) a permis de mesurer la valeur des estimations faites précédemment et de préciser les caractéristiques réelles de ces deux UAL. Les deux UAL choisies pour l'implantation sont relativement rapides (par rapport à l'UAL initiale aussi implantée), facilement testables et reconfigurables. L'UAL "Carry-Skip" présente l'avantage de tripler la fréquence d'utilisation (20 MHZ), mais l'inconvénient d'avoir une grande taille (0,929 mm^2) et une surface non reconfigurable de 40%. L'UAL de Manchester présente l'avantage d'avoir une surface comparable (0,687 mm^2) à celle de l'UAL initiale et d'utiliser les mêmes stratégies de test et de reconfiguration, mais par contre sa fréquence d'utilisation est de 15 MHZ.

Les trois types d'UAL sont réalisés en technologie HCMOS3C 1,2 µm.

Chapitre 4 :
circuit d'analyse de signature

Dans ce chapitre nous rappellerons tout d'abord les principes de la compaction d'information très largement utilisée pour l'observation des résultats de test [ROB87].

La technique choisie consiste en une division polynômiale de l'information à observer (la séquence de sortie de test) et à un stockage du reste appelé signature.

Dans une deuxième partie, nous présenterons deux circuits réalisés à cette fin sur silicium. Le premier est un circuit "universel" destiné à réaliser cette compaction avec un polynôme programmable. C'est un circuit sur 8 bits, cascadable et intégrable comme boîtier autonome sur une carte.

Cette réalisation a été faite dans le cadre du service national "CMP" et a conduit à un circuit opérationnel en technologie CMOS 2 µm. Le deuxième circuit est un bloc intégré dans le microprocesseur HYETI et réalisant comme indiqué au chapitre 1, la compaction de n'importe quelle donnée circulant sur le bus à des fins, en particulier, de test en ligne.

Dans une troisième partie nous nous intéresserons au test hors ligne de ces circuits eux-mêmes. Le principe même de ces circuits, consistant à compacter leur information, peut introduire des difficultés de test. Il convient donc d'étudier soigneusement leur testabilité. La méthode proposée ici tient compte de la structure itérative de tous les circuits de traitement de données. Elle cherche dans un premier temps à atteindre toutes les valeurs d'entrée de la cellule de base (test exhaustif de la cellule de base) et toutes les transitions des points mémoire. Il s'agit en fait d'étendre les propriétés de C-testabilité des circuits itératifs logiques aux circuits comprenant des cellules de base avec bascule D. La C-testabilité de deux types de circuits de signature a été examinée :

- circuits de signature à entrée série,

- circuits de signature à entrées parallèles.

Cette méthode est d'abord appliquée au circuit de signature universel isolé pour lequel on montre qu'un ensemble de 2^4 vecteurs de test (4 est le nombre d'entrées de la cellule de base) réalise un test fonctionnel quasi-exhaustif. Ceci permet de contourner l'obstacle des hypothèses de panne toujours sujettes à caution dans les technologies actuelles.

Pour le bloc d'analyse de signature réalisé dans le microprocesseur, on montre que des problèmes de contrôlabilité existent et que des modifications du microprocesseur sont nécessaires pour assurer ce même test fonctionnel exhaustif. Les modifications seront explicitées.

On montre ensuite la possibilité d'éviter ces modifications en ajoutant quelques vecteurs de test (environ 25%).

1. Principe de la compaction d'information [DAV80] [HAS82b] [CAR86] [JAY88]

Pour le test en ligne comme pour le test hors ligne, la quantité d'informations de test à stocker pose de sérieux problèmes. Des techniques de compaction permettent de résoudre ces problèmes et de ne comparer qu'une donnée significative appelée signature. Cette technique déjà mentionnée au chapitre 1 pour le test en ligne sera rappelée ici.

Un test hors ligne est effectué en deux étapes :

- application des vecteurs de test sur les entrées du circuit sous test, puis sur un circuit de référence.
- comparaison des valeurs de sortie.

Le principe de test avec compaction d'information est illustré par la figure 4.1 [ROB87].

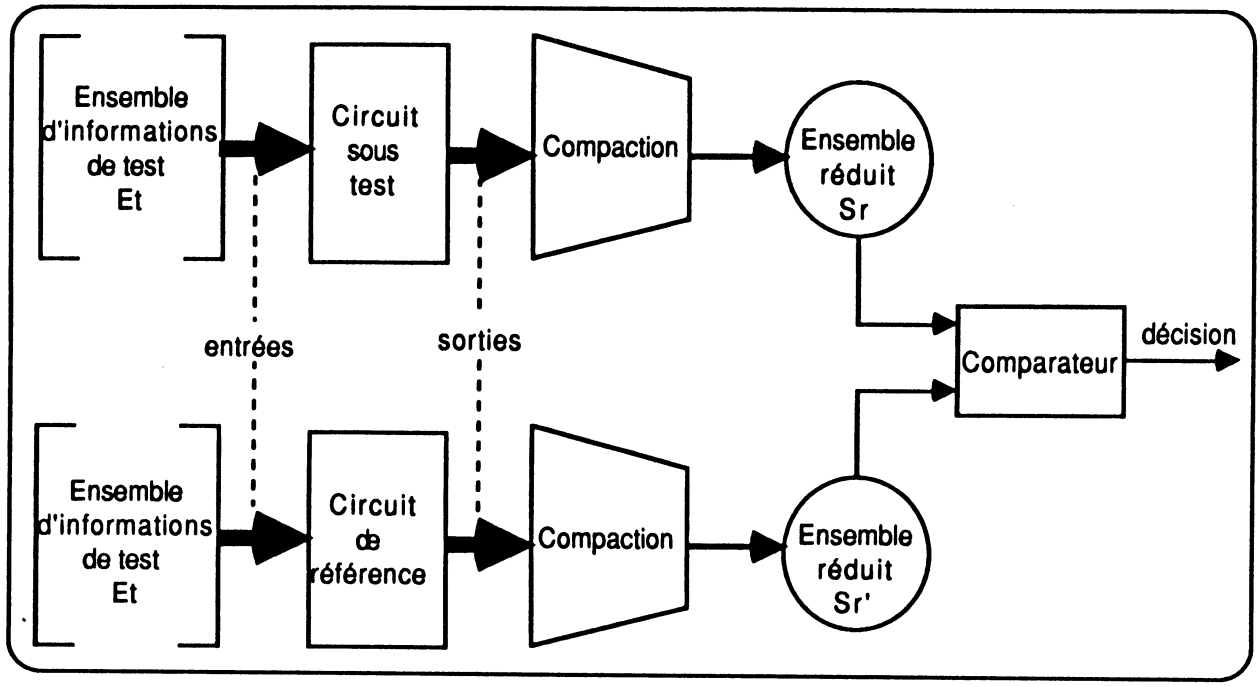


Figure 4.1 : stratégie de test

Et est l'ensemble des informations appliquées sur les entrées du circuit sous test et sur les entrées du circuit de référence.

Sr : est la réponse réduite du circuit sous test lors de l'application de Et.

Sr' : est la réponse réduite du circuit de référence lors de l'application de Et.

L'ensemble réduit (Sr ou Sr') est un résumé représentatif des événements qui se sont manifestés sur les sorties des deux circuits.

Le contenu de Sr est la signature des informations résultant de la séquence de test (Et).

Il existe plusieurs méthodes de compaction. On peut citer parmi elles à titre d'exemple :

- compte des transitions **0→1** et **1→ 0** en sortie du circuit [HAY76] [VEN80].
- méthodes vérifiant le syndrome (vérification du nombre des 1) [SAV80] [SAV81] [GUP88] [ROB88].

- méthodes vérifiant les coefficients de Walsh [SUS81] [MUZ82] [SUS83].
- méthodes utilisant la division polynômiale [SMI80] [SHR82] [WAN86c] [WAN86d] [SMI87] [KAR88] [JAY88].

La méthode choisie appartient à la dernière catégorie. Il faut signaler que le taux de couverture des pannes dépend :

* du taux de couverture de la séquence initiale de test.
* du taux de pannes masquées par l'opération de signature.

Nous nous intéressons au deuxième point en cherchant à diminuer le taux de pannes masquées. Il a été montré dans [JAY86] que l'amélioration de l'efficacité de la compaction par division polynômiale repose sur trois points :

1°) augmentation de la taille de la signature, ce qui entraîne une augmentation du nombre de bascules, d'où une augmentation de la surface de silicium.

2°) stockage de plusieurs signatures pour la même séquence de test.

3°) compaction d'une même séquence de test à l'aide de différents polynômes diviseurs.

La solution retenue est la dernière, car elle n'exige pas d'augmentation de surface, et elle améliore l'efficacité de la compaction. Considérons l'exemple suivant [SOU86] :

supposons que l'exécution correcte d'une suite d'instructions se traduit par l'apparition au sein d'un registre interne de la suite de valeurs **3,8,1,5,7,6,3,1.**

La vérification sans compaction du bon fonctionnement implique la comparaison de la suite de valeurs obtenues par le circuit sous test avec des valeurs de référence au fur et à mesure de leurs apparitions.

La méthode de compaction consiste à diviser le nombre constitué par la suite de valeurs par un polynôme diviseur : le reste (signature) doit être

comparé avec celui obtenu par un circuit référence.

Soit un diviseur décimal égal à **100**, nous avons :

$$\frac{\text{suite de référence}}{100} = \text{quotient} + \frac{\text{reste}}{100}$$

$$\frac{38157631}{100} = 381576 + \frac{31}{100} \rightarrow \text{signature}$$

Dans cet exemple, si l'un des **6** premiers chiffres est perturbé, il n'y aura pas de détection d'erreur, car la signature obtenue avec la séquence de référence sera la même que celle obtenue avec la séquence fausse. Si maintenant le diviseur choisi est **173**, et si l'exécution de la suite d'instructions donne **3, 8, 2, 5, 7, 6, 3, 1**, on aura pour la signature de référence :

$$\frac{38157631}{173} = 220564 + \frac{59}{173} \rightarrow \text{signature de référence}$$

Et pour la séquence fausse :

$$\frac{38257631}{173} = 221142 + \frac{65}{173} \rightarrow \text{signature de la séquence fausse}$$

Cette différence mène à la détection d'une erreur. On constate que la possibilité de pouvoir changer la valeur du diviseur est primordiale.

2. Principe de réalisation d'un circuit universel de compaction d'informations par division polynômiale

2.1. Principe mathématique et schéma général [HAS82] [SIE82] [SHR82] [CAR86] [WAN86a] [SOU86] [WAN86b] [JAY88]

Le dispositif de signature est un circuit diviseur modulo 2 qui sert à compacter un flot d'informations en série ou en parallèle [SOU86] [HAS82]. Il est caractérisé par un polynôme diviseur appelé polynôme caractéristique.

Soit **P(x)** un polynôme de degré **k** qui peut être écrit sous la forme :

$$(1)\quad P(x) = p_k x^k + p_{k-1} x^{k-1} + \ldots\ldots + p_1 x + p_0$$

Nous voulons réaliser un circuit qui associe à un signal d'entrée **E(x)**, un signal de sortie **S(x)** tel que :

$$(2)\quad S(x) = \frac{E(x)}{P(x)} \Rightarrow \text{CIRCUIT DIVISEUR}$$

E(X) est également écrit sous forme polynômiale.

Nous devons avoir :

$$(3)\quad S(x)\ P(x) = E(x)$$

$$(4)\quad S(x)\ [p_k x^k + p_{k-1} x^{k-1} + p_{k-2} x^{k-2} + \ldots\ldots + p_0] = E(x)$$

$$(5)\quad S(x)\ p_k x^k = E(x) - S(x)\ [p_{k-1} x^{k-1} + p_{k-2} x^{k-2} + \ldots\ldots + p_0]$$

$$(6)\quad S(x) = \frac{E(x)}{p_k x^k} - S(x)\ \left[\frac{p_{k-1}}{p_k x} + \frac{p_{k-2}}{p_k x^2} + \ldots + \frac{p_0}{p_k x^k}\right]$$

Dans cette expression, on trouve trois types d'opérations : addition,

soustraction,multiplication.

Pour un système modulo **2**, la somme et la soustraction se réalisent par une porte OU exclusif et la multiplication par une porte ET (AND).

Cette expression est réalisée par le circuit de la figure 4.2 pour un système modulo 2.

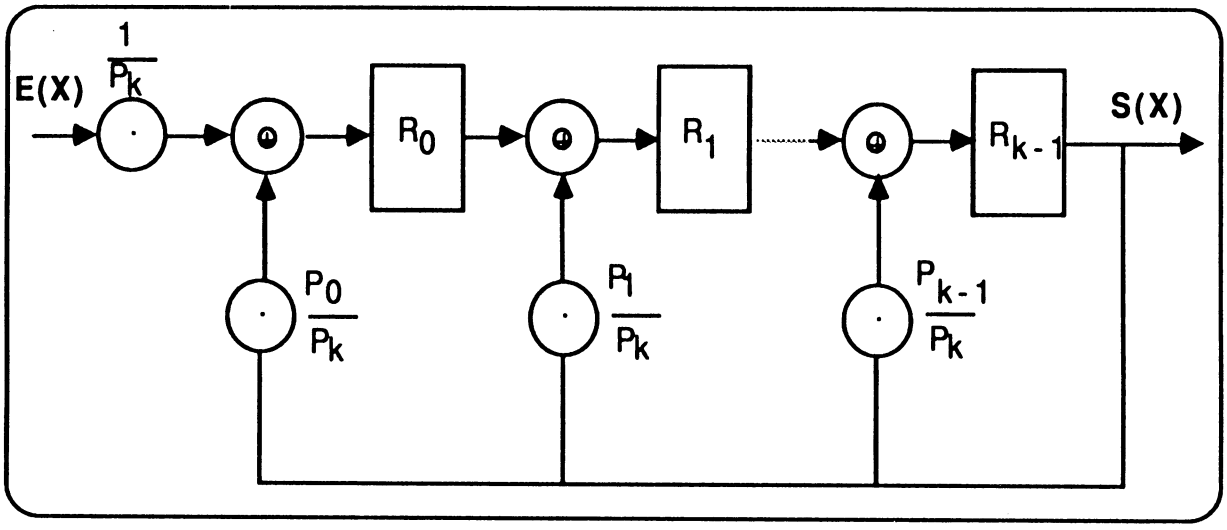


Figure 4.2 : circuit diviseur.

Remarques :

- pour un polynôme diviseur de degré K, il y a K retards (R_0 à R_{k-1}),
- P_k doit être différent de 0, ceci implique que $P_k = 1$ (système modulo 2),
- si $P_i = 1$, la sortie de R_{k-1} ou **S(X)** est reliée à l'entrée de la porte OU exclusif de R_i ($0 \leq i \leq k-1$), sinon il n'y a pas de rebouclage.

La réalisation matérielle du circuit de signature utilisera des composants élémentaires (bascules et portes OU exclusif "additionneur/ soustracteur").

Prenons un exemple qui met en évidence le fonctionnement du circuit de signature série.

Soit le polynôme diviseur :

$P(X) = X^5 + X^4 + X^2 + 1$,

et le polynôme d'entrée :

$E(X) = X^7 + X^6 + X^5 + X^4 + X^2 + 1$

Le circuit de signature série effectuant cette division polynômiale est donné en figure 4.3.

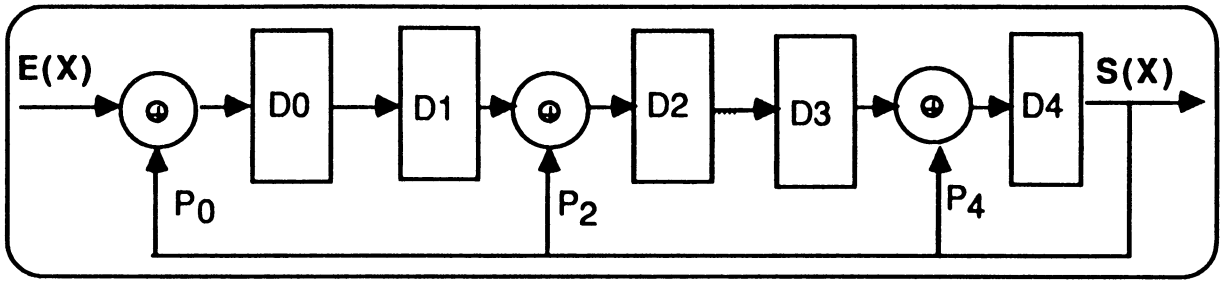


Figure 4.3 : Schéma d'un circuit de signature série

Le schéma précédent est déduit du schéma général de la figure 4.2 après avoir pris en compte les coefficients du polynôme diviseur.

La division polynômiale s'écrit :

$$\begin{array}{l|l} X^7 + X^6 + X^5 + X^4 + X^2 + 1 & X^5 + X^4 + X^2 + 1 \\ \hline -(X^7 + X^6 \quad + X^4 + X^2) & X^2 + 1 \\ \hline +X^5 \quad + 1 \\ \hline -(X^5 + X^4 + X^2 + 1) \\ \hline X^4 + X^2 \end{array}$$

Nous pouvons écrire en binaire :

E(x) = 11110101

P(x) = 110101

La division polynômiale s'écrit alors :

(DIVIDENDE) 11110101	(DIVISEUR) 110101
110101 (EXOR)	101 (QUOTIENT)
00100001 000000 (EXOR)	
100001 110101 (EXOR)	
010100 (RESTE)	

Les différentes phases exécutées par le dispositif sont :

Flot d'entrées (DIVISEUR) LSB MSB	Contenu des registres D1 D2 D3 D4 D5	Flot de sortie
1 0 1 0 1 1 1 1	0 0 0 0 0	Initialisation
1 0 1	0 1 1 1 1	Aprés 5 décalages
1 0	0 0 0 1 0	1 Aprés 6 décalages
1	0 0 0 0 1	0 1 Aprés 7 décalages
	0 0 1 0 1	1 0 1 Aprés 8 décalages

Le défaut du schéma précédent est de ne pouvoir traiter que des informations en série.

On peut montrer qu'un circuit de signature à entrées parallèles (cf. figure 4.4) est équivalent au circuit à entrée série, à l'entrée duquel nous appliquerons :

$$E(X) = E_0(X) + X\,E_1(X) + X^2\,E_2(X) + \ldots\ldots + X^{k-1}\,E_{k-1}(X),$$

avec $E_0(X)$, $E_1(X)$, $E_2(X)$, ..., $E_{k-1}(X)$ polynômes d'entrées au circuit à entrées parallèles.

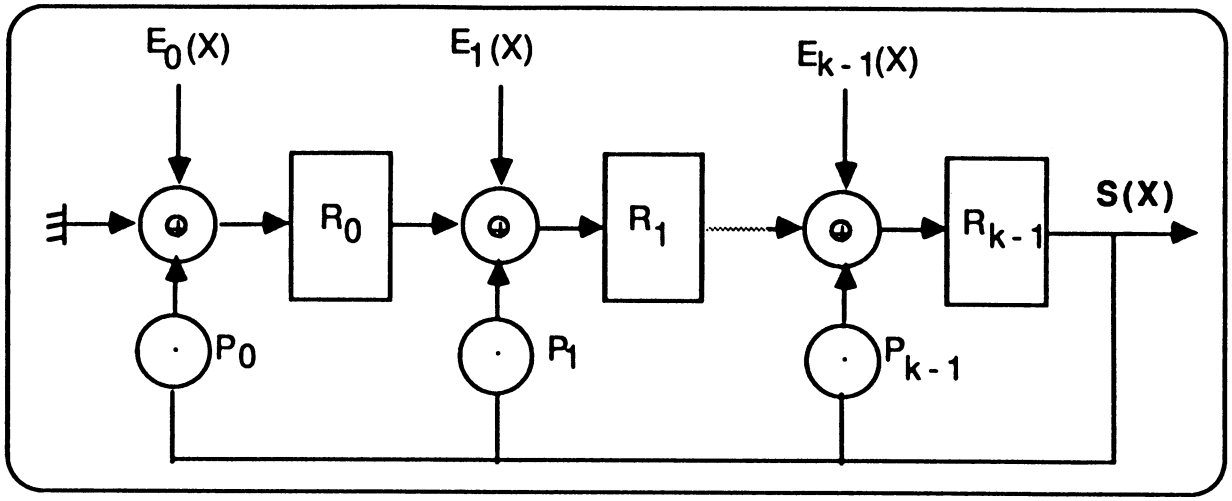


Figure 4.4 : Structure parallèle générale du circuit de signature.

Soit le polynôme diviseur :

$P(X) = X^3 + X + 1$

et les polynômes d'entrées parallèles :

$E_0(X) = X + 1 = 0011$

$E_1(X) = X^3 + X^2 + 1 = 1101$

$E_2(X) = X^2 = 0100$

Le polynôme d'entrée série est :

$E(X) = X + 1 + X(X^3 + X^2 + 1) + X^2(X^2)$

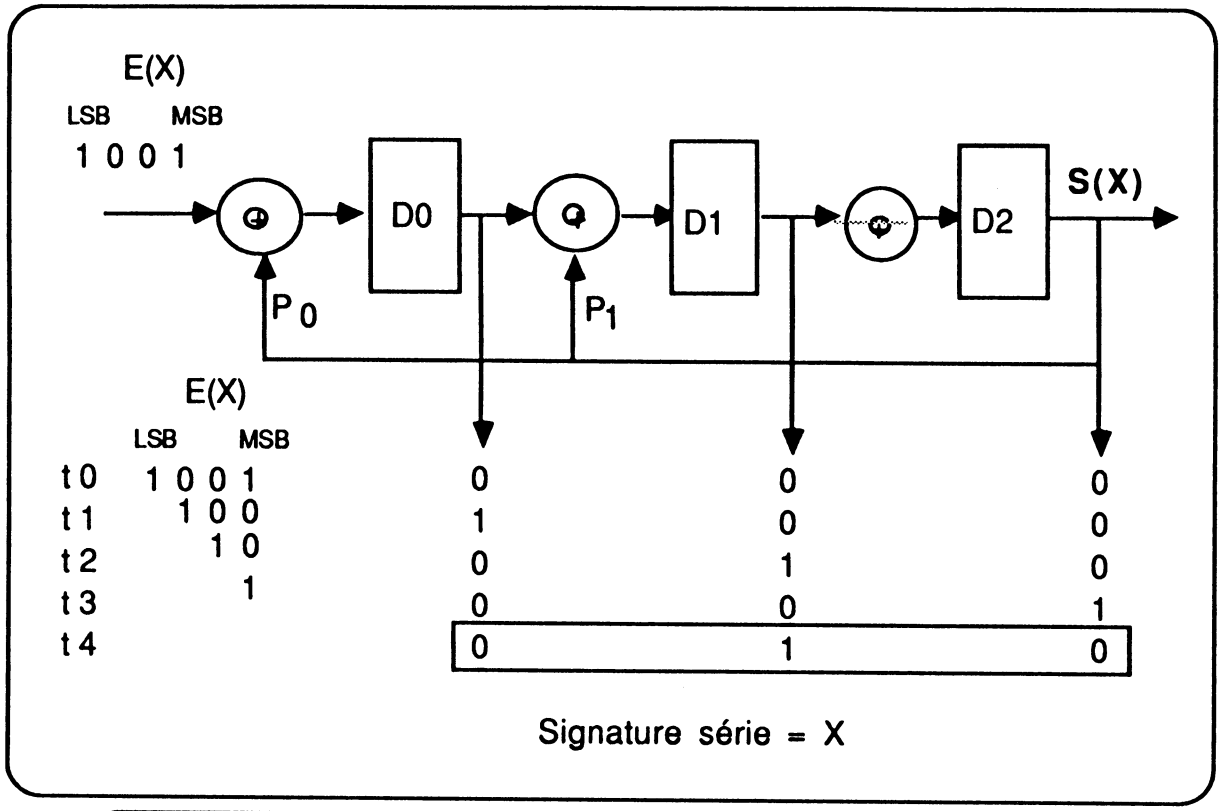
$= X + 1 + X^4 + X^3 + X + X^4$

$= X^3 + 1 = 1001$

La division polynômiale sera :

$X^3 + 1$	$X^3 + X + 1$
$X^3 + X + 1$	1 (QUOTIENT)
X (RESTE)	

Ceci est représenté sur la figure 4.5



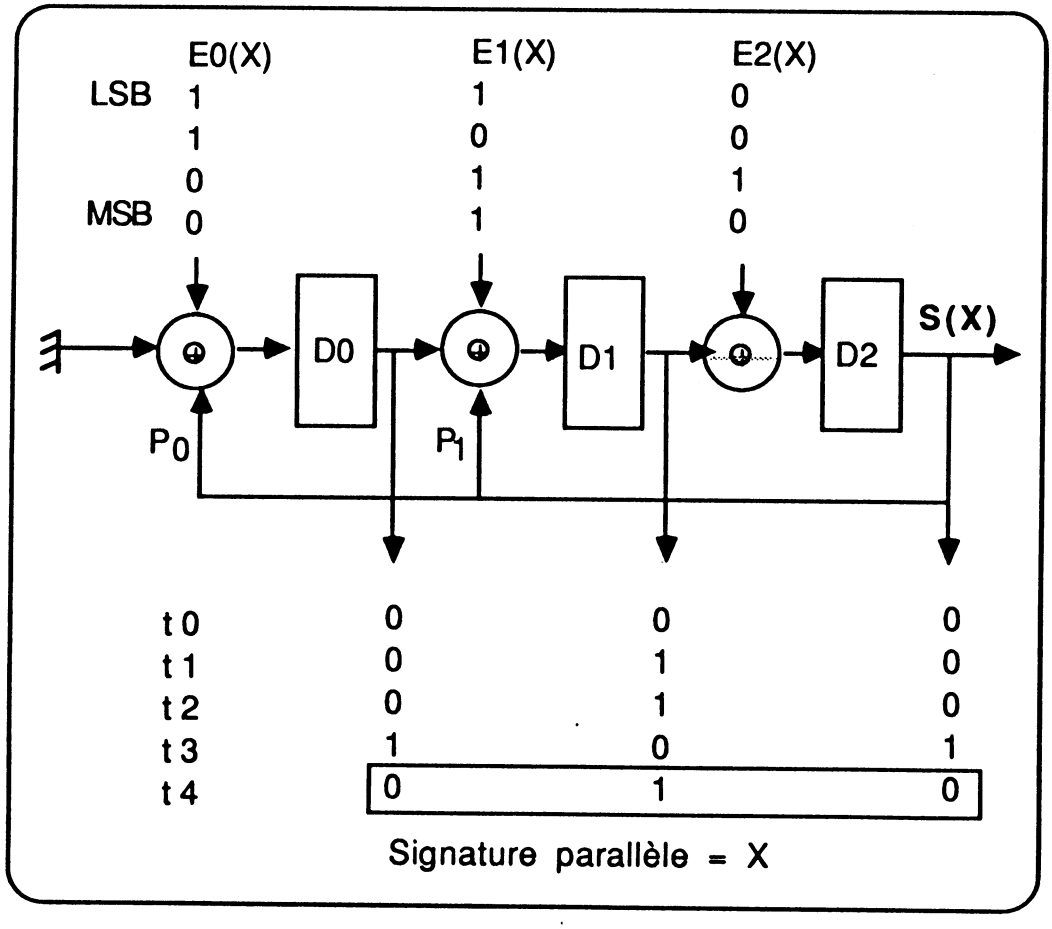


Figure 4.5 : Equivalence entre circuits de signature série et parallèle.

2.2. Réalisation du circuit de signature parallèle, programmable et cascadable sur 8 bits

Nous avons réalisé un dispositif de compaction d'informations fondé sur la division polynômiale par un polynôme programmable, sur 8 bits et cascadable.

Partant de l'étude théorique (paragraphe 2.1), une architecture est proposée, puis dans un premier temps une simulation logique est effectuée sur une cellule de base de 1 bit et sur le circuit de 8 bits afin de valider son fonctionnement.

Dans un deuxième temps, le schéma logique est traduit en schéma électrique pour lequel on réalise de nombreuses simulations en ajustant des paramètres critiques tels que : dimensions des transistors, temps de propagation, ...

Ces deux étapes nécessaires nous amènent au dessin des masques du circuit au micron, en technologie CMOS 2 métaus, en vue d'une réalisation dans le cadre du CMP.

2.2.1. Choix de l'architecture du circuit

Ce circuit doit travailler sur des mots de 8 bits en parallèle. Pour avoir la possibilité de travailler sur des mots de longueurs multiples de 8 bits, il doit de plus être cascadable. Afin d'être en mesure de détecter le maximum d'erreurs, il doit posséder un polynôme diviseur programmable.

La figure 4.6 donne le diagramme architectural du circuit de signature programmable cascadable sur 8 bits.

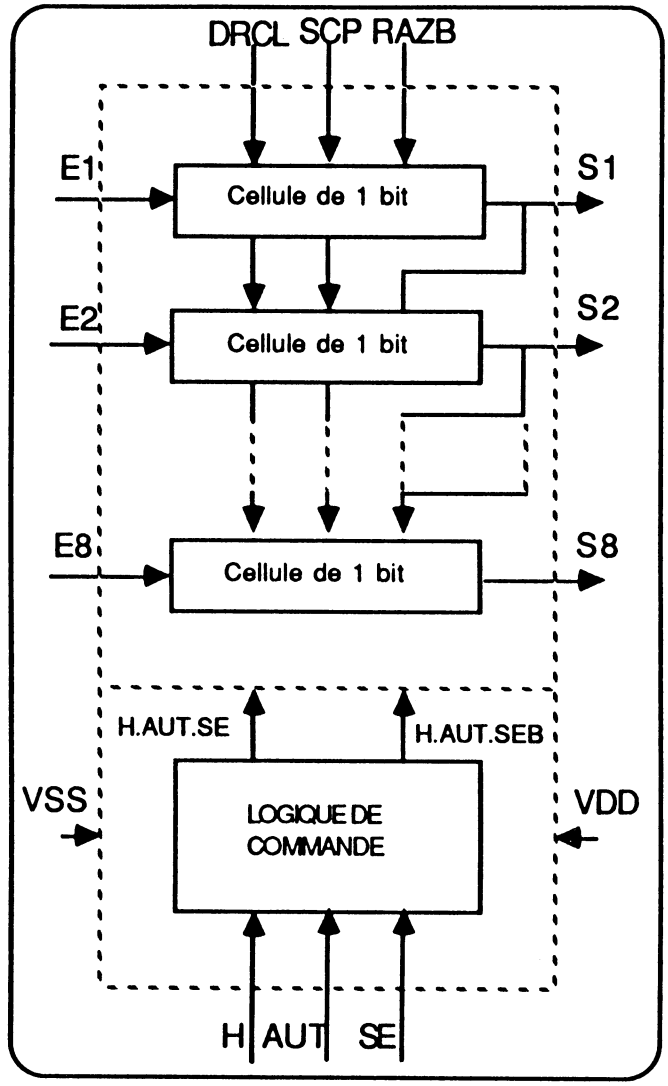


Figure 4.6 : circuit de signature parallèle sur 8 bits.

Le circuit possède les entrées-sorties suivantes :

- E1, ... E8 : entrées des données à compacter ou entrées du polynôme diviseur suivant les valeurs de AUT et SE,

- S1, ... S8 : sorties du circuit donnant la signature (reste de la division polynômiale),

- AUT : autorisation de signature, valide à l'état haut. A l'état bas, la programmation du diviseur n'est pas valide, la signature n'est pas autorisée,

- SE : sélection. Si SE = 1 les entrées E1, ... E8 constituent un mot binaire sur 8 bits représentant le diviseur, sinon ces entrées sont des

données à signer,

- RAZB : remise à zéro active à l'état bas (signal d'initialisation des bascules de signature),
- SCP : entrée du report de sortie de la cellule précédente,
- DRCL : entrée du report de la dernière cellule,
- V_{dd} , V_{ss} : alimentation (+ 5V) et masse.

2.2.2. Circuit logique

Tout d'abord, le schéma logique de base du circuit de signature 1 bit est illustré dans la figure 4.7.

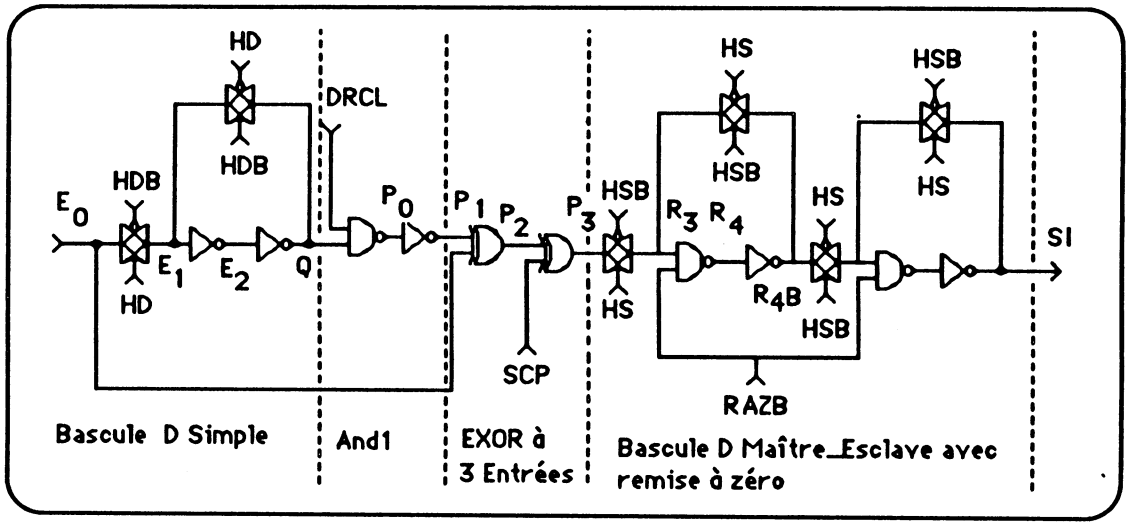


Figure 4.7 : schéma logique de 1 bit du circuit de signature.

Ce schéma élémentaire contient les éléments de base suivants :

- une bascule D simple dite "polynôme diviseur" qui stocke la valeur P_I du polynôme si son horloge est validée. Ceci impose :

$$HD = \overline{HDB} = H.AUT.SE$$

- une porte ET qui laisse passer la dernière sortie vers une entrée d'une porte OU exclusif (EXOR à 3 entrées) et qui reboucle la dernière sortie, si la valeur correspondante du polynôme vaut 1;

- une porte OU exclusif à trois entrées attaquée par la sortie de la dernière cellule **(DRCL)** (si la sortie du polynôme diviseur vaut 1, sinon cette entrée vaut 0), par une entrée à signer **(E_I)** et par la sortie de la cellule précédente **(SCP)**;

- une bascule D Maître-Esclave qui représente un élément de retard attaqué par la sortie **(P_3)** de la porte OU exclusif, par une entrée de commande de remise à zéro **(RAZB)** (active à l'état bas) et par des signaux d'horloge :

$$HS = H \cdot AUT \cdot \overline{SE} = \overline{HSB}$$

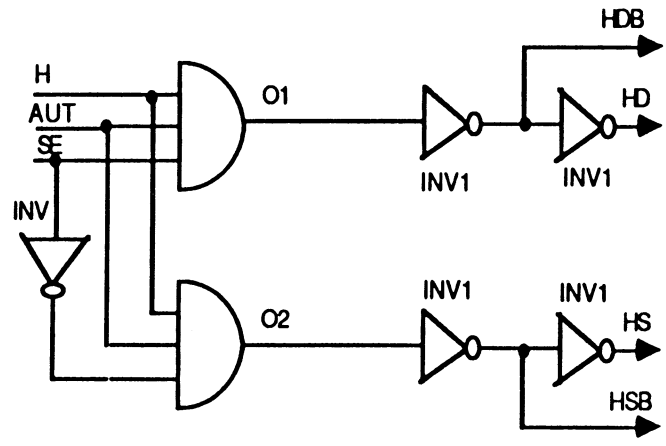
Sa sortie (SI) représente un bit du résultat de signature.

En répétant huit fois la cellule de base de signature 1 bit, on construit le circuit de signature programmable cascadable sur 8 bits (Figure 4.6).

La table de vérité (figure 4.8a) permet de calculer l'équation booléenne du circuit de commande et de déduire le schéma logique correspondant (figure 4.8b).

H	AUT	SE	FONCTION
0	0	0	RIEN
0	0	1	RIEN
0	1	0	RIEN
0	1	1	RIEN
1	0	0	RIEN
1	0	1	RIEN
1	1	0	SIGNATURE
1	1	1	programmation du polynôme diviseur

a) table de vérité



b) schéma logique

Figure 4.8 : table de vérité et circuit de commande correspondant.

2.2.3. Simulation logique

Une simulation est effectuée sur une cellule de base de 1 bit puis sur le circuit de 8 bits afin de valider son fonctionnement. L'interprétation du programme et les fichiers de simulation (EPISODE), ainsi que leurs résultats, sont présentés dans [SOU86].

2.2.4. Circuit électrique et simulation

Les circuits électriques sont déduits des schémas logiques. Les tailles des transistors sont déterminées grâce à une simulation électrique. Le circuit de base est réalisé par l'association des sous-circuits élémentaires (bascule D, porte ET, porte OU exclusif...).

La simulation électrique est effectuée sur tous les sous-circuits, sur le circuit de commande et sur le circuit de signature 1 bit associé au circuit logique de commande [SOU86].

2.2.5. Dessin des masques

Un dessin en stick-diagram est fait pour chaque cellule de base (sous-circuit). Ensuite, un dessin au micron est effectué à la main en technologie CMOS 2 métaux (règles fines) [ANT86]. Les cellules de base sont digitalisées en utilisant le logiciel LUCIE [PAI85]. Le plan de masse du circuit complet est donné dans la figure 4.9, et le dessin au micron du circuit de signature parallèle sur 8 bits avec sa logique de commande est donné en figure 4.10.

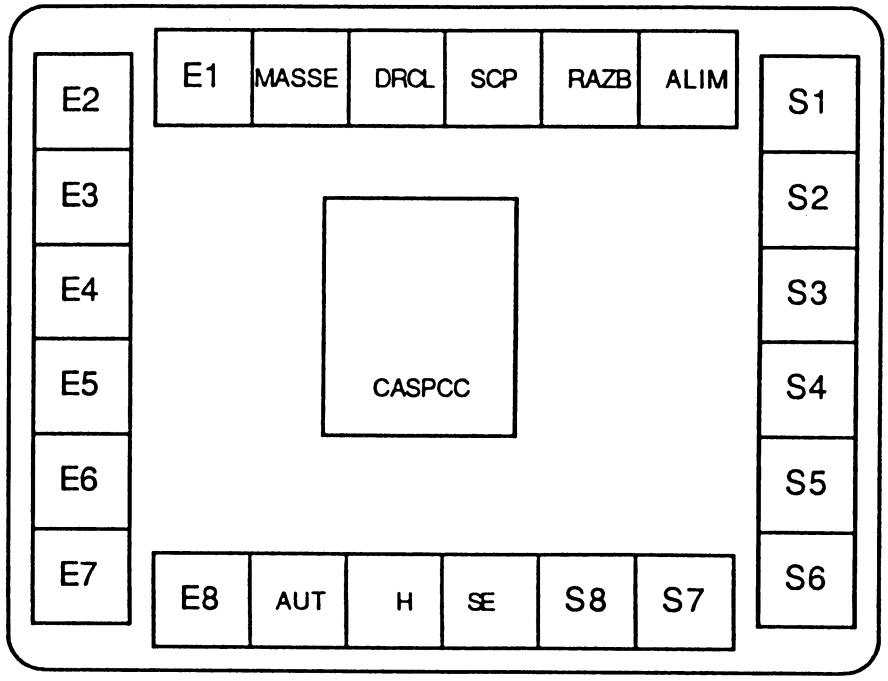


Figure 4.9 : plan de masse du circuit de signature universel.

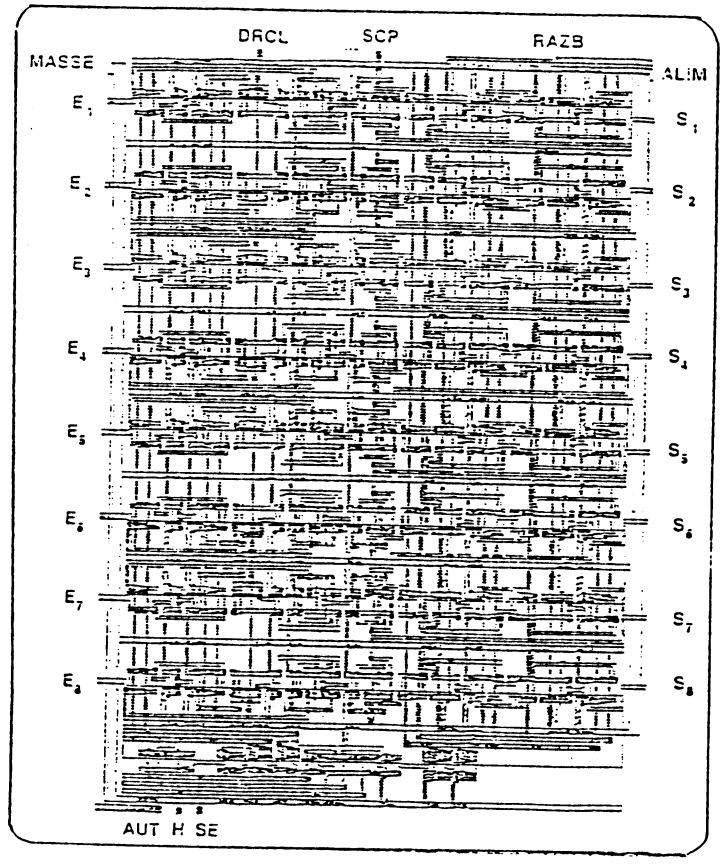


Figure 4.10 : dessin au micron du circuit de signature universel.

2.2.6. Réalisation du circuit

Il est utile de noter que la surface occupée par le circuit proprement dit, avec sa logique de commande, est de 447 × 675 μm^2 (figure 4.10). Ce circuit contient 394 transistors. Le circuit final avec ses plots d'entrées-sorties a une surface de 2720 × 2100 μm^2 . Il contient 14 plots d'entrées, 8 plots de sorties, un plot d'alimentation (Vdd) et un plot de masse (Vss).

Le montage du circuit dans un boîtier de 28 broches a été réalisé au CIME (Centre Interuniversitaire de MicroElectronique) de Grenoble. Le brochage du circuit est donné en figure 4.11.

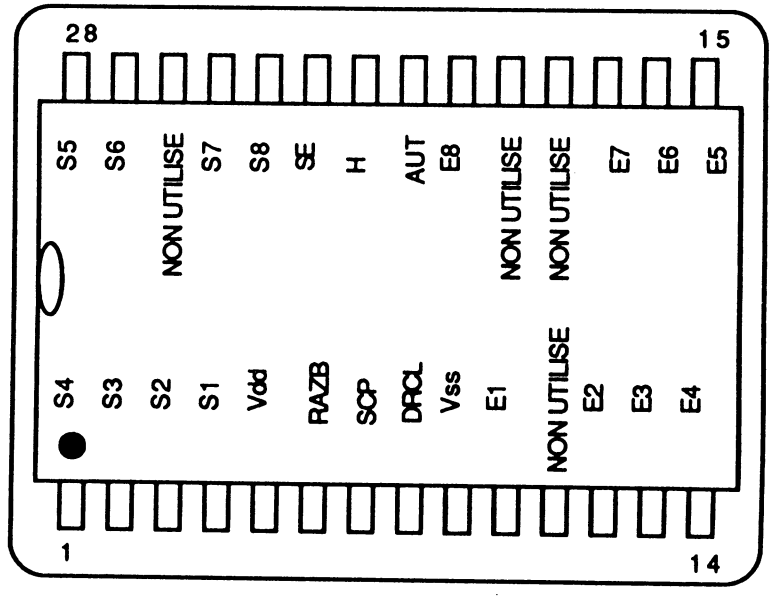


Figure 4.11 : boîtier du circuit de signature universel.

Le test du circuit a montré son bon fonctionnement.

La figure 4.12 représente un schéma d'application avec plusieurs boîtiers, l'ensemble des boîtiers formant un circuit de signature parallèle avec un format d'entrée de n fois 8 bits.

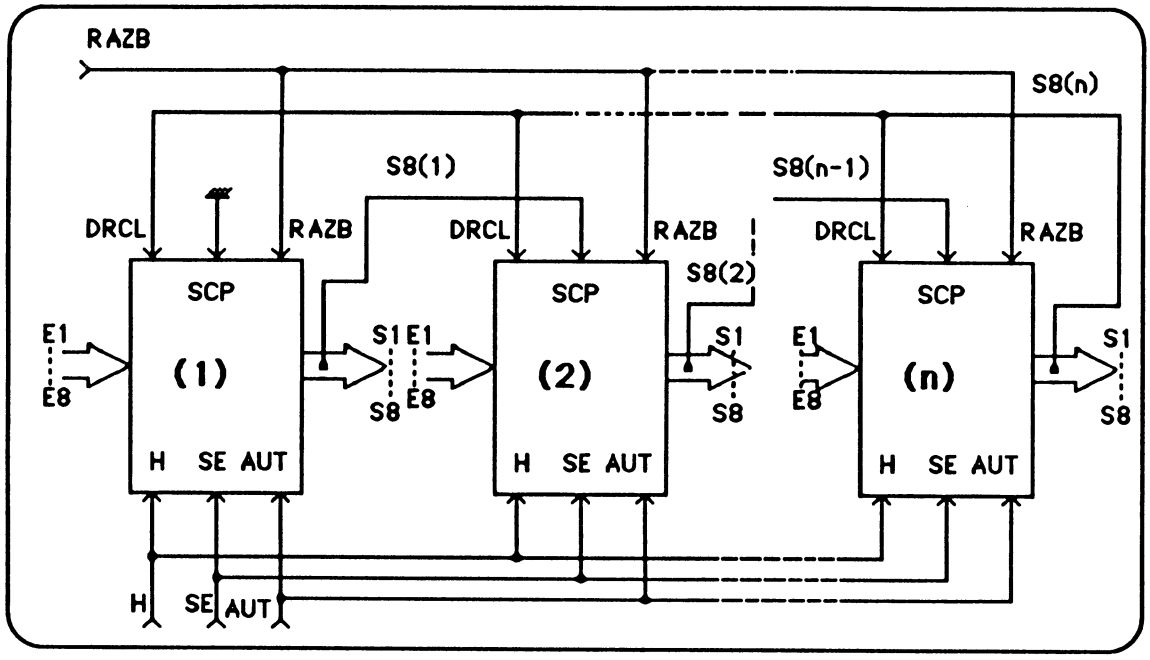


Figure 4.12 : schéma d'application de plusieurs boîtiers.

3. Circuit de signature intégré au sein du microprocesseur

Comme annoncé au chapitre 1, un circuit de compaction d'informations a été inséré dans la partie opérative du microprocesseur afin de faciliter le test en ligne des applications de contrôle temps réel. Il est clair que le circuit pourra être utilisé également à des fins d'observation compactée dans le test hors ligne de ce microprocesseur.

3.1. Description du circuit

Ce circuit comporte en plus du registre à décalage "LFSR" (SIG), deux registres programmables, l'un contenant le polynôme diviseur, et l'autre le type d'information à signer (RTIS). Un circuit combinatoire (AS) reçoit d'une part des informations de la partie contrôle (INFA, INFB), et d'autre part le contenu du registre RTIS afin de commander le registre LFSR. Ces possibilités de programmation du polynôme diviseur et du type

d'information signé rendent ce dispositif très souple d'emploi et en font toute l'originalité.

Le schéma général du dispositif de signature est donné en figure 4.13.

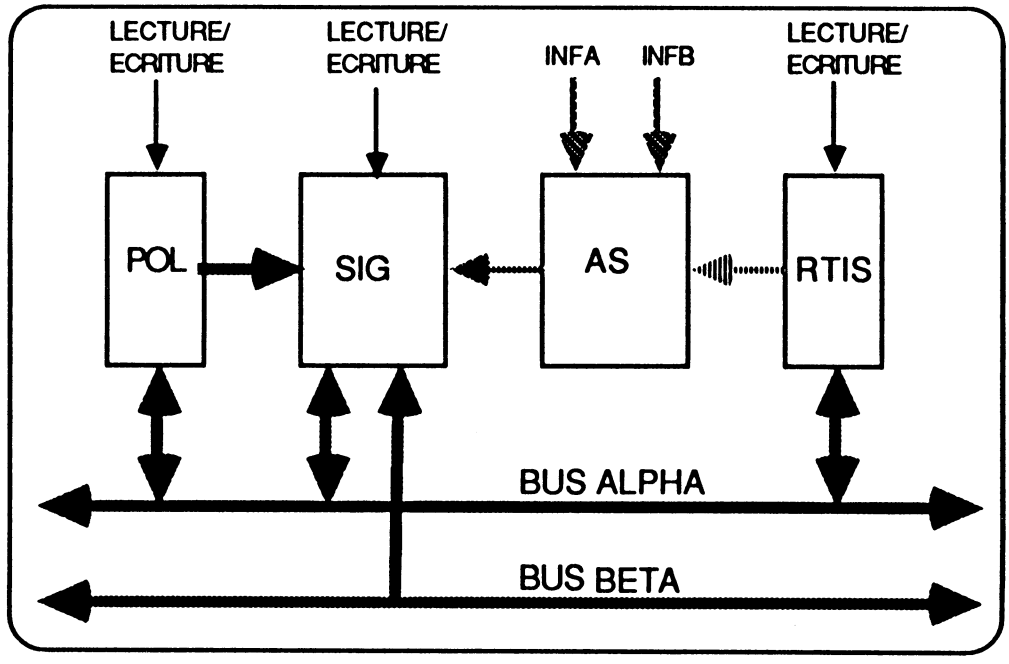


Figure 4.13 : schéma général du dispositif de signature.

- POL est un registre de 16 bits qui reçoit le polynôme diviseur du LFSR au moyen de la commande Ecriture de la PC. Il est aussi possible de lire son contenu en utilisant la commande Lecture, surtout pendant le test hors ligne.

- RTIS est un registre programmable par l'utilisateur qui permet de déterminer le type d'information à signer depuis le bus ALPHA ou BETA. Parmi les 16 bits constituant ce registre, les 8 bits de poids fort sont mis à zéro et seulement les 8 bits de poids faible ont une signification :

bit 0 = 1 : pas de signature,

bit 1 = 1 : signature de toute information sur le bus ALPHA,

bit 2 = 1 : signature de toute information sur le bus BETA,

bit 3 = 1 : signature des codes d'instructions,

bit 4 = 1 : signature des données écrites,

bit 5 = 1 : signature des données lues,

bit 6 = 1 : signature des adresses des zones données,

bit 7 = 1 : signature des adresses des zones programme.

Le bit 0 "pas de signature" a la priorité absolue sur les autres bits de RTIS. Si ce bit vaut 1, la signature sera interdite quelles que soient les valeurs des autres bits.

Les bits 1 ou 2 inhibent tous les autres bits de RTIS (le bit 0 mis à part). La signature depuis le bus BETA est plus prioritaire que celle depuis le bus ALPHA. Avec les bits 3 à 7, 2^5 configurations sont possibles.

- AS est un circuit combinatoire autorisant ou non la compaction des données en fonction du contenu du registre RTIS (bits 0 à 7) et des informations issues de la PC (INFA et INFB) indiquant à chaque cycle le type d'information circulant sur les bus internes. La signification de ces champs est donnée en figure 4.14.

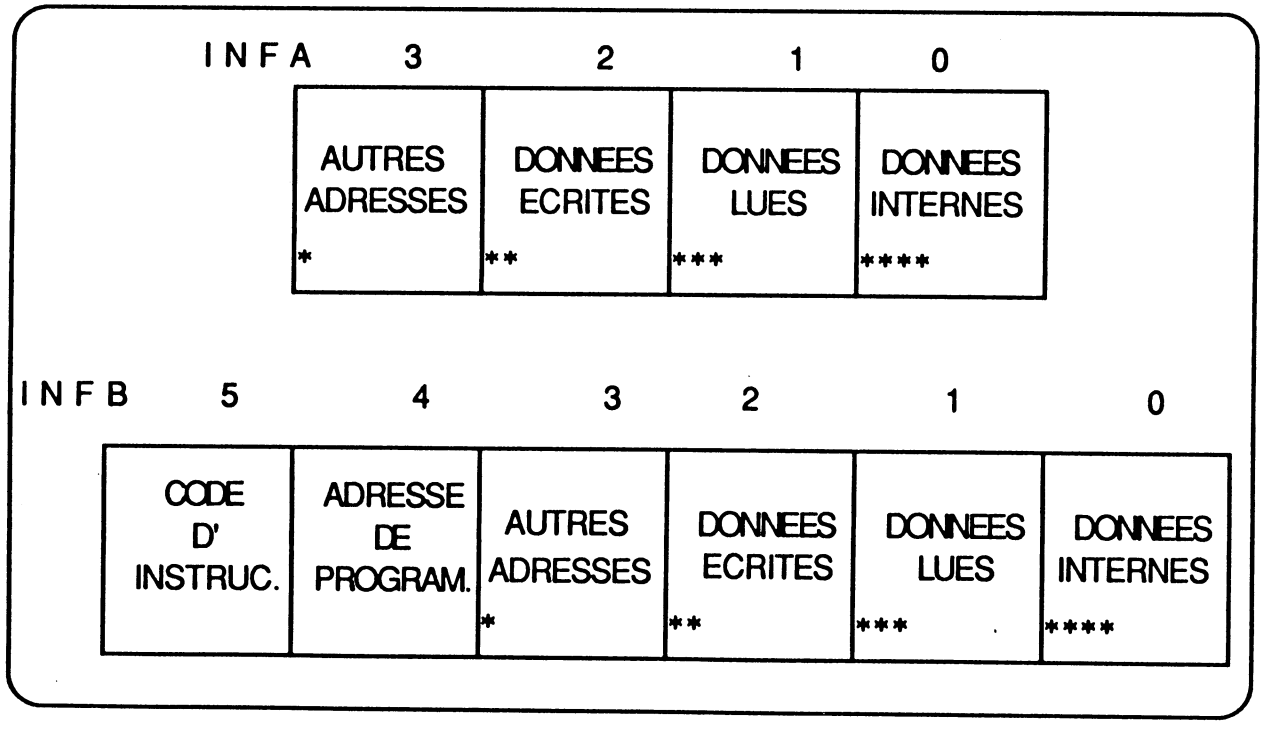


Figure 4.14 : signification des champs de commande INFA et INFB.

* adresse d'une donnée,

** données écrites dans la mémoire externe (TM —> mémoire),

*** données lues de la mémoire externe (mémoire —> TM),

**** données circulant sur le bus ALPHA.

Trois signaux de commande (CK, ASALP, ASBET) fournis par le circuit AS au LFSR sont déterminés par les équations suivantes :

1) $ASALPH = \overline{POLSA} \cdot \overline{POLEA} \cdot \overline{SIGSA} \cdot \overline{SIGEA} \cdot \overline{RTISSA} \cdot \overline{RTISEA} \cdot \overline{RTIS0}$

$\overline{ASBETA}$ [RTIS1 (INFA0 + INFA1 + INFA2 + INFA3) +

RTIS4 . INFA2 + RTIS5 . INFA1 + RTIS6 . INFA3]

2) $ASBETA = \overline{POLSA} \cdot \overline{POLEA} \cdot \overline{SIGSA} \cdot \overline{SIGEA} \cdot \overline{RTISSA} \cdot \overline{RTISEA} \cdot \overline{RTIS0}$

[RTIS2 (INFB0 + INFB1 + INFB2 + INFB3 + INFB4 + INFB5)

+ RTIS3 . INFB5 + RTIS4 . INFB2 + RTIS5 . INFB1 + RTIS6 .

INFB3 + RTIS7 . INFB4]

3) CK = ASALPHA + ASBETA

Le schéma complet du sous-circuit AS déduit des équations précédentes est donné dans la figure 4.15.

- SIGN est un registre LFSR parallèle et programmable qui peut être chargé depuis le bus ALPHA afin de l'initialiser.

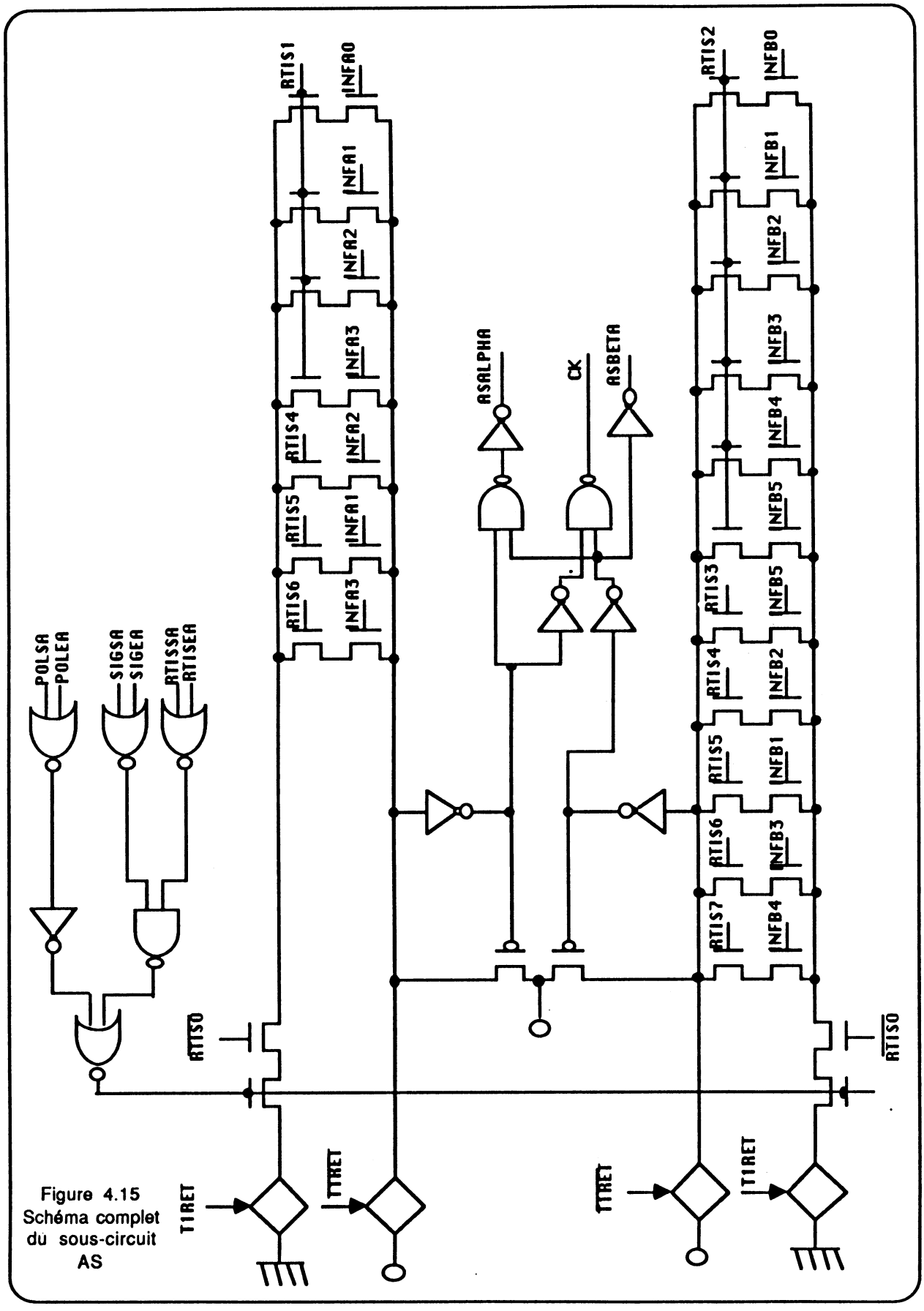


Figure 4.15
Schéma complet du sous-circuit AS

Le schéma logique d'un bit du registre LFSR est donné en figure 4.16.

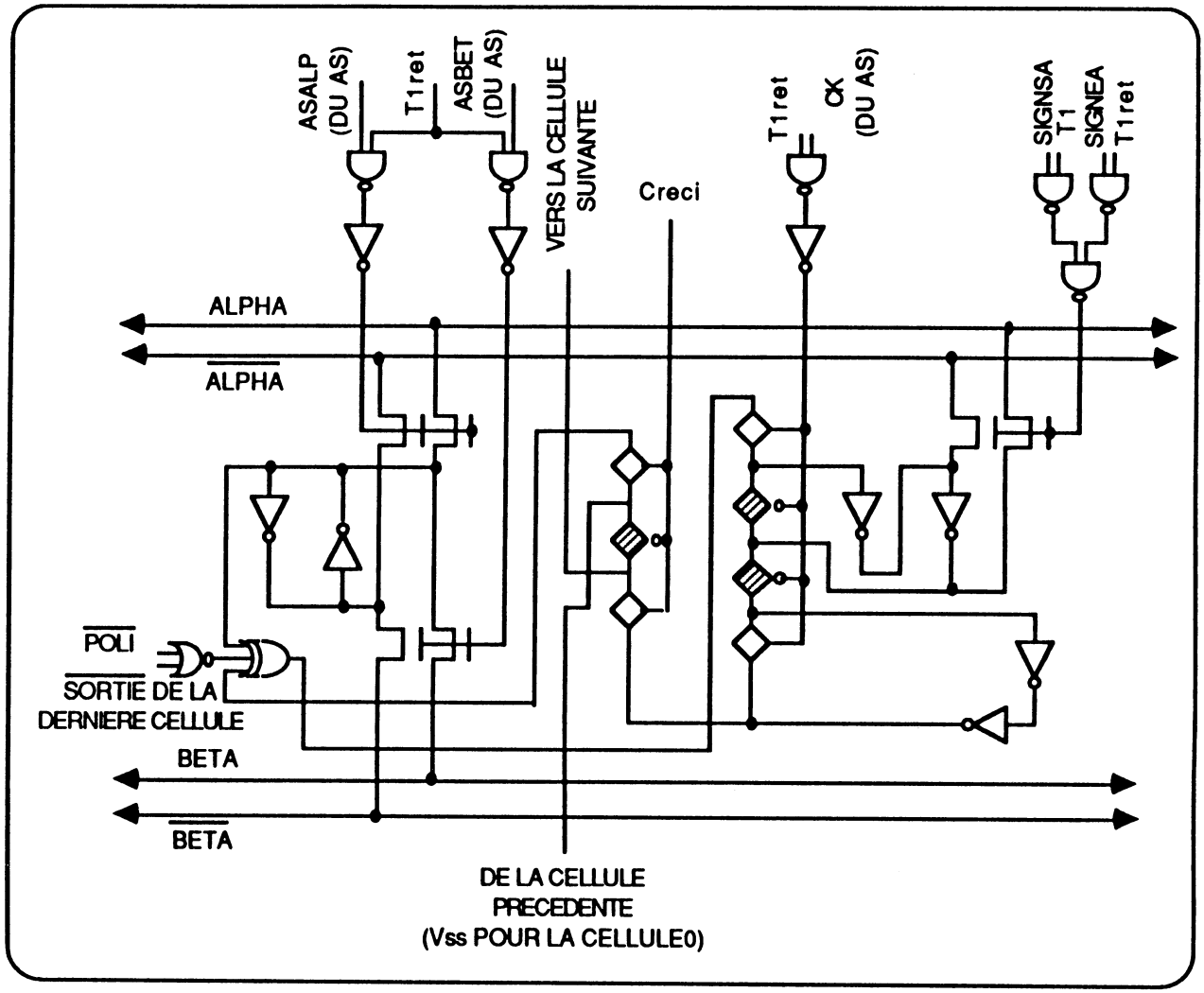


Figure 4.16 : schéma logique d'un bit du registre SIG.

3.2. Réalisation du circuit

Le dispositif de signature inséré dans la partie opérative du HYETI a été dessiné en deux parties :

- POL et SIG dans un bloc reconfigurable,
- AS et RTIS dans un bloc non reconfigurable.

Ce dispositif contient 1456 transistors et a une superficie de 573,5 × 1614,8 μm^2. Ceci prend en compte les éléments de

reconfiguration et la logique de commande. De plus, un circuit de signature de 5 bits a été dessiné dans le but de valider la conception du dispositif de signature intégré au sein du microprocesseur. Sa fabrication a été effectuée par la société SGS-Thomson Microelectronics en Technologie HCMOS3C 1,2 µm. Le plan de masse du circuit de 5 bits et son dessin au micron sont donnés successivement en figures 4.17 et 4.18.

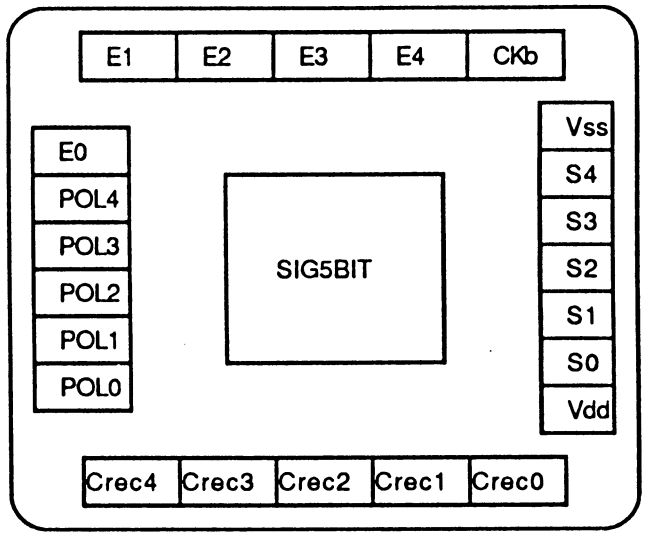


Figure 4.17 : plan de masse du circuit de signature sur 5 bits.

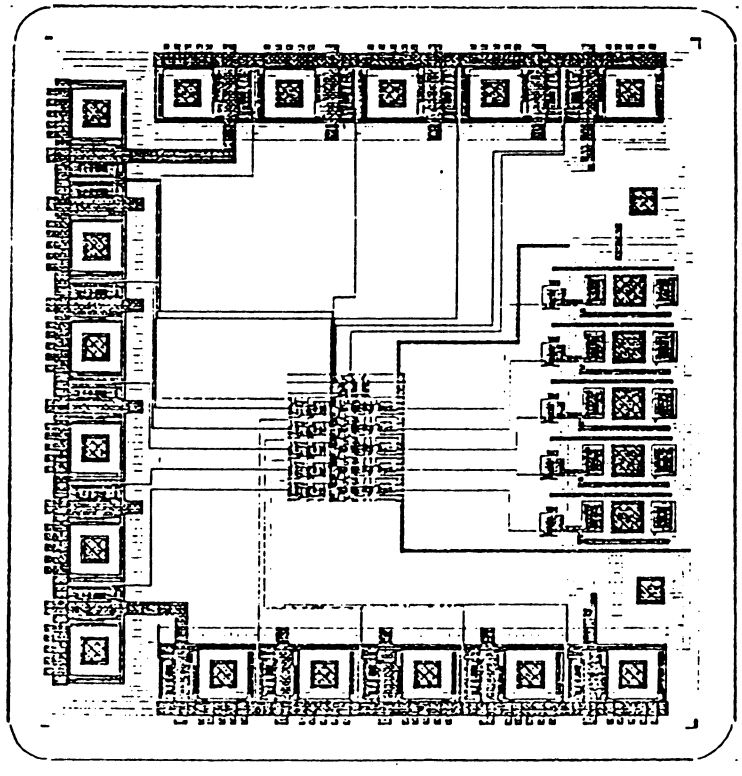


Figure 4.18 : Dessin au micron du circuit de signature sur 5 bits.

La surface occupée par le circuit sans les plots d'entrée/sortie est de 441,1 × 481,3 μm^2. Cette surface contient 318 transistors.

4.Test et testabilité

Nous présentons dans ce paragraphe une extension de la méthode classique du test exhaustif des circuits combinatoires itératifs aux circuits séquentiels à base de registre à décalage (LFSR), tels que l'analyseur de signature. Ce type de circuit étant implanté pour le test en ligne des circuits à haute sûreté de fonctionnement, il est impératif d'en assurer une bonne validation en fin de conception et de fabrication.

La méthode développée ici permet entre autres un coût minimal du test de fin de fabrication avec une excellente couverture.

Notre étude concerne essentiellement le registre LFSR série et parallèle programmable.

Notre méthode est fondée sur le principe de test des circuits combinatoires itératifs unidirectionnels avec les hypothèses générales des erreurs classiques définies par Kautz [KAU67] et prises en compte par différents auteurs [FRI73], [DIA76], [SRI81], [VAR88], [ENS88].

4.1. Rappel de la méthode du test de circuits itératifs

Les différentes idées concernant le test et la testabilité des circuits logiques itératifs telles que le modèle d'erreur utilisé, les conditions de testabilité des réseaux itératifs, la C-testabilité et le RT-graphe sont rappelées dans le chapitre 3 (§ 1.3).

4.2. Extension à des circuits de signature

La méthode de test des circuits itératifs logiques peut être étendue aux circuits de signature à entrée série ou à entrées parallèles ayant des polynômes diviseurs programmables.

4.2.1. Test des circuits de signature à entrée série

Le schéma général du circuit de signature série avec un polynôme diviseur programmable est donné en figure 4.19.

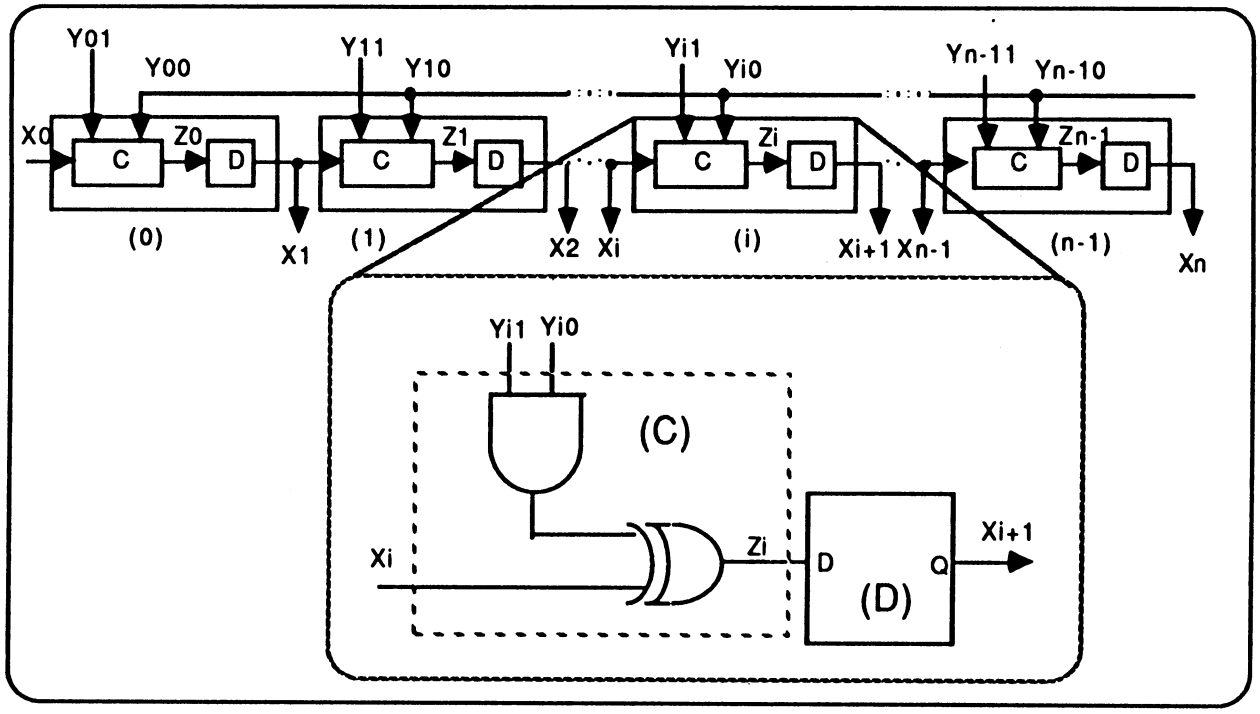


Figure 4.19 : schéma général du circuit de signature à entrée série.

Ce schéma représente un circuit itératif unidirectionnel (il n'y a pas de rebouclage) formé de n cellules identiques. La cellule de base est constituée d'un circuit combinatoire C et d'une bascule D. Le circuit combinatoire reçoit une entrée Xi de la cellule gauche voisine (Xi représente en même temps la sortie de la cellule i-1) et deux entrées

externes contrôlables Yi1 (coefficient du polynôme diviseur) et Yi0 (report de sortie de la dernière cellule). Il génère la sortie Zi qui est l'entrée de la bascule D. Cette bascule transmet sa sortie Xi+1 à l'entrée de la cellule droite voisine. A l'exception de l'entrée X0 (entrée contrôlable), les Xi ($1 \leq i \leq n-1$) sont en même temps des entrées non directement contrôlables et des sorties observables.

Le modèle d'erreur considéré ici est celui rappelé au chapitre 3 (§3.1 définition 2). Il s'agit des hypothèses générales d'erreur concernant la détection des erreurs simples. Cependant la détection des erreurs multiples peut être envisagée.

L'étude du graphe des états déduit du tableau des états de la partie combinatoire de la cellule de base montre qu'il est de type RT-graphe, car ce graphe n'a que des transitions répétables (cf. figure 4.20b). Ceci implique que le circuit combinatoire complet est de type C-testable, c'est-à-dire que le nombre de vecteurs de test exhaustif ne dépend que du nombre d'entrées de la cellule de base. Nous supposons que la bascule D est exhaustivement testée si son entrée D reçoit les deux valeurs logiques possibles (0 et 1) pour les deux états de sa sortie. On constate alors que la bascule D est exhaustivement testée par la séquence de test du circuit combinatoire.

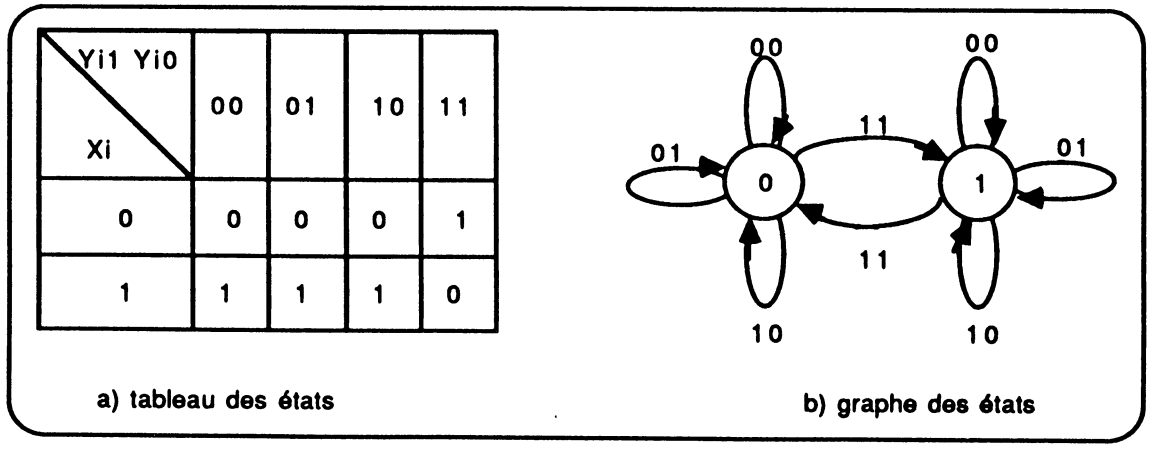


Figure 4.20 : tableau des états et son graphe associé.

Pour établir la séquence de vecteurs de test exhaustif d'un tel circuit, on suppose que le rebouclage n'est pas fait pendant la phase de test. Les entrées Yi0 reçoivent la même valeur logique à un instant donné. En fait, il n'y a pas de procédure générale donnant une séquence minimale. Le principe général consiste à appliquer toutes les combinaisons possibles sur les entrées de la première cellule et à chercher une séquence appliquée aux entrées des cellules suivantes de façon à pouvoir propager toutes les valeurs appliquées à l'entrée externe X0 à toute autre entrée interne Xi.

Dans le cas du circuit étudié, la séquence de vecteurs de test de la première cellule est obtenue en parcourant tous les états et les arcs sans repasser 2 fois par le même arc. Les mêmes valeurs appliquées à l'entrée externe X0 se retrouvent à la sortie X1. Donc, pour que les valeurs de X1 se propagent sur les entrées internes Xi des cellules suivantes, la même séquence des entrées externes (Y00,Y01) de la première cellule peut être appliquée aux entrées (Yi1, Yi0) des cellules suivantes (1 à n-1). Le tableau 4.1 donne la séquence de vecteurs de test exhaustif du circuit de signature à entrée série.

Le nombre de vecteurs de test est de 8 quel que soit le nombre de cellules constituant le circuit (cf. tableau 4.1).

Séquences de vecteurs de test	X0	Y01	Y00	Z0	X1	Y11	Y10	Z1	X2	...	Xn-1	Yn-11	Yn-10	Zn-1	Xn
1	0	0	0	0	0	0	0	0	0		0	0	0	0	0
2	0	0	1	0	0	0	1	0	0		0	0	1	0	0
3	0	1	0	0	0	1	0	0	0		0	1	0	0	0
4	0	1	1	1	0	1	1	1	0		0	1	1	1	0
5	1	0	0	1	1	0	0	1	1		1	0	0	1	1
6	1	0	1	1	1	0	1	1	1		1	0	1	1	1
7	1	1	0	1	1	1	0	1	1		1	1	0	1	1
8	1	1	1	0	1	1	1	0	1		1	1	1	0	1

Tableau 4.1 : séquences de vecteurs de test exhaustif du circuit de signature à entrée série

4.2.2. Test d'un circuit de signature à entrées parallèles avec un polynôme diviseur programmable

Le schéma général d'un circuit de signature à entrées parallèles avec un polynôme diviseur programmable est donné en figure 4.21.

Comme le circuit de signature à entrée série, le circuit de signature à entrées parallèles est aussi un circuit itératif dont la cellule de base est constituée d'une partie combinatoire et d'une bascule D. On suppose que le rebouclage n'est pas établi pendant la phase de test (la sortie Xn de la dernière cellule n'est pas reliée aux entrées Yi0) et que l'entrée X0 est contrôlable. La partie combinatoire de la cellule de base reçoit 4 entrées, Xi étant la sortie de la cellule gauche voisine, Yi0 étant le report de la sortie de la dernière cellule pendant le fonctionnement normal du circuit

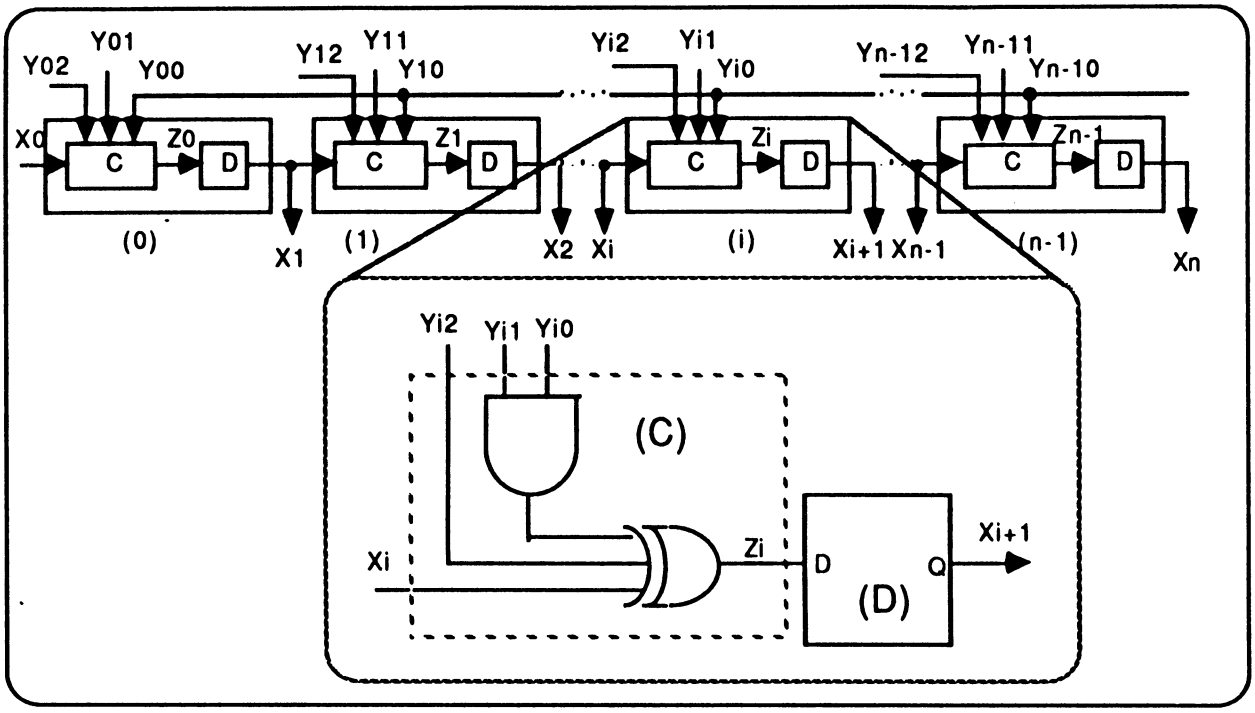


Figure 4.21 : schéma général du circuit de signature à entrées parallèles avec un polynôme diviseur programmable.

(toutes les entrées Yi0 sont reliées entre elles formant une seule entrée), Yi1 étant le $i^{ème}$ coefficient du polynôme diviseur et Yi2 étant la $i^{ème}$ donnée à signer. La sortie de la partie combinatoire de la cellule de base est l'entrée de la bascule D qui transmet sa sortie Xi+1 à l'entrée de la cellule droite voisine. Pendant la phase de test, les entrées Yi0, Yi1, Yi2 ($0 \leq i \leq n-1$) et X0 sont contrôlables et les sorties Xi ($1 \leq i \leq n$) sont observables.

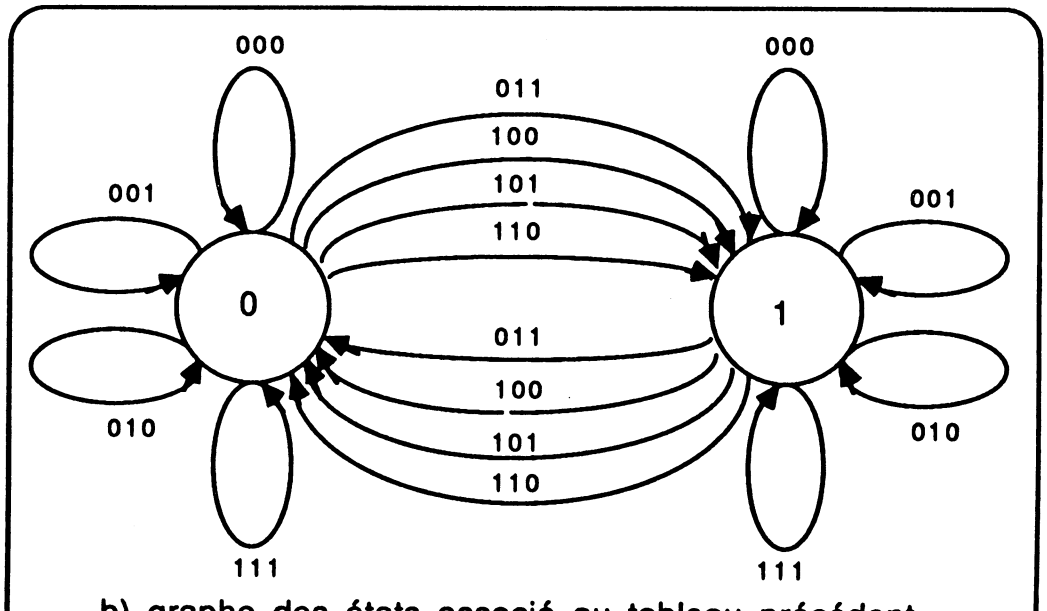
Le modèle d'erreur utilisé est celui défini dans le chapitre 3 (§1.3).

A partir du schéma logique de la cellule de base, le tableau des états de la partie combinatoire est établi (cf. figure 4.22a). Ce tableau décrit le fonctionnement de la cellule i. Il donne en fait les valeurs de la sortie Zi en fonction des valeurs des entrées Yi0, Yi1, Yi2, et Xi. Le graphe des états associé au tableau des états est déduit (cf. figure 4.22b). Ce graphe n'a que des transitions répétables, il est donc de type RT-graphe. On suppose que la bascule D est testée d'une façon exhaustive si son entrée reçoit les deux

valeurs logiques possibles (0 et 1) pour les deux états de sa sortie. On constate que le circuit complet est de type C-testable. Le nombre de vecteurs de test exhaustif ne dépend que du nombre d'entrées de la cellule

Yi2, Yi1, Yi0 / Xi	000	001	010	011	100	101	110	111
0	0	0	0	1	1	1	1	0
1	1	1	1	0	0	0	0	1

a) table des états de la cellule de base



b) graphe des états associé au tableau précédent

Figure 4.22 : tableau des états et graphe correspondant à la cellule de base d'un circuit de signature à entrées parallèles.

de base (4), il est de $2^4 = 16$. Le tableau 4.2 donne la séquence de vecteurs de test exhaustif de ce circuit. On constate que ces 16 vecteurs de test réalisent un test exhaustif de la partie combinatoire et de la bascule D de chaque cellule dans le circuit.

Séquence de test	X0	Y02	Y01	Y00	Z0	X1	....	Xn-1	Yn-12	Yn-11	Yn-10	Zn-1	Xn
1	0	0	1	1	1	0	....	0	0	1	1	1	0
2	1	0	0	0	1	1	....	1	0	0	0	1	1
3	1	0	1	1	0	1	....	1	0	1	1	0	1
4	0	0	0	0	0	0	....	0	0	0	0	0	0
5	0	1	0	0	1	0	....	0	1	0	0	1	0
6	1	0	0	1	1	1	....	1	0	0	1	1	1
7	1	1	0	0	0	1	....	1	1	0	0	0	1
8	0	0	0	1	0	0	....	0	0	0	1	0	0
9	0	1	0	1	1	0	....	0	1	0	1	1	0
10	1	0	1	0	1	1	....	1	0	1	0	1	1
11	1	1	0	1	0	1	....	1	1	0	1	0	1
12	0	0	1	0	0	0	....	0	0	1	0	0	0
13	0	1	1	0	1	0	....	0	1	1	0	1	0
14	1	1	1	1	1	1	....	1	1	1	1	1	1
15	1	1	1	0	0	1	....	1	1	1	0	0	1
16	0	1	1	1	0	0	....	0	1	1	1	0	0

Tableau 4.2 : séquence de vecteurs de test exhaustif du circuit de signature à entrées parallèles avec un polynôme diviseur programmable.

4.2.2.1. Application de la méthode de test au circuit de signature parallèle sur 8 bits cascadable

Le schéma de principe est celui donné en figure 4.21 mais sur 8 bits (n=8). Le schéma logique de la cellule de base, le tableau des états et le

graphe des états sont donc ceux donnés en figures 4.21 et 4.22.

Le tableau 4.2 donne l'ensemble des vecteurs de test exhaustif du circuit.

Une simulation logique a été effectuée en utilisant les 16 vecteurs de test exhaustif avec les notations des entrées et des sorties suivantes : Ei ≡ Yi2, Pi ≡ Yi1, DRC ≡ Yi0, SIM1 ≡ X0, Si ≡ Xi+1. Le résultat (figure 4.23) permet de valider la méthode proposée.

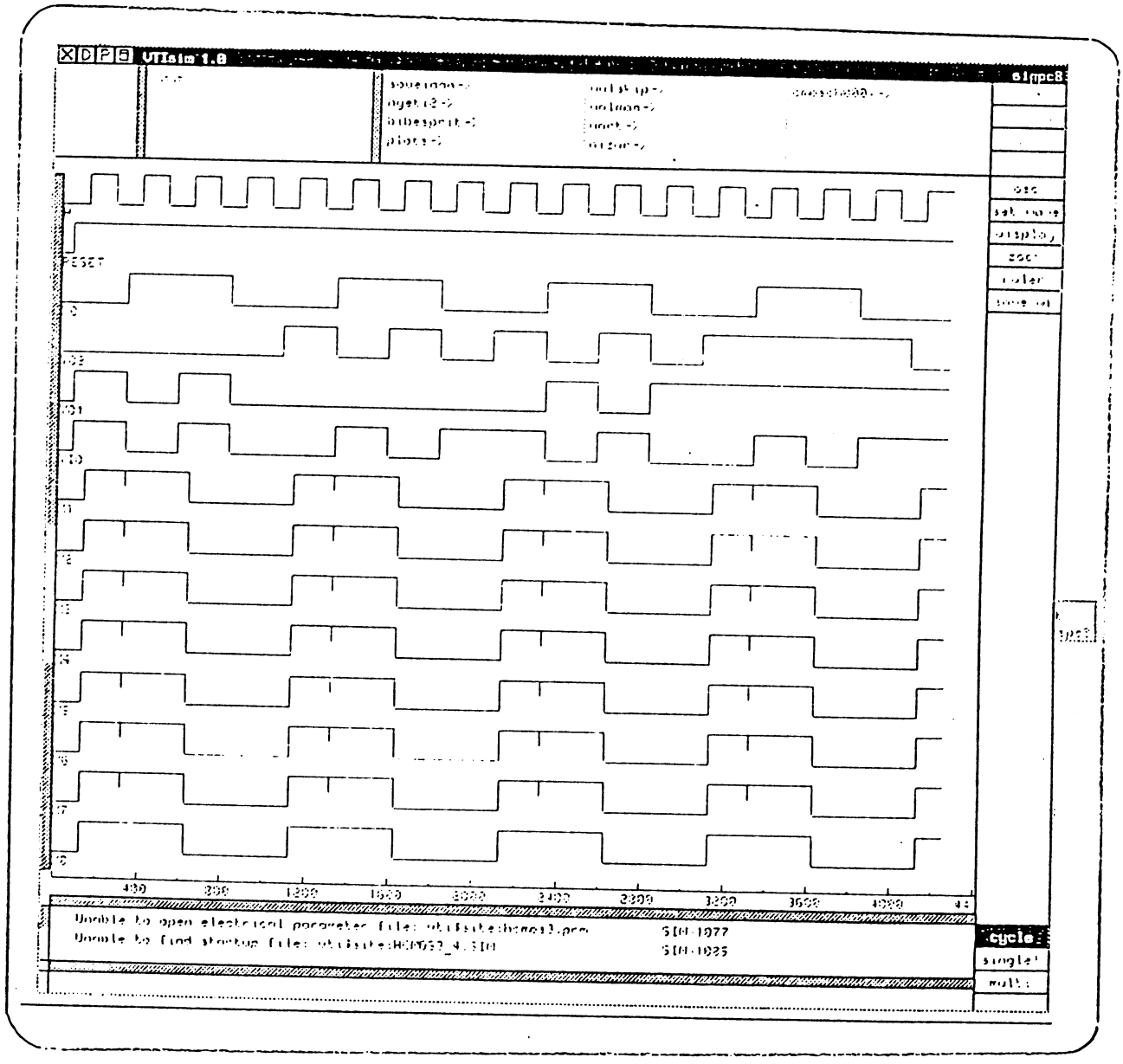


Figure 4.23 : Résultat de la simulation logique.

4.2.2.2. Application de la méthode de test au circuit de signature intégré au sein du microprocesseur

On recherche tout d'abord un test fonctionnel exhaustif défini comme précédemment. Des modifications du circuit seront proposées afin de rendre ce test possible.

Dans un deuxième temps, une séquence de vecteurs de test quasi-exhaustif sera proposée sans aucune modification du circuit.

a) Test du circuit modifié

Le schéma de principe du circuit de signature implanté au sein du microprocesseur est celui donné en figure 4.21 avec les modifications suivantes :

- au niveau de la cellule de base, la porte "AND" réalisant la fonction Yi0 . Yi1 est remplacée par une porte "NOR", à deux entrées complémentées, donnant la même fonction;

$$\overline{\overline{Yi0} + \overline{Yi1}} = Yi0 \,.\, Yi1$$

- au niveau du circuit, l'entrée X0 est reliée à la masse en permanence et la sortie complémentée de la dernière cellule est reliée aux entrées Yi0 complémentées ($0 \leq i \leq 15$).

Dans le but de rendre, pendant la phase de test, le schéma du circuit de signature intégré au sein du microprocesseur identique au schéma du circuit général donné en figure 4.23, deux multiplexeurs à deux entrées sont introduits (cf. figure 4.26). Ils sont commandés par la commande externe TEST. De plus, outre le plot de la commande TEST, deux plots supplémentaires (X0, Yi0) sont rajoutés. Deux modes de fonctionnement sont définis et résumés dans le tableau 4.3.

TEST	FONCTION	REMARQUES
0	$X0 = 0$, $\overline{Yi0} = \overline{X16}$	MODE NORMAL
1	X0 ← PLOT X0, $\overline{Yi0}$ ← PLOT $\overline{Yi0}$	MODE TEST

Tableau 4.3 : modes de fonctionnement du circuit de signature modifié.

La structure du circuit en mode test (cf. figure 4.24), composé des 16 cellules identiques, est de type itératif.

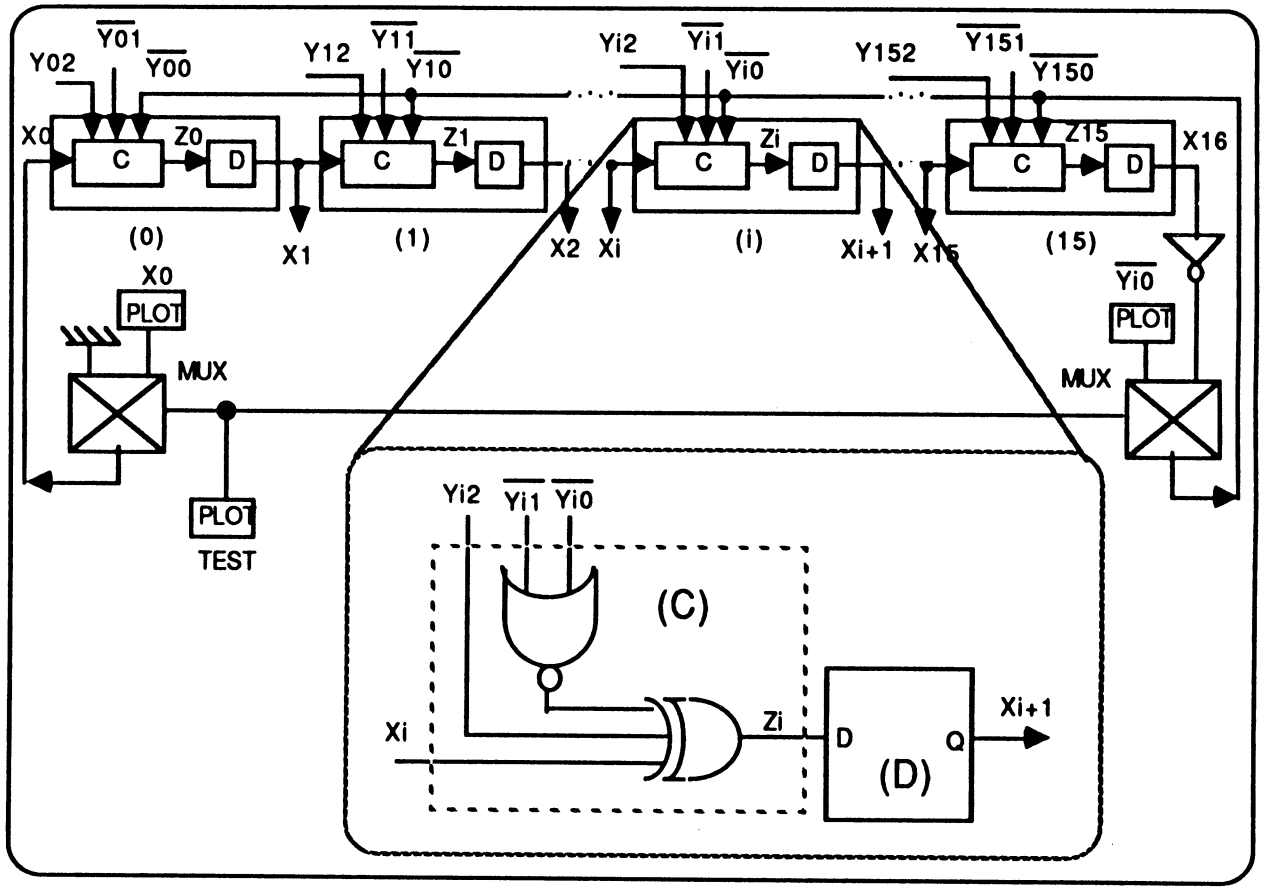


Figure 4.24 : schéma du circuit de signature à entrées parallèles modifié.

A partir du schéma logique de la cellule de base (cf. figure 4.24), le tableau des états est établi (cf. figure 4.25a). Ce tableau donne les valeurs de la sortie de la partie combinatoire Zi en fonction de toutes les combinaisons possibles appliquées aux entrées suivantes :

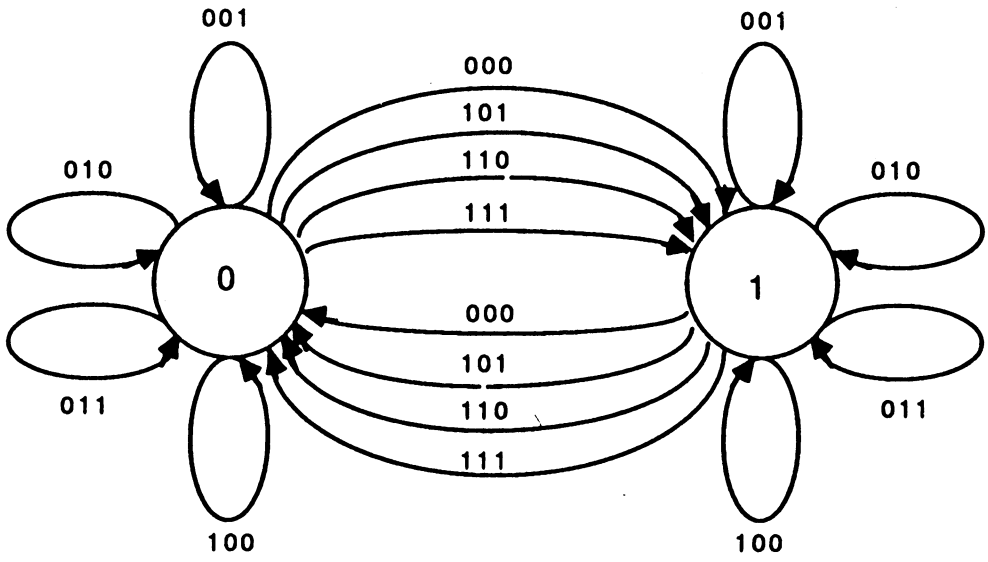
Xi, Yi2, $\overline{Yi1}$ et $\overline{Yi0}$

Nous en déduisons le graphe des états associé (cf. figure 4.25b).

On constate que le graphe des états est maintenant de type RT-graphe. Par conséquent, le circuit est de type C-testable. Comme le nombre des entrées de la cellule de base est de 4, le nombre de vecteurs permettant le test exhaustif est de $2^4 = 16$.

Xi \ Yi2, $\overline{Yi1}$, $\overline{Yi0}$	000	001	010	011	100	101	110	111
0	1	0	0	0	0	1	1	1
1	0	1	1	1	1	0	0	0

a) tableau des états de la cellule de base



b) graphe des états associé au tableau précédent

Figure 4.25 : tableau des états et graphe de la cellule de base du circuit de signature intégré au sein du microprocesseur.

Le tableau 4.4 donne la séquence des vecteurs de test exhaustif du circuit.

Séquence de test	X0	Y02	$\overline{Y01}$	$\overline{Y00}$	Z0	X1	····	Xn-1	Yn-12	$\overline{Yn-11}$	$\overline{Yn-10}$	Zn-1	Xn
1	0	0	0	0	1	0	····	0	0	0	0	1	0
2	1	0	0	1	1	1	····	1	0	0	1	1	1
3	1	0	0	0	0	1	····	1	0	0	0	0	1
4	0	0	0	1	0	0	····	0	0	0	1	0	0
5	0	1	0	1	1	0	····	0	1	0	1	1	0
6	1	0	1	0	1	1	····	1	0	1	0	1	1
7	1	1	0	1	0	1	····	1	1	0	1	0	1
8	0	0	1	0	0	0	····	0	0	1	0	0	0
9	0	1	1	0	1	0	····	0	1	1	0	1	0
10	1	0	1	1	1	1	····	1	0	1	1	1	1
11	1	1	1	0	0	1	····	1	1	1	0	0	1
12	0	0	1	1	0	0	····	0	0	1	1	0	0
13	0	1	1	1	1	0	····	0	1	1	1	1	0
14	1	1	0	0	1	1	····	1	1	0	0	1	1
15	1	1	1	1	0	1	····	1	1	1	1	0	1
16	0	1	0	0	0	0	····	0	1	0	0	0	0

Tableau 4.4 : séquence des vecteurs de test exhaustif du circuit modifié.

L'obtention de cette séquence se fait en parcourant tous les états et les arcs sans repasser deux fois par le même arc.

b) Test du circuit non modifié

Si les contraintes en surface, ne permettent absolument pas de modifier

le circuit, nous proposons une méthode de test adapté avec une augmentation du nombre de vecteurs de test.

Le schéma général du circuit non modifié est donné en figure 4.26.

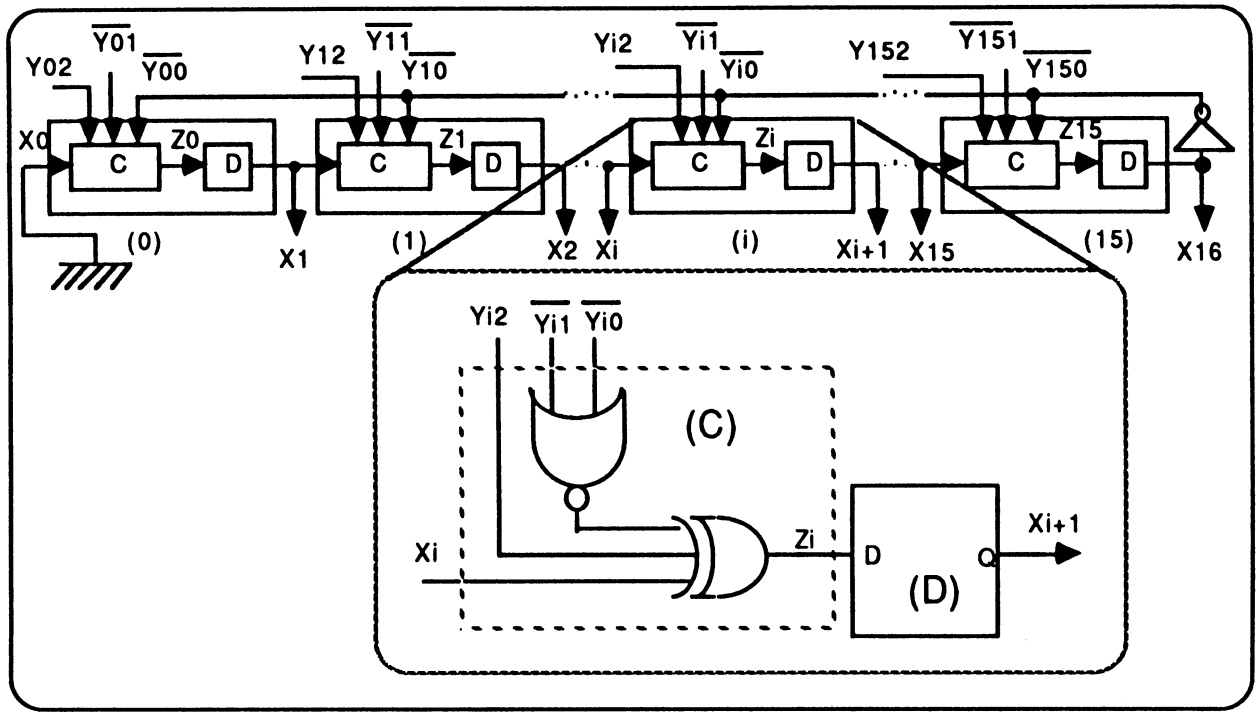


Figure 4.26 : schéma général du circuit non modifié.

Le schéma logique de la cellule de base du circuit non modifié est identique au précédent. Donc, le tableau des états et le graphe associé sont ceux donnés pour le circuit modifié (cf. figure 4.25).

A l'exception de la première et de la dernière cellule, le circuit est testable avec la séquence de 16 vecteurs de test désignés pour le circuit modifié. L'obtention des vecteurs de test de la première cellule se fait en recherchant une séquence d'entrées (Y02,Y01) donnant les valeurs souhaitées sur les sorties (Xi+1). Ceci est possible avec l'hypothèse que X0 ait 0 en permanence et que Yi0 puisse recevoir les valeurs désirées pour les cellules 1 à 14. Pour la dernière cellule, on cherche une séquence à appliquer aux entrées Y140 et Y141, connaissant les valeurs reçues de la cellule gauche voisine (13) sur l'entrée X14, pour générer les valeurs

désirées à appliquer sur toutes les entrées Yi0 à partir de la sortie de la dernière cellule X16.

Le tableau 4.5 donne la séquence de test exhaustif du circuit non modifié. On constate que les 16 premiers vecteurs recouvrent un test exhaustif concernant les cellules de 0 à 14. Ces 16 vecteurs permettent d'appliquer 12 combinaisons parmi les 16 possibles aux entrées de la cellule 15. Les vecteurs de 17 à 21 permettent d'appliquer les 4 combinaisons restantes. En fait, l'initialisation des sorties des cellules aux valeurs indiquées par le vecteur 17 est indispensable. Un vecteur supplémentaire (19) est aussi nécessaire pour pouvoir appliquer les deux dernières combinaisons.

Le nombre de vecteurs de test exhaustif du circuit sans aucune modification est de 21. L'initialisation avant l'application du vecteur 1 et celle avant l'application du vecteur 17 ne sont pas comptées.

4.2.3. Conclusion

La méthode proposée réalise un test fonctionnel quasi-exhaustif avec 16 vecteurs de test pour le circuit de signature sur 8 bits parallèles, programmable et cascadable .

La même méthode peut être appliquée sur le circuit de signature intégré au sein du microprocesseur. Ceci exige une modification de l'architecture du circuit de signature. Le nombre de vecteurs de test exhaustif est de 16 seulement quelle que soit la longueur du circuit.

Afin d'éviter cette modification, une procédure manuelle permettant un test exhaustif avec 21 vecteurs de test est proposée, soit 25% de plus de la longueur de la séquence initiale.

Séquence de test	X0	Y02	$\overline{Y01}$	$\overline{Y00}$	Z0	X1	····	X15	Y152	$\overline{Y151}$	$\overline{Y150}$	Z15	X16
1	0	0	0	0	1	0	····	0	0	1	0	0	1
2	0	1	0	1	1	1	····	1	0	0	1	1	0
3	0	0	1	0	0	1	····	1	0	0	0	0	1
4	0	0	0	1	0	0	····	0	0	0	1	0	0
5	0	1	1	1	1	0	····	0	1	0	1	1	0
6	0	1	1	0	1	1	····	1	1	1	0	0	1
7	0	0	1	1	0	1	····	1	0	1	1	1	0
8	0	1	0	0	0	0	····	0	0	0	0	1	1
9	0	1	1	0	1	0	····	0	1	0	0	0	1
10	0	1	0	1	1	1	····	1	0	0	1	1	0
11	0	0	1	0	0	1	····	1	1	1	0	0	1
12	0	0	0	1	0	0	····	0	0	1	1	0	0
13	0	1	1	1	1	0	····	0	1	1	1	1	0
14	0	0	0	0	1	1	····	1	0	0	0	0	1
15	0	0	0	1	0	1	····	1	0	0	1	1	0
16	0	1	0	0	0	0	····	0	1	1	0	1	1
17	0	0	0	0	1	1	····	1	0	1	0	1	1
18	0	0	0	0	1	1	····	1	1	0	0	1	1
19	0	0	0	0	1	1	····	1	0	0	0	0	1
20	0	1	1	1	1	1	····	1	1	0	1	0	0
21	0	1	1	1	1	1	····	1	1	1	1	0	0

Tableau 4.5 : séquence des vecteurs de test exhaustif du circuit non modifié

Conclusion générale

Il est possible de distinguer deux sortes de retombées à cette thèse. Les premières sont aussi précises que claires et concernent les techniques de reconfiguration de certains blocs dans les circuits intégrés. Pour les Unités Arithmétiques et Logiques (UAL), notre étude a été menée jusqu'au bout, c'est-à-dire jusqu'à l'implantation sur silicium des structures les plus intéressantes compte-tenu de nos critères. Les compromis entre performances, surface, facilité de test (C-testabilité) et reconfigurabilité ont été discutés et une comparaison multi critère chiffrée est maintenant disponible. La reconfigurabilité des PLAs, à un coût raisonnable, a par ailleurs été démontrée par nos partenaires du projet ESPRIT 824. Les difficultés que l'on pouvait prévoir ont été surmontées. En particulier, la génération automatique de vecteurs de test permettant la localisation des défauts au niveau des monômes a été maîtrisée et l'étude est concluante. Il s'agit donc de conclusions incontestables concernant des blocs d'usage très général.

De façon plus large, il est possible de définir une certaine catégorie de blocs aisément reconfigurables avec les techniques étudiées. Il s'agit des blocs fonctionnels du chemin de données qui, implantés selon une structure en tranches, sont strictement semblables sur l'ensemble des tranches. De tels blocs forment un réseau itératif unidimensionnel avec une cellule sur chaque tranche. Dans une telle configuration, toute cellule défectueuse peut être remplacée par la cellule de la tranche de réserve et le coût de la reconfiguration reste faible. Ainsi, dans le microprocesseur étudié, de nombreux éléments ont été reconfigurés en plus de l'UAL. Il s'agit essentiellement du dispositif d'analyse de signature, du dispositif de masquage, de la logique de bus (amplification et précharge) et des registres généraux. Il faut également noter que les interconnexions externes des blocs considérés doivent être prises en compte. Ces

interconnexions doivent être incluses dans la structure itérative régulière pour que le bloc puisse être considéré comme facilement reconfigurable. Par exemple, le registre instruction du microprocesseur pourrait être supposé aussi facilement reconfigurable que tout autre registre simple. Néanmoins, ce registre est connecté extérieurement de façon irrégulière puisque chaque bit a un sens particulier en entrée du contrôleur. En conséquence, le registre instruction n'est pas reconfigurable sans une augmentation considérable de surface liée à la reconfiguration des interconnexions.

Lorsque seules une ou deux tranches diffèrent des autres, la reconfiguration est encore possible à un coût modéré, bien que plus important que dans le premier cas. Ainsi, il nous a été possible de rendre reconfigurables les générateurs de constantes et le registre servant à effectuer les opérations de décalage.

Un bloc "non reconfigurable" nécessite une augmentation de surface de l'ordre de 100% (au niveau de ce bloc). Parmi les blocs non reconfigurables, il est possible de citer le dispositif AS d'autorisation de signature qui, bien qu'intégré dans le chemin de données, n'a pas à proprement parler une structure en tranches. Un autre exemple est le registre d'état. Ce dernier possède en effet, dans son ensemble, une structure en tranches mais des circuits logiques sont de plus nécessaires sur les diverses tranches pour générer les différents indicateurs. La reconfiguration de l'ensemble a alors un coût dépassant très largement la limite fixée par le projet ESPRIT.

Le bilan de la reconfiguration du microprocesseur réalisé dans le cadre de ce projet est donc très fortement tributaire de la contrainte initiale. Dans ce cadre, nous rappelons que, sur la surface du coeur de la puce, 20% sont utilisés par des dispositifs de test et de reconfiguration et 20% sont composés d'éléments non reconfigurables.

En acceptant les formules de rendement présentées dans la thèse de M. Genestier, nous pouvons grossièrement estimer qu'une augmentation de rendement de 15 à 20% peut être attendue. Le "gain" semble en définitive bien modeste, comparé au travail nécessaire pour rendre un maximum d'éléments reconfigurables (coût de développement) et aux contraintes en fin de fabrication (localisation des défauts, coupure des fusibles laser, second test). Il faut souligner que ceci n'est vrai que dans le cadre strict de l'étude, où l'augmentation de surface a été bornée arbitrairement en début de projet. Pour cette raison, les calculs d'optimisation du rendement n'ont, dans notre cas, pu conduire qu'à des choix restreints.

Une suite possible à cette étude serait de supprimer la borne supérieure d'augmentation de la surface. Les calculs d'optimisation du rendement seraient alors tout différents. Ils pourraient considérer comme un paramètre la surface nécessaire à la reconfiguration. Ceci pourrait conduire à la définition de compromis beaucoup plus intéressants entre le rendement et la surface totale du circuit. Ainsi, il deviendrait possible, malgré le coût associé à cette démarche, de reconfigurer l'ensemble des éléments du chemin de données. Une reconfiguration à la fois structurelle et fonctionnelle serait alors nécessaire mais pourrait conduire globalement à une réduction de la surface non reconfigurable de la puce. Ceci pourrait se traduire par une augmentation globale du rendement de la puce. Une étude détaillée serait naturellement nécessaire pour établir l'intervalle des compromis possibles et l'augmentation de surface résultant.

L'étude du HYETI ne peut donc pas apporter des conclusions définitives sur l'intérêt de la reconfiguration. Il est clair qu'un microprocesseur "de catalogue" a fort peu de chances d'utiliser un jour de telles méthodes. Par contre, de très gros circuits spécifiques (embarqués par exemple)

contenant une mémoire de forte capacité, un contrôleur et un chemin de données pourront voir leur conception modifiée à partir des techniques étudiées ici afin d'améliorer leur rendement. Quant aux autres points forts abordés dans cette thèse (intégration de test en ligne, modifications de conception pour assurer la C-testabilité), ils soulèvent peu de réserve et pourront être utilisés dans de nombreuses applications.

Cette thèse aura donc enrichi de façon incontestable le savoir-faire en tolérance aux défauts de fin de fabrication, même si elle n'a pas exhibé une application industrialisable.

Annexes

[Annexe 1.1] : Jeu d'instructions

Cette annexe comporte, pour chaque instruction, la description de l'opération effectuée, les modes d'adressage pouvant être utilisés, le type de l'opérande OP et les bits du registre d'état positionnés (sauf si ce registre est destination).

légende :

R est l'un des 16 registres d'accès général,

Rint est l'un des 26 registres (registre d'accès général ou registre interne spécialisé, registre d'instruction mis à part),

OP est le deuxième opérande,

$\leftarrow$ représente l'affectation,

$\leftrightarrow$ représente la permutation des contenus de deux registres,

( ) est le contenu d'une adresse,

In indique le mode d'adressage Inherent,

Rel indique le mode d'adressage Relatif,

Im indique le mode d'adressage Immédiat,

Di indique le mode d'adressage Direct,

Reg indique le mode d'adressage Registre,

Inr indique le mode d'adressage Indirect Registre,

Inpr indique le mode d'adressage Indirect Registre Pré-modifié,

Inpo indique le mode d'adressage Indirect Registre Post-modifié,

0 indique que le bit concerné dans le registre d'état est remis à zéro,

1 indique que le bit concerné dans le registre d'état est remis à un,

X indique que le bit concerné dans le registre d'état est inchangé,

$\uparrow$ indique que le bit concerné dans le registre d'état est modifié,

+ indique que le mode d'adressage est autorisé pour l'opération.

1.1 INSTRUCTIONS ARITHMETIQUES (7)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpc	H	L	W	T	G	V	N	Z	I	C
ADD	ADDITION	R ← R + OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
ADDC	ADDITION AVEC RETENUE	R ← R + OP + C	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
SUB	SOUSTRACTION	R ← R - OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
SUBC	SOUSTRACTION AVEC RETENUE	R ← R - OP - C	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
INC	INCREMENTATION	R ← R + 1	RIEN	+								X	X	↑	↑	↑	↑	↑	↑	X	↑
DEC	DECREMENTATION	R ← R - 1	RIEN	+								X	X	↑	↑	↑	↑	↑	↑	X	↑
CLR	REMISE A ZERO D'UN REGISTRE	R ← 0	RIEN	+								X	X	X	X	X	X	X	X	X	X

1.2 INSTRUCTIONS LOGIQUES (12)

* : LA MODIFICATION DE CES BITS N'A PAS DE SIGNIFICATION

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpc	II	L	W*	T*	G*	V*	N	Z	I	C*
NOT	COMPLEMENTATION LOGIQUE A 1	R ← $\overline{R}$	RIEN	+								X	X	↑	↑	↑	↑	↑	↑	X	↑
NOTM	COMPLEMENTATION LOGIQUE A 1 (MASQUEE)	R ← $\overline{R}$ (MASQUE)	RIEN	+								↑	↑	↑	↑	↑	↑	↑	↑	X	↑
AND	ET LOGIQUE	R ← R AND OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
ANDM	ET LOGIQUE (MASQUEE)	R ← R AND OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑
OR	OU LOGIQUE	R ← R OR OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
ORM	OU LOGIQUE (MASQUEE)	R ← R OR OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑
NAND	ET NON LOGIQUE	R ← R NAND OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
NANDM	ET NON LOGIQUE (MASQUEE)	R ← R NAND OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑

1.2 INSTRUCTIONS LOGIQUES (SUITE 1 ET FIN)

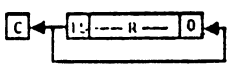
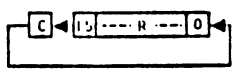
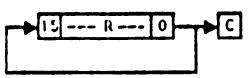
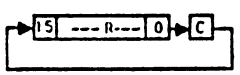
MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	H	L	W*	T*	G*	V*	N	Z	I	C*
NOR	OU NON LOGIQUE	R ← R NOR OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
NORM	OU NON LOGIQUE (MASQUE)	R ← R NOR OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑
EOR	OU-EXCLUSIF	R ← R EOR OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
EORM	OU-EXCLUSIF (MASQUE)	R ← R EOR OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑

1.3 INSTRUCTIONS DE DECALAGE ET DE ROTATION (10)

* : LA MODIFICATION DU BIT N'A PAS DE SIGNIFICATION

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Impr	Impd	H	L	W	T	G	V	N*	Z	I	C
LSL	DECALAGE LOGIQUE A GAUCHE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
ASL	DECALAGE ARITHMETIQUE A GAUCHE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
LSR	DECALAGE LOGIQUE A DROITE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
ASR	DECALAGE ARITHMETIQUE A DROITE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
LSLN	DECALAGE LOGIQUE A GAUCHE (N BITS)	N OPERATIONS LSL	LA VALEUR N			+	+	+	+	+	+	X	X	X	X	X	X	↑	I	X	↑
ASLN	DECALAGE ARITHMETIQUE A GAUCHE (N BITS)	N OPERATIONS ASL	LA VALEUR N			+	+	+	+	+	+	X	X	X	X	X	X	↑	I	X	↑
LSRN	DECALAGE LOGIQUE A DROITE (N BITS)	N OPERATIONS LSR	LA VALEUR N			+	+	+	+	+	+	X	X	X	X	X	X	↑	I	X	↑
ASRN	DECALAGE ARITHMETIQUE A DROITE (N BITS)	N OPERATIONS ASR	LA VALEUR N			+	+	+	+	+	+	X	X	X	X	X	X	↑	I	X	↑

1.3 INSTRUCTIONS DE DECALAGE ET DE ROTATION (SUITE 1 ET FIN)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	H	L	W	T	G	V	N*	Z	I	C
ROL	ROTATION A GAUCHE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
ROLC	ROTATION A GAUCHE AVEC RETENUE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
ROR	ROTATION A DROITE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑
RORC	ROTATION A DROITE AVEC RETENUE (1 BIT)		RIEN	+								X	X	X	X	X	X	X	X	X	↑

1.4 INSTRUCTIONS DE COMPARAISON (2)

* : LA MODIFICATION DE CES BITS PAR COMPM N'A PAS DE SIGNIFICATION

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpo	H	L	W*	T*	G*	Y*	N	Z	I	C*
COMP	COMPARAISON	R - OP	DONNEE			+	+	+	+	+	+	X	X	↑	↑	↑	↑	↑	↑	X	↑
COMPM	COMPARAISON MASQUEE	R EOR OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	↑	↑	↑	↑	↑	↑	X	↑

1.5 INSTRUCTIONS DE TRANSFERT (5)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	II	L	W	T	G	V	N	Z	I	C
EXCH	PERMUTATION DU CONTENU DE DEUX REGISTRES	R ↔ OP	REGISTRE ACCES GENERAL					+				X	X	X	X	X	X	↑	↑	X	X
LOAD	CHARGEMENT D'UN REGISTRE	R ← OP	DONNEE			+	+	+	+	+	+	X	X	X	X	X	X	↑	↑	X	X
LOADM	CHARGEMENT MASQUE D'UN REGISTRE	R ← OP (MASQUE)	DONNEE			+	+	+	+	+	+	↑	↑	X	X	X	X	↑	↑	X	X
STORE	RANGEMENT D'UN REGISTRE DANS LA MEMOIRE	R → MEMOIRE	ADRESSE DE LA MEMOIRE EXTERNE			+	+	+	+	+	+	X	X	X	X	X	X	↑	↑	X	X
STOREM	RANGEMENT MASQUE D'UN REGISTRE DANS LA MEMOIRE	R → MEMOIRE (MASQUE)	ADRESSE DE LA MEMOIRE EXTERNE			+	+	+	+	+	+	↑	↑	X	X	X	X	↑	↑	X	X

1.6 INSTRUCTIONS AVEC UN OPERANDE IMPLICITE (5)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Dl	Reg	Inr	Inpr	Inpd	H	L	W	T	G	V	N	Z	I	C
CLC	REMISE A ZERO DU BIT DE RETENUE (C)	C ← 0	RIEN	+								X	X	X	X	X	X	X	X	X	0
SEC	REMISE A UN DU BIT DE RETENUE (C)	C ← 1	RIEN	+								X	X	X	X	X	X	X	X	X	1
CLI	REMISE A ZERO DU MASQUE D'INTERRUPTION (I)	I ← 0	RIEN	+								X	X	X	X	X	X	X	X	0	X
SEI	REMISE A UN DU MASQUE D'INTERRUPTION (I)	I ← 1	RIEN	+								X	X	X	X	X	X	X	X	1	X
NOP	AUCUNE OPERATION	RIEN	RIEN	+								X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (42)

* : INSTRUCTIONS UTILISEES POUR TESTER DES NOMBRES NON SIGNES (DE 0000 A FFFF)!!
** : INSTRUCTIONS UTILISEES POUR TESTER DES NOMBRES SIGNES EN COMPLEMENT A DEUX
+ : G = C OR Z
++ : T = N EOR V
+++ : W = Z OR (N EOR V)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpo	H	L	W	T	G	V	N	Z	I	C
BCC *	BRANCHEMENT SI LA RETENUE VAUT 0	CO ← CO + OP SI C = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BCS *	BRANCHEMENT SI LA RETENUE VAUT 1	CO ← CO + OP SI C = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BEQ	BRANCHEMENT SI LE RESULTAT EST NUL	CO ← CO + OP SI Z = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BNE	BRANCHEMENT SI LE RESULTAT N'EST PAS NUL	CO ← CO + OP SI Z = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BVC	BRANCHEMENT SI PAS DE DEPASSEMENT	CO ← CO + OP SI V = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BVS	BRANCHEMENT SI DEPASSEMENT	CO ← CO + OP SI V = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BMI	BRANCHEMENT SI LE RESULTAT EST NEGATIF	CO ← CO + OP SI N = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BPL	BRANCHEMENT SI LE RESULTAT EST POSITIF	CO ← CO + OP SI N = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (SUITE 1)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	DI	Reg	Inr	Inpr	Inpo	H	L	W	T	G	Y	N	Z	I	C
BHI *	BRANCHEMENT SI PLUS GRAND (R > OP)	CO ← CO + OP SI G = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLS *	BRANCHEMENT SI PLUS PETIT OU EGAL (R ≤ OP)	CO ← CO + OP SI G = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BGE **	BRANCHEMENT SI SUPERIEUR OU EGAL (R ≥ OP)	CO ← CO + OP SI T = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLT **	BRANCHEMENT SI STRICTEMENT INFERIEUR (R < OP)	CO ← CO + OP SI T = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BGT **	BRANCHEMENT SI STRICTEMENT SUPERIEUR (R > OP)	CO ← CO + OP SI W = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLE **	BRANCHEMENT SI INFERIEUR OU EGAL (R ≤ OP)	CO ← CO + OP SI W = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLH	BRANCHEMENT SI L VAUT 1	CO ← CO + OP SI L = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLL	BRANCHEMENT SI L VAUT 0	CO ← CO + OP SI L = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (SUITE 2)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	DI	Reg	Inr	Inpr	Inpc	H	L	W	T	G	V	N	Z	I	C
BHH	BRANCHEMENT SI H VAUT 1	CO ← CO + OP SI H = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BHL	BRANCHEMENT SI H VAUT 0	CO ← CO + OP SI H = 0 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BIH	BRANCHEMENT SI I VAUT 1	CO ← CO + OP SI I = 1 SINON CO ← CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BIL	BRANCHEMENT SI I VAUT 0	CO ← CO + OP SI I = 0 SINON CO ←CO + 1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BRA	BRANCHEMENT INCONDITIONNEL	CO ← CO + OP	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BSR	BRANCHEMENT INCONDITIONNEL VERS UN SOUS-PROGRAMME	(PP)← CO PUIS CO ← CO + OP	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BCCSR	IDENTIQUE A BCC MAIS VERS UN SOUS-PROGRAMME	(PP)← CO PUIS CO ← CO + OP SI C=0 CO ← CO + 1 SINON	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BCSSR	IDENTIQUE A BCS MAIS VERS UN SOUS-PROGRAMME	(PP)← CO PUIS CO ← CO + OP SI C=1 CO ← CO + 1 SINON	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (SUITE 3)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpo	H	L	W	T	G	V	N	Z	I	C
BEQSR	IDENTIQUE A BEQ MAIS VERS UN SOUS-PROGRAMME	(PP)← CO PUIS CO ← CO + OP CO ← CO + 1 SI Z=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BNESR	IDENTIQUE A BNE MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI Z=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BVCSR	IDENTIQUE A BVC MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI V=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BVSSR	IDENTIQUE A BVS MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI V=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BMISR	IDENTIQUE A BMI MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI N=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BPLSR	IDENTIQUE A BPL MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI N=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BHISR	IDENTIQUE A BHI MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI G=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLSSR	IDENTIQUE A BLS MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ←CO + OP CO ←CO + 1 SI G=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (SUITE 4)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	II	L	W	T	G	Y	N	Z	I	C
BGESR	IDENTIQUE A BGE MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI T=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLTSR	IDENTIQUE A BLT MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI T=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BGTSR	IDENTIQUE A BGT MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI W=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLESR	IDENTIQUE A BLE MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI W=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLHSR	IDENTIQUE A BLH MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI L=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BLLSR	IDENTIQUE A BLL MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI L=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BHHSR	IDENTIQUE A BHH MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI H=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BHLSR	IDENTIQUE A BHL MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI H=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

1.7 INSTRUCTIONS DE BRANCHEMENT (SUITE 5 ET FIN)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	H	L	W	T	G	V	N	Z	I	C
BIHSR	IDENTIQUE A BIH MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI I=1	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X
BILSR	IDENTIQUE A BIH MAIS VERS UN SOUS-PROGRAMME	(PP) ← CO PUIS CO ← CO + OP CO ← CO + 1 SI I=0	DEPLACEMENT		+							X	X	X	X	X	X	X	X	X	X

I.8 INSTRUCTIONS DE SAUT OU DE MANIPULATION DE PILE (7)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpi	Inpo	II	L	W	T	G	V	N	Z	I	C
JMP	SAUT INCONDITIONNEL	CO ← OP	ADRESSE DU SAUT			+	+	+	+	+	+	X	X	X	X	X	X	X	X	X	X
JSR	SAUT INCONDITIONNEL VERS UN SOUS-PROGRAMME	(PP) ← CO PP ← PP - 1 PUIS CO ← OP	ADRESSE DU SAUT			+	+	+	+	+	+	X	X	X	X	X	X	X	X	X	X
RSR	FIN DE SOUS-PROGRAMME ET RETOUR AU PROGRAMME PRINCIPAL	CO ← (PP) ET PP ← PP + 1	RIEN	+								X	X	X	X	X	X	X	X	X	X
PUSH	EMPILER UN OPERANDE SUR LE SOMMET DE LA PILE	(PP) ← OP ET PP ← PP - 1	DONNEE			+	+	+	+	+	+	X	X	X	X	X	X	X	X	X	X
PULL	DEPILER UN OPERANDE DU SOMMET DE LA PILE	R ← (PP) ET PP ← PP + 1	RIEN	+								X	X	X	X	X	X	X	X	X	X
LINK	REALISATION D'UN LIEN ; RESERVATION D'UNE ZONE DE LA PILE	(PP) ← R, PP ← PP - 1 PUIS R ← PP PUIS PP ← PP - OP	DEPLACEMENT POSITIF JUSQU'A /32767/			+	+	+	+	+	+	X	X	X	X	X	X	X	X	X	X
UNLINK	ANNULATION D'UN LIEN CREE PAR LINK	PP ← R PUIS R ← (PP), PP ← PP + 1	RIEN	+								X	X	X	X	X	X	X	X	X	X

1.9 INSTRUCTIONS D'INTERRUPTION (3)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpd	H	L	W	T	G	V	N	Z	I	C
SWI	PROCEDURE D'EXCEPTION (INTERRUPTION LOGICIELLE)	(PP)← CO PUIS (PP)← RE PUIS (PP)← RPPR, PP← PP - 3 PUIS RPPR← RPEX PUIS CO← 8	RIEN	+								X	X	X	X	X	X	X	X	X	X
WAIT	ATTENTE D'UNE INTERRUPTION MATERIELLE	(PP) ←CO PUIS (PP)← RE PUIS (PP)←RPPR PP←PP - 3 PUIS PUIS RPPR←RPEX PUIS ATTENTE NMI OU IRQ OU RESET	RIEN	+								X	X	X	X	X	X	X	X	X	X
RTI	FIN D'UNE PROCEDURE D'EXCEPTION ET RETOUR AU PROGRAMME PRINCIPAL	RPPR← (PP) PUIS RE← (PP) PUIS CO← (PP), PP← PP + 3	RIEN	+								X	X	X	X	X	X	X	X	X	X

1.10 INSTRUCTIONS DE TEST (5)

MNEMONIQUE	INTERPRETATION	OPERATION EFFECTUEE	TYPE DE L'OPERANDE	MODE D'ADRESSAGE								REGISTRE D'ETAT (RE)									
				In	Rel	Im	Di	Reg	Inr	Inpr	Inpx	II	L	W	T	G	V	N	Z	I	C
RAZ	REMISE A ZERO DU REGISTRE DE SIGNATURE	SIGN ← 0	RIEN	+								X	X	X	X	X	X	X	X	X	X
LRS	CHARGEMENT D'UN REGISTRE INTERNE	Rint ← OP	DONNEE			+	+					X	X	X	X	X	X	↑	↑	X	X
SRS	RANGEMENT D'UN REGISTRE INTERNE	MEMOIRE ← Rint	ADRESSE DE LA MEMOIRE			+	+					X	X	X	X	X	X	↑	↑	X	X
AJS	AJUSTEMENT DU CONTENU DU REGISTRE DE SIGNATURE A UNE VALEUR VOULUE	SIGN ← SIGN EOR OP	DONNEE			+	+	+	+	+	+	X	X	X	X	X	X	↑	↑	X	X
AJST	AJUSTEMENT DU CONTENU DU REGISTRE DE SIGNATURE A UNE VALEUR VOULUE PUIS TEST DU RESULTAT	SIGN ← SIGN EOR OP PUIS ALARME = 1 SI LE RESULTAT EST DIFFERENT DE 0	DONNEE			+	+	+	+	+	+	X	X	X	X	X	X	↑	↑	X	X

[Annexe 1.2] : Format et codage des instructions

1.2.1 Format des instructions

Les bits 13 à 15 dans le premier mot de l'instruction, dit code d'instruction, définissent le mode d'adressage utilisé par l'instruction, selon la correspondance indiquée dans le tableau 1.

BIT DU CODE INSTRUCTION			MODE D'ADRESSAGE
15	14	13	
0	0	0	INHERENT
0	0	1	RELATIF
0	1	0	IMMEDIAT
0	1	1	DIRECT
1	0	0	REGISTRE
1	0	1	INDIRECT REGISTRE
1	1	0	INDIRECT REGISTRE PRE-MODIFIE
1	1	1	INDIRECT REGISTRE POST-MODIFIE

Tableau 1 : code du mode d'adressage.

Les bits de 0 à 12 ont des significations différentes selon le mode d'adressage.

Notations

Reg. Dest. : registre destination.

Reg. OP. : registre d'opérande (source).

Reg. Ind. : registre d'indirection.

Code-OP : code de l'opération.

X signifie l'indétermination (0 ou 1).

Le tableau 2 récapitule les codes des différentes ressources de la machine utilisables comme source ou destination.

Les 16 registres correspondant aux 16 premiers codes sont des registres d'accès général.

1.2.2. Codage des instructions utilisant le mode d'adressage inhérent

Le code de l'opération à effectuer (Code-OP) est donné dans les bits 8 à 12 de l'instruction. Le format de codage des différentes instructions utilisant le mode inhérent est donné dans le tableau 3.

CODE DECIMAL	CODE BINAIRE 4 3 2 1 0	RESSOURCE	CODE DECIMAL	CODE BINAIRE 4 3 2 1 0	RESSOURCE
0	0 0 0 0 0	**R 0**	16	1 0 0 0 0	**CO**
1	0 0 0 0 1	**R1**	17	1 0 0 0 1	**RE**
2	0 0 0 1 0	**R2**	18	1 0 0 1 0	**S**
3	0 0 0 1 1	**R3**	19	1 0 0 1 1	**TM**
4	0 0 1 0 0	**R4**	20	1 0 1 0 0	**E1**
5	0 0 1 0 1	**R5**	21	1 0 1 0 1	**E2**
6	0 0 1 1 0	**R6**	22	1 0 1 1 0	**T1**
7	0 0 1 1 1	**R7**	23	1 0 1 1 1	**POL**
8	0 1 0 0 0	**R8**	24	1 1 0 0 0	**RTIS**
9	0 1 0 0 1	**RPA**	25	1 1 0 0 1	**SIGN**
10	0 1 0 1 0	**RMA**	26	1 1 0 1 0	**RI**
11	0 1 0 1 1	**PP**	27	1 1 0 1 1	
12	0 1 1 0 0	**RPPR**	28	1 1 1 0 0	
13	0 1 1 0 1	**RPDO**	29	1 1 1 0 1	
14	0 1 1 1 0	**RPPI**	30	1 1 1 1 0	
15	0 1 1 1 1	**RPEX**	31	1 1 1 1 1	

Tableau 2 : codage des ressources.

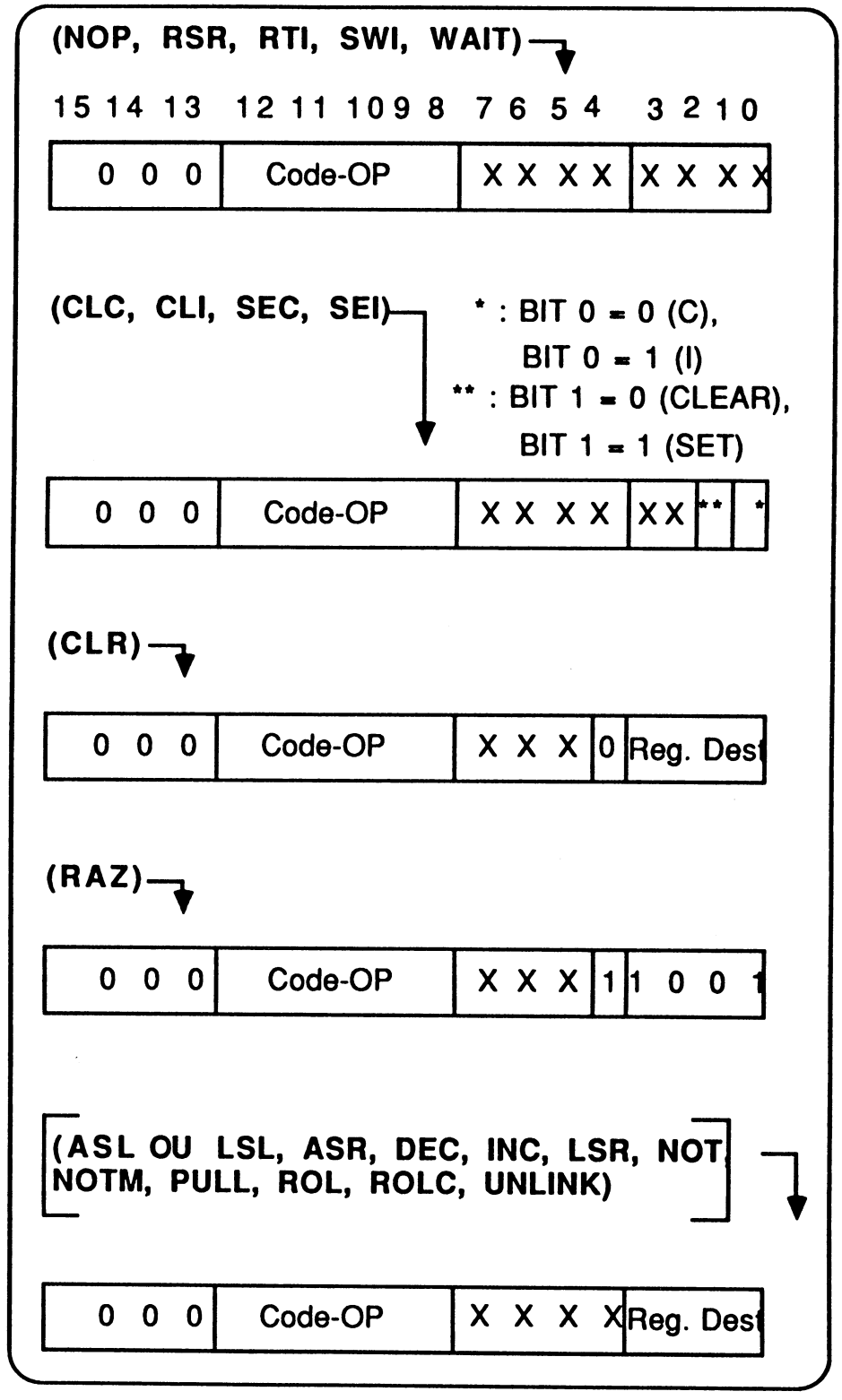


Tableau 3 : codage des différentes instructions utilisant le mode inhérent.

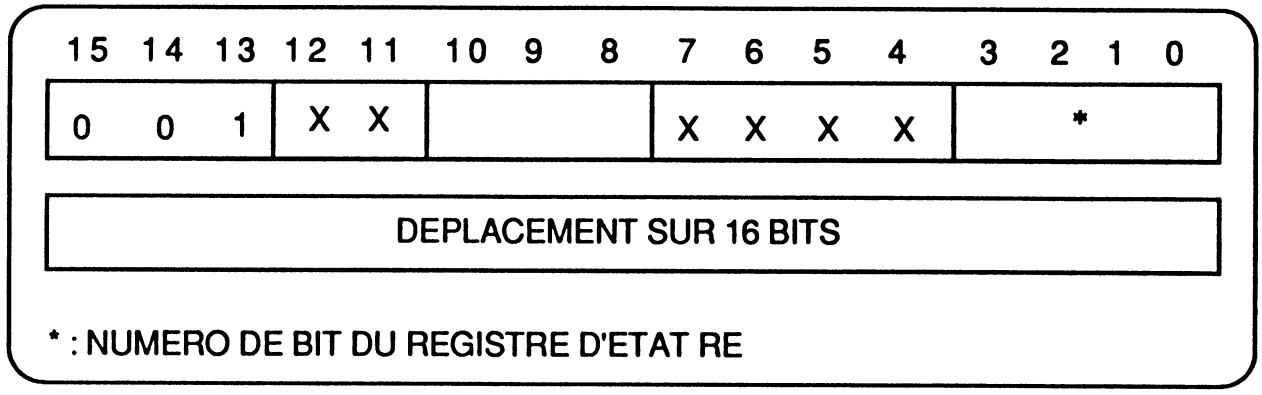
Le tableau 4 donne les opérations et leur Code-OP en mode inhérent.

CODE DECIMAL	Code-OP 12 11 10 9 8	OPERATION	CODE DECIMAL	Code-OP 12 11 10 9 8	OPERATION
0	0 0 0 0 0	NOP	16	1 0 0 0 0	ROL
1	0 0 0 0 1	CLC, SEC, CLI, SEI	17	1 0 0 0 1	ROR
2	0 0 0 1 0	CLR	18	1 0 0 1 0	ROLC
3	0 0 0 1 1	RAZ	19	1 0 0 1 1	non utilisé
4	0 0 1 0 0	RSR	20	1 0 1 0 0	non utilisé
5	0 0 1 0 1	RTI	21	1 0 1 0 1	non utilisé
6	0 0 1 1 0	WAIT	22	1 0 1 1 0	non utilisé
7	0 0 1 1 1	SWI	23	1 0 1 1 1	non utilisé
8	0 1 0 0 0	PULL	24	1 1 0 0 0	non utilisé
9	0 1 0 0 1	NOT	25	1 1 0 0 1	NOTM
10	0 1 0 1 0	INC	26	1 1 0 1 0	non utilisé
11	0 1 0 1 1	DEC	27	1 1 0 1 1	non utilisé
12	0 1 1 0 0	LSR	28	1 1 1 0 0	non utilisé
13	0 1 1 0 1	LSL = ASL	29	1 1 1 0 1	non utilisé
14	0 1 1 1 0	ASR	30	1 1 1 1 0	non utilisé
15	0 1 1 1 1	UNLINK	31	1 1 1 1 1	non utilisé

Tableau 4 : opération et codes opération en mode inhérent.

1.2.3 Codage des instructions pour les autres modes d'adressage

En ce qui concerne le mode d'adressage relatif, deux mots sont indispensables afin de coder l'instruction.



Le premier mot contient :

- le code du mode d'adressage (bit 13 à bit 15),
- le bit 8 qui détermine si le branchement est conditionnel (bit 8 = 1) ou inconditionnel (bit 8 = 0),
- le bit 9 qui distingue le branchement simple (bit 9 = 0) du branchement vers un sous-programme (bit 9 = 1),
- le bit 10 qui détermine si le branchement s'effectue lorsque le bit testé est à 1 (bit 10 = 1) ou à 0 (bit 10 = 0),
- un champ (bits 0 à 3) définissant le bit de RE à tester dans le cas du branchement conditionnel. Le tableau 5 donne la correspondance entre le bit de RE sélectionné et son numéro.

Le deuxième mot est un déplacement sur 16 bits.

CODE DECIMAL	BITS 3 2 1 0	BIT CONCERNE DE RE	CODE DECIMAL	BITS 3 2 1 0	BIT CONCERNE DE RE
0	0 0 0 0	**C**	8	1 0 0 0	**L**
1	0 0 0 1	**I**	9	1 0 0 1	**H**
2	0 0 1 0	**Z**	10	1 0 1 0	**AUCUN**
3	0 0 1 1	**N**	11	1 0 1 1	**AUCUN**
4	0 1 0 0	**V**	12	1 1 0 0	**AUCUN**
5	0 1 0 1	**G**	13	1 1 0 1	**AUCUN**
6	0 1 1 0	**T**	14	1 1 1 0	**AUCUN**
7	0 1 1 1	**W**	15	1 1 1 1	**AUCUN**

Tableau 4 : correspondance entre le bit de RE sélectionné et son numéro (bits 0 à 3).

Pour les 6 modes d'adressage restant, le Code-OP est donné dans les bits 8 à 12, comme dans le cas du mode inhérent. Le tableau 6 récapitule le Code-OP des opérations utilisant ces modes d'adressage.

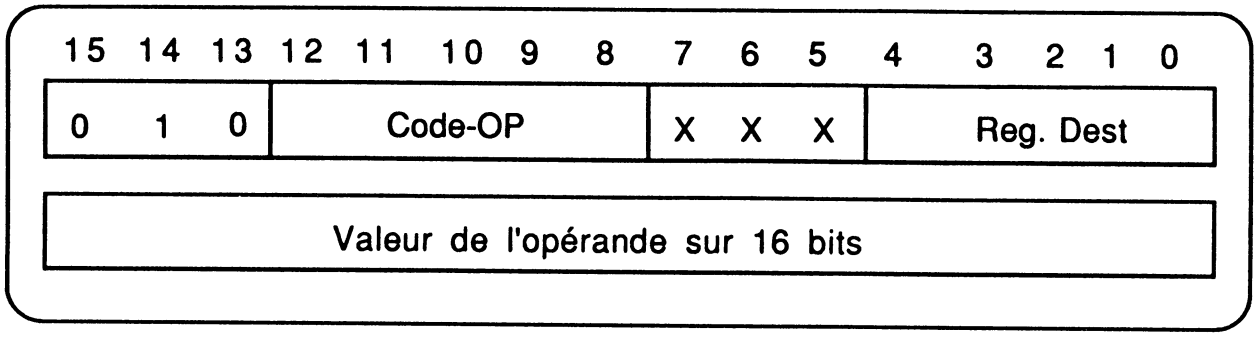
Il faut noter que les modes d'adressage immédiat et direct exigent deux mots pour une instruction.

CODE DECIMAL	Code-OP 12 11 10 9 8	OPERATION	CODE DECIMAL	Code-OP 12 11 10 9 8	OPERATION
0	0 0 0 0 0	ADD	16	1 0 0 0 0	JMP
1	0 0 0 0 1	SUB	17	1 0 0 0 1	JSR
2	0 0 0 1 0	ADDC	18	1 0 0 1 0	PUSH
3	0 0 0 1 1	SUBC	19	1 0 0 1 1	LINK
4	0 0 1 0 0	AND	20	1 0 1 0 0	ANDM
5	0 0 1 0 1	OR	21	1 0 1 0 1	ORM
6	0 0 1 1 0	NAND	22	1 0 1 1 0	NANDM
7	0 0 1 1 1	NOR	23	1 0 1 1 1	NORM
8	0 1 0 0 0	EOR	24	1 1 0 0 0	EORM
9	0 1 0 0 1	COMP	25	1 1 0 0 1	COMPM
10	0 1 0 1 0	LOAD	26	1 1 0 1 0	LOADM
11	0 1 0 1 1	STORE	27	1 1 0 1 1	STOREM
12	0 1 1 0 0	LSRN	28	1 1 1 0 0	SRS
13	0 1 1 0 1	LSLN = ASLN	29	1 1 1 0 1	LRS
14	0 1 1 1 0	ASRN	30	1 1 1 1 0	AJS
15	0 1 1 1 1	EXCH	31	1 1 1 1 1	AJST

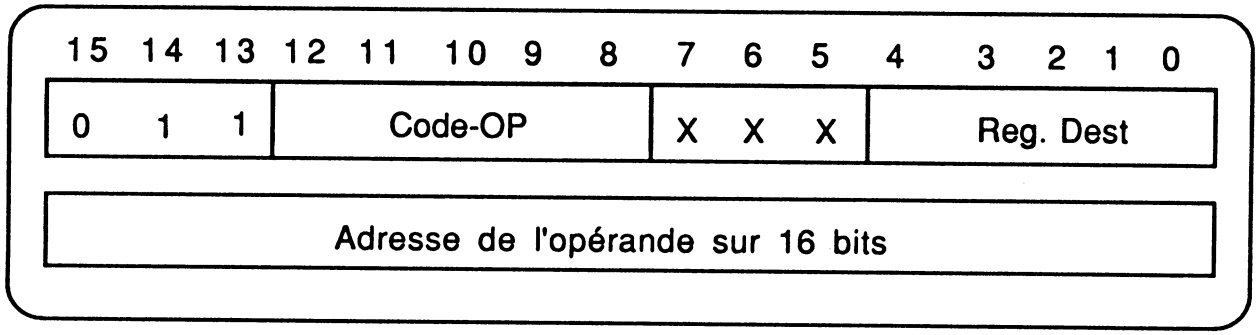
Tableau 6 : code-OP pour les 6 modes d'adressage restant.

Les codes d'instructions SRS et LRS sont :

- mode immédiat : 2 mots.

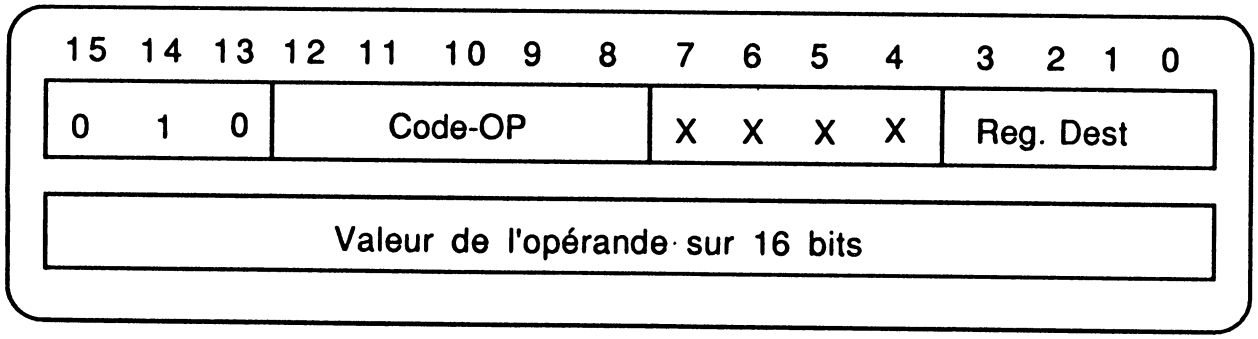


- mode direct : 2 mots.

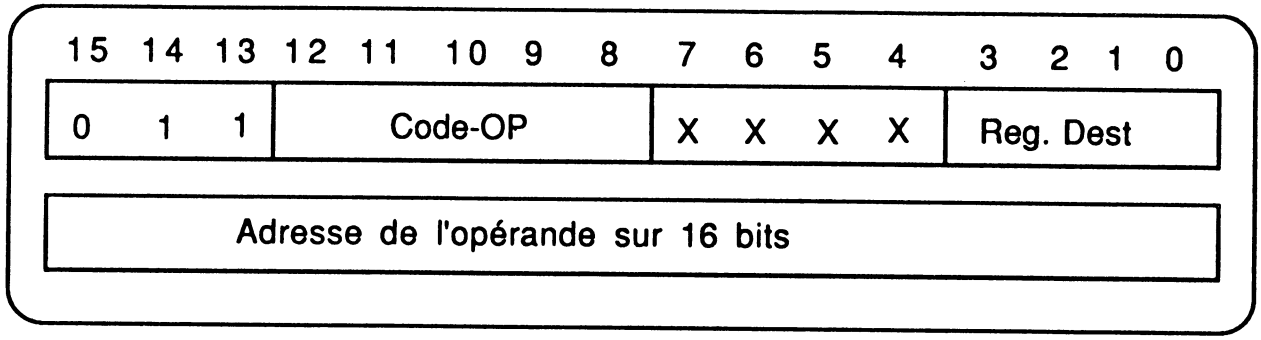


En ce qui concerne les autres instructions, les formats sont :

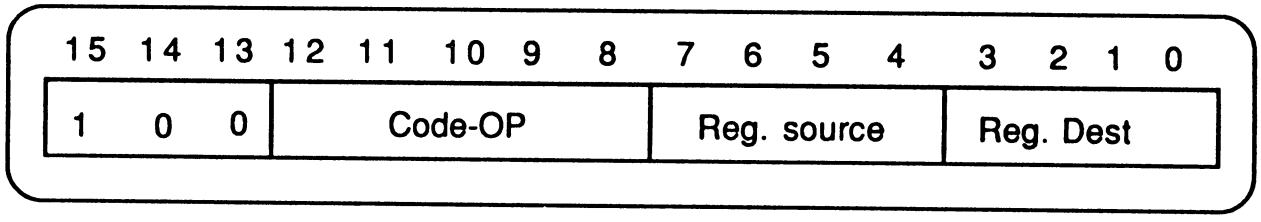
- mode immédiat : 2mots.



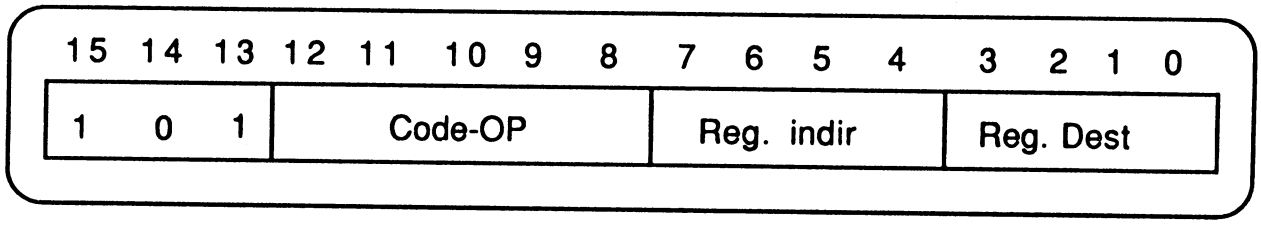
- mode direct : 2 mots.



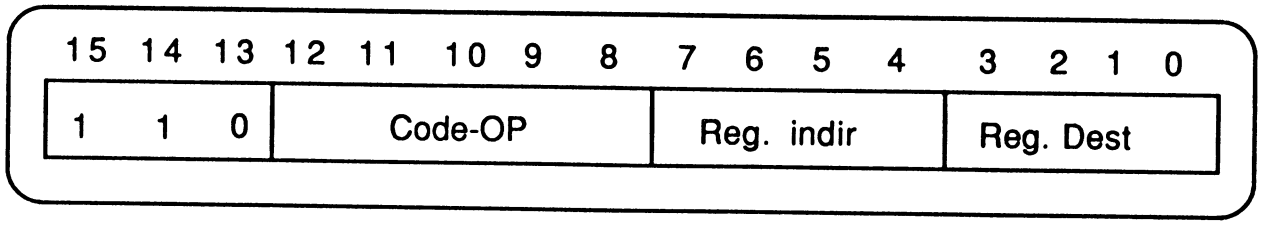
- mode registre : 1 mot.



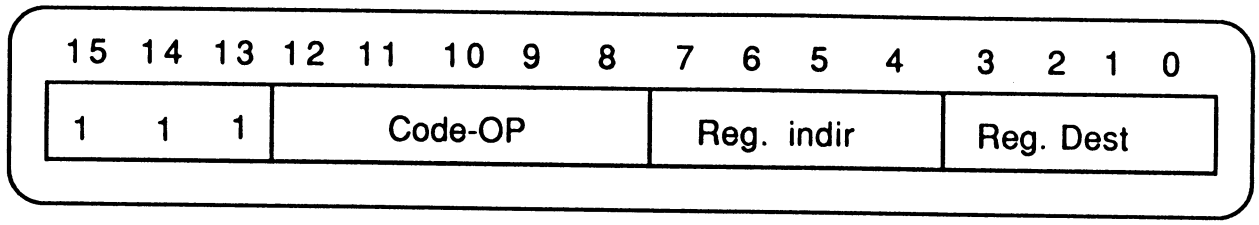
- mode indirect registre : 1 mot



- mode indirect registre pré-modifié : 1 mot



- mode indirect registre post-modifié : 1 mot



[Annexe 3.1] : vecteurs de test (UAL initiale).

Opération : ADD,

Commandes : CM1 = 1, CM2 = 0, CM3 = 0, CM4 = 1, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	1A5A5 H	0C6C6 H	06C6C H	0
2	0	0D2D2 H	16363 H	03636 H	0
3	0	16969 H	1B1B1 H	11B1B H	0
4	0	0B4B4 H	0D8D8 H	18D8D H	1
5	1	05A5A H	06C6C H	0C6C6 H	1
6	1	12D2D H	03636 H	16363 H	1
7	1	09696 H	11B1B H	1B1B1 H	1
8	1	14B4B H	18D8D H	0D8D8 H	0

Opération : SUB,

Commandes : CM1 = 0, CM2 = 1, CM3 = 1, CM4 = 0, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	0C6C6 H	06C6C H	05A5A H	0
2	0	16363 H	03636 H	12D2D H	0
3	0	1B1B1 H	11B1B H	09696 H	0
4	0	0D8D8 H	18D8D H	14B4B H	1
5	1	06C6C H	0C6C6 H	1A5A5 H	1
6	1	03636 H	16363 H	0D2D2 H	1
7	1	11B1B H	1B1B1 H	16969 H	1
8	1	18D8D H	0D8D8 H	0B4B4 H	0

Opération : INC,

Commandes : CM1 = 0, CM2 = 0, CM3 = 0, CM4 = 0, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	0FFFE H	0FFFE H	0FFFF H	1
2	0	1FFFD H	0FFFC H	1FFFE H	1
3	0	0FFFE H	1FFFF H	0FFFF H	1
4	0	1FFFB H	1FFF9 H	1FFFC H	1
5	1	0000 H	0000 H	0000 H	1
6	1	0FFFF H	0000 H	0FFFF H	1
7	1	00000 H	0FFFF H	00000 H	1
8	1	0FFFF H	0FFFF H	0FFFF H	1
9	0	0FFFD H	0FFFE H	0FFFE H	1
10	0	0FFF7 H	0FFF3 H	0FFF8 H	1
11	0	0FFFB H	0FFFC H	0FFFC H	1
12	0	0FFEF H	0FFE7 H	0FFF0 H	1
13	0	0FFF7 H	0FFF8 H	0FFF8 H	1
14	0	0FFDF H	0FFCF H	0FFE0 H	1
15	0	0FFEF H	0FFF0 H	0FFF0 H	1
16	0	0FFBF H	0FF9F H	0FFC0 H	1
17	0	0FFDF H	0FFE0 H	0FFE0 H	1
18	0	0FF7F H	0FF3F H	0FF80 H	1
19	0	0FFBF H	0FFC0 H	0FFC0 H	1
20	0	0FEFF H	0FE7F H	0FF00 H	1

Opération : INC (suite et fin),

Commandes : CM1 = 0, CM2 = 0, CM3 = 0, CM4 = 0, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
21	0	0FF7F H	0FF80 H	0FF80 H	1
22	0	0FDFF H	0FCFF H	0FE00 H	1
23	0	0FEFF H	0FF00 H	0FF00 H	1
24	0	0FBFF H	0F9FF H	0FC00 H	1
25	0	0FDFF H	0FE00 H	0FE00 H	1
26	0	0F7FF H	0F3FF H	0F800 H	1
27	0	0FBFF H	0FC00 H	0FC00 H	1
28	0	0EFFF H	0E7FF H	0F000 H	1
29	0	0F7FF H	0F800 H	0F800 H	1
30	0	0DFFF H	0CFFF H	0E000 H	1
31	0	0EFFF H	0F000 H	0F000 H	1
32	0	0BFFF H	09FFF H	0C000 H	1
33	0	0DFFF H	0E000 H	0E000 H	1
34	0	07FFF H	03FFF H	08000 H	1
35	0	0BFFF H	0C000 H	0C000 H	1
36	0	0FFFF H	07FFF H	10000 H	1
37	0	07FFF H	08000 H	08000 H	1
38	0	0EFFF H	1FFFF H	10000 H	1
39	0	1FFFF H	07FFF H	00000 H	1
40	0	1FFFF H	1FFFF H	00000 H	1

Opération : DEC,

Commandes : CM1 = 1, CM2 = 1, CM3 = 1, CM4 = 1, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	00000 H	0
2	0	0FFFF H	10000 H	0FFFF H	0
3	0	10000 H	0FFFF H	10000 H	0
4	0	1FFFF H	1FFFF H	1FFFF H	0
5	1	00000 H	00000 H	1FFFF H	1
6	1	00001 H	00000 H	00000 H	0
7	1	00002 H	00001 H	00001 H	0
8	1	00001 H	00001 H	00000 H	0
9	1	00004 H	00002 H	00003 H	0
10	1	00002 H	00002 H	00001 H	0
11	1	00008 H	00004 H	00007 H	0
12	1	00004 H	00004 H	00003 H	0
13	1	00010 H	00008 H	0000F H	0
14	1	00008 H	00008 H	00007 H	0
15	1	00020 H	00010 H	0001F H	0
16	1	00010 H	00010 H	0000F H	0
17	1	00040 H	00020 H	0003F H	0
18	1	00020 H	00020 H	0001F H	0
19	1	00080 H	00040 H	0007F H	0
20	1	00040 H	00040 H	0003F H	0

Opération : DEC (suite et fin),

Commandes : CM1 = 1, CM2 = 1, CM3 = 1, CM4 = 1, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
21	1	00100 H	00080 H	000FF H	0
22	1	00080 H	00080 H	0007F H	0
23	1	00200 H	00100 H	001FF H	0
24	1	00100 H	00100 H	000FF H	0
25	1	00400 H	00200 H	003FF H	0
26	1	00200 H	00200 H	001FF H	0
27	1	00800 H	00400 H	007FF H	0
28	1	00400 H	00400 H	003FF H	0
29	1	01000 H	00800 H	00FFF H	0
30	1	00800 H	00800 H	007FF H	0
31	1	02000 H	01000 H	01FFF H	0
32	1	01000 H	01000 H	00FFF H	0
33	1	04000 H	02000 H	03FFF H	0
34	1	02000 H	02000 H	01FFF H	0
35	1	08000 H	04000 H	07FFF H	0
36	1	04000 H	04000 H	03FFF H	0
37	1	00000 H	18000 H	1FFFF H	1
38	1	08000 H	08000 H	07FFF H	0
39	1	10000 H	00000 H	0FFFF H	0
40	1	10000 H	10000 H	0FFFF H	0

Opération : NOT,

Commandes : CM1 = 0, CM2 = 0, CM3 = 0, CM4 = 0, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	1FFFF H	0
2	0	1FFFF H	00000 H	00000 H	0
3	0	00000 H	1FFFF H	1FFFF H	0
4	0	1FFFF H	1FFFF H	00000 H	0
5	1	00000 H	00000 H	1FFFF H	1
6	1	00000 H	1FFFF H	1FFFF H	1
7	1	1FFFF H	00000 H	1FFFF H	1
8	1	1FFFF H	1FFFF H	00000 H	1

Opération : AND,

Commandes : CM1 = 1, CM2 = 0, CM3 = 1, CM4 = 1, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	00000 H	0
2	0	1FFFF H	00000 H	00000 H	0
3	0	00000 H	1FFFF H	00000 H	0
4	0	1FFFF H	1FFFF H	1FFFF H	0
5	1	00000 H	00000 H	00000 H	1
6	1	00000 H	1FFFF H	00000 H	1
7	1	1FFFF H	00000 H	00000 H	1
8	1	1FFFF H	1FFFF H	1FFFF H	1

Opération : OR,

Commandes : CM1 = 1, CM2 = 1, CM3 = 1, CM4 = 0, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	00000 H	0
2	0	1FFFF H	00000 H	1FFFF H	0
3	0	00000 H	1FFFF H	1FFFF H	0
4	0	1FFFF H	1FFFF H	1FFFF H	0
5	1	00000 H	00000 H	00000 H	1
6	1	00000 H	1FFFF H	1FFFF H	1
7	1	1FFFF H	00000 H	1FFFF H	1
8	1	1FFFF H	1FFFF H	1FFFF H	1

Opération : NAND,

Commandes : CM1 = 0, CM2 = 1, CM3 = 0, CM4 = 0, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	1FFFF H	0
2	0	1FFFF H	00000 H	1FFFF H	0
3	0	00000 H	1FFFF H	1FFFF H	0
4	0	1FFFF H	1FFFF H	00000 H	0
5	1	00000 H	00000 H	1FFFF H	1
6	1	00000 H	1FFFF H	1FFFF H	1
7	1	1FFFF H	00000 H	1FFFF H	1
8	1	1FFFF H	1FFFF H	00000 H	1

Opération : NOR,

Commandes : CM1 = 0, CM2 = 0, CM3 = 0, CM4 = 1, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	1FFFF H	0
2	0	1FFFF H	00000 H	00000 H	0
3	0	00000 H	1FFFF H	00000 H	0
4	0	1FFFF H	1FFFF H	00000 H	0
5	1	00000 H	00000 H	1FFFF H	1
6	1	00000 H	1FFFF H	00000 H	1
7	1	1FFFF H	00000 H	00000 H	1
8	1	1FFFF H	1FFFF H	00000 H	1

Opération : EOR,

Commandes : CM1 = 0, CM2 = 1, CM3 = 1, CM4 = 0, CM5 = 1.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	0	00000 H	00000 H	00000 H	0
2	0	1FFFF H	00000 H	1FFFF H	0
3	0	00000 H	1FFFF H	1FFFF H	0
4	0	1FFFF H	1FFFF H	00000 H	0
5	1	00000 H	00000 H	00000 H	1
6	1	00000 H	1FFFF H	1FFFF H	1
7	1	1FFFF H	00000 H	1FFFF H	1
8	1	1FFFF H	1FFFF H	00000 H	1

[Annexe 3.2] : vecteurs de test des blocs de saut (UALCarry-Skip).

Opération : ADD

Commandes : CM1 = 1, CM2 = 0, CM3 = 0, CM4 = 1, CM5 = 0.

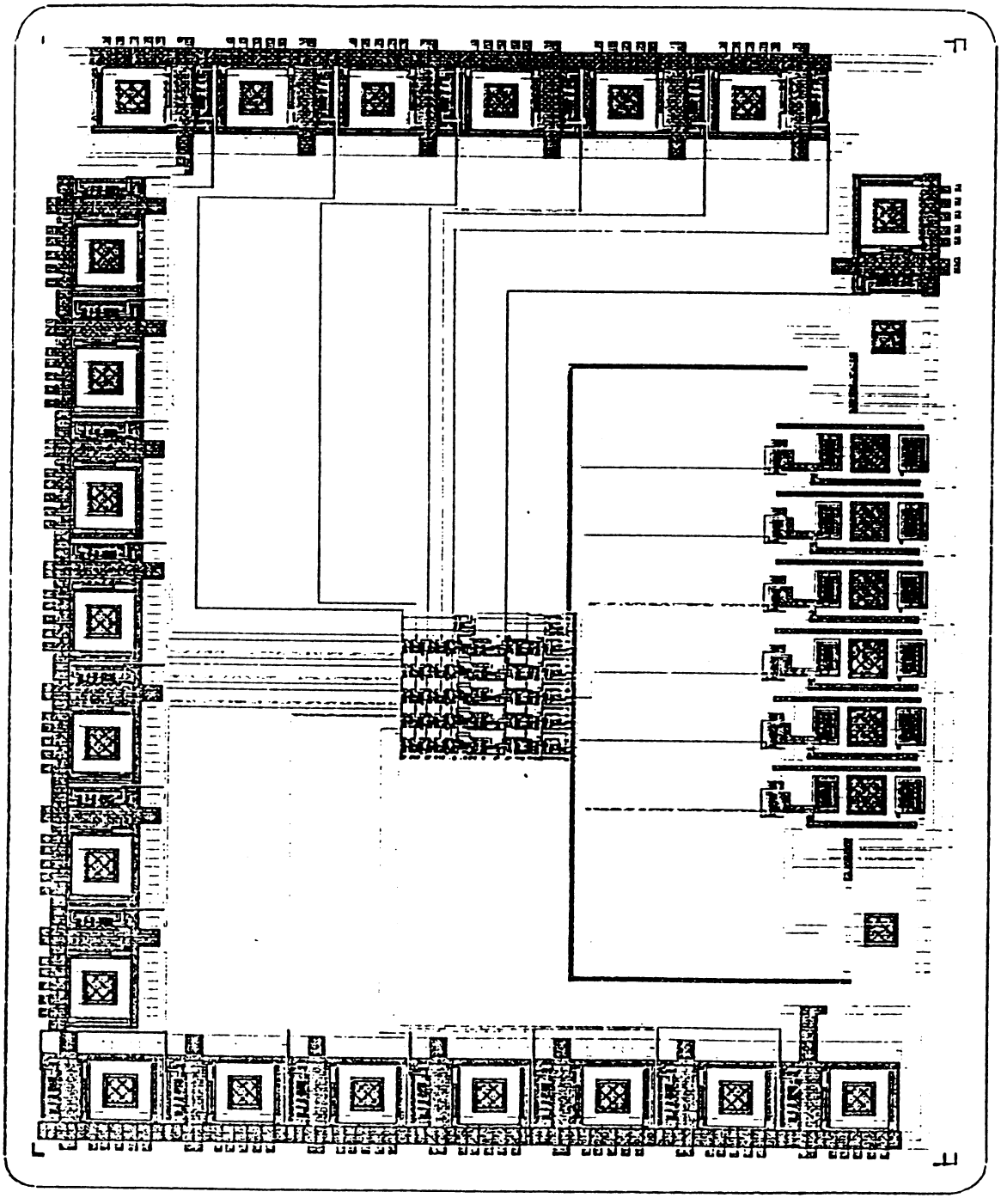
Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
1	1	00000 H	00000 H	00000 H	1
2	1	00000 H	14892 H	14892 H	1
3	1	08441 H	12008 H	1A449 H	1
4	1	10441 H	1E89A H	0ECDB H	0
5	0	1F000 H	1F224 H	1D225 H	0
6	0	1F000 H	0AAB6 H	09AB7 H	0
7	0	1F440 H	1422D H	1366E H	0
8	1	07440 H	00ABF H	07F00 H	1
9	1	00F03 H	00E03 H	01D06 H	1
10	1	00F03 H	14691 H	15594 H	1
11	1	00B42 H	02E0B H	0394D H	1
12	1	10B43 H	1669B H	071DE H	0
13	0	17F03 H	16C27 H	0EB2B H	0
14	0	1FF03 H	024B5 H	023B9 H	0
15	0	1FB43 H	14C2E H	14772 H	0
16	0	0F443 H	04BA0 H	13FE4 H	1
17	1	070FC H	070FC H	0E1F8 H	1
18	1	070FC H	0386E H	0A96A H	1
19	1	074BD H	050F4 H	0C5B1 H	1
20	1	1F4BD H	198A6 H	18D63 H	0

Opération : ADD (suite et fin)

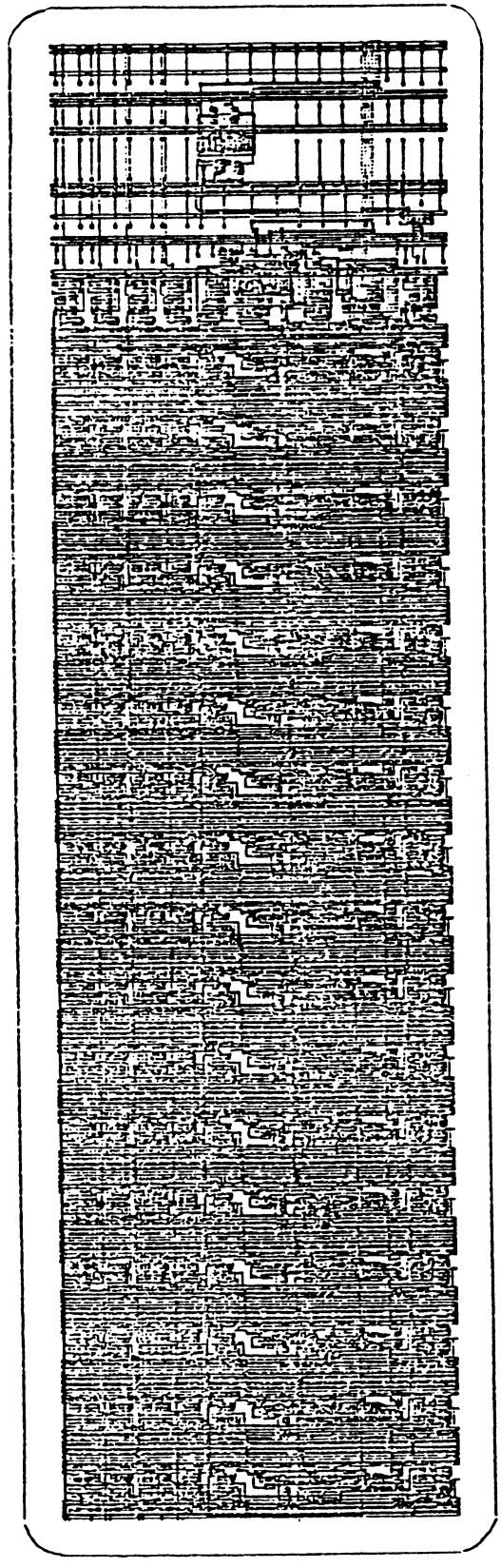
Commandes : CM1 = 1, CM2 = 0, CM3 = 0, CM4 = 1, CM5 = 0.

Vecteurs	Retenue entrante	Entrées		Sorties	Retenue sortante
	$\overline{C0}$	ai	bi	Si	$\overline{C16}$
21	0	1FFFC H	1FDD8 H	1FDD5 H	0
22	0	1FFFC H	1B54A H	1B547 H	0
23	0	1FBBC H	1DDD1 H	1D98E H	0
24	0	07BBF H	01540 H	09100 H	1
25	1	000FF H	011FF H	012FE H	1
26	1	000FF H	0596D H	05A6C H	1
27	1	004BE H	031F7 H	036B5 H	1
28	1	104BE H	17965 H	07E23 H	0
29	0	17FFF H	16CDB H	0ECDB H	0
30	0	17FFF H	12449 H	0A449 H	0
31	0	17BBF H	14CD2 H	0C892 H	0
32	0	07BBF H	14CD2 H	0C892 H	1

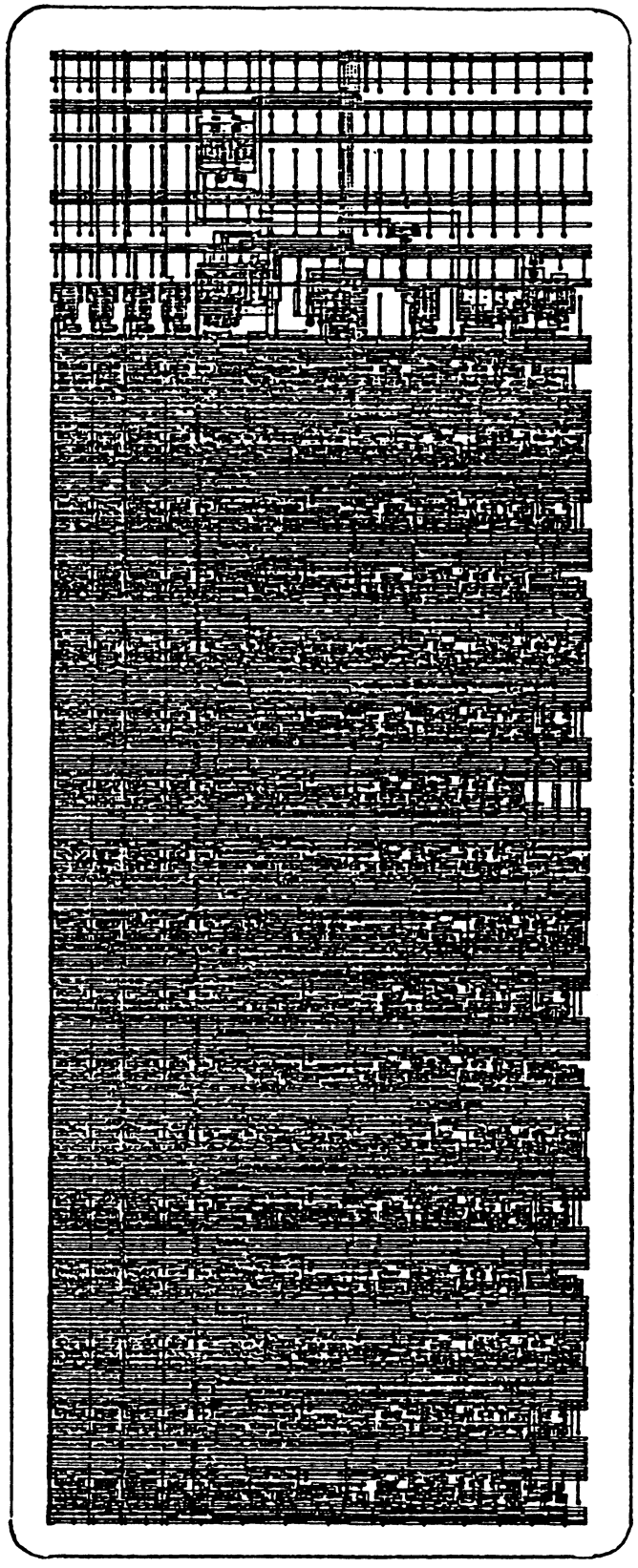
[Annexe 3.3] : schéma d'implantation de l'UAL initiale sur 5 bits



[Annexe 3.4] : schéma d'implantation de l'UAL Manchester sur 17 bits



[Annexe 3.5] : schéma d'implantation de l'UAL Carry-Skip sur 17 bits



Références bibliographiques

[ABO84] E.M. ABOULHAMID and E. CERNY,
"Built-In Testing of One-Dimensional Unilateral Iterative Arrays",
IEEE Trans. on Comput., Vol. C-33, No. 6, June 1984, PP. 560-564.

[ANT86] C. ANTONINO, B. BOSC, H. DELORI, A. GUYOT et J.-F. PAILLOTIN,
"Le CMP français 84-85",
Rapport de Recherche, R.R. n°602, IMAG, Grenoble (France), Février 1986.

[BAR87] P.H. BARDELL, W.H. McANNEY and J. SAVIR,
" Built-In Test for VLSI, Pseudorandom Techniques",
John Wiley & Sons, New York (USA), 1987.

[BED62] O.J. BEDRIJ,
"Carry-Select Adders",
IRE Trans. on Elect. Comput., EC-11, No.3, June 1962, PP. 340-346.

[BEL85] C. BELLON, M. CRASTES DE PAULET, S. HANRIAT, J. RARIVOMANANA and G. SAUCIER,
"CADOC System : A Tool for Multilevel Description and Test Generation for VLSI Circuits",
Proc. of 7th Int. Conf. on Computer Hardware Description Languages, Tokyo (Japan), 1985, PP. 364-380.

[BEW88] G. BEWICK, P. SONG, G.D. MICHELI, and M.J. FLYNN,
"Approaching a Nanosecond : A 32-bit Adder",
Proc. of. ICCD '88, Rye Brook, New York (USA), October 1988,
PP. 221-226.

[BIR88] M. BIRMAN, G. CHU, L. HU, J. MCLEOD, N. BEDARD, F. WARE,
L. TORBAN and C.M. LIM,
" Design of A High-Speed Arithmetic Datapath",
Proc. of ICCD '88, Rye Brook, New York (USA), October 1988,
PP. 214-216.

[BOZ84] S. BOZORGUI-NESBAT and E.J. McCIUSKEY,
"Lower Overhead Design for Testability of PLAs",
Proc. of ITC '84, Philadelphia, Pennsylvania (USA), October 1984,
PP. 856-865.

[BOZ85] S. BOZORGUI-NESBAT and J. KHAKBAZ,
"Minimizing Extra Hardware for Fully Testable PLA Design",
Proc. of ICCD '85, November 1988, PP. 102-104.

[BRE82] R.P. BRENT and H.T. KUNG,
"A Regular Layout for Parallel Adders",
IEEE Trans. on Comput., Vol. C-31, No. 3, March 1982,
PP. 260-264.

[CAR86] W. CARTER,
"Improved Parallel Signature Checkers/Analyzers",
Proc. of 16th FTCS, Vienna (Austria), July 1986, PP. 416-421.

[CER88] E. CERNY, M. ABOULHAMID, G. BOIS, and J. CLOUTIER,
"Built-In Self-Test of A CMOS ALU",
IEEE Design & Test of Comput., August 1988, PP. 38-48.

[CHE87] W.-T. CHENG and J.H. PATEL,
"A Minimum Test Set For Multiple Fault Detection In Ripple Carry Adders",
IEEE Trans. On Comput., Vol. C-36, No. 7, July 1987, PP. 891-895.

[CHE85] G. CHEVALIER and G. SAUCIER,
"A Programmable Switch for Fault-Tolerant Wafer Scale Integration of Processor Arrays",
International Work. On Wafer Scale Integ., Southampton University, July 1985.

[D34] R. LEVEUGLE, F. TIAR and M. SOUEIDAN (INPG), M. GLESNER and N. WHEN (THD),
"Test Strategy of the HYETI Microprocessor (First Run) and Architecture Modifications for the Second Run",
ESPRIT Project 824, Task C, Third Period, Grenoble (France), November 1987.

[D33] R. LEVEUGLE and M. SOUEIDAN (INPG), M. GLESNER and N. WEHN (THD),
"Test of the Two Subsystems (First Run) and Design of the HYETI Microprocessor",
ESPRIT Project 824, Task C, Fourth Period, Grenoble (France), June 1988.

[D35] R. LEVEUGLE and M. SOUEIDAN (INPG), M. GLESNER and N. WEHN (THD),
"The HYETI Microprocessor (Second Run)",
ESPRIT Project 824, Task C, Grenoble (France), November 1988.

[DAV80] R. DAVID,
"Testing by Feedback Shift Register",
IEEE Trans. on Comput., Vol. C-29, No. 7, July 1980, PP. 668-673.

[DIA76] F.J.O. DIAS,
"Truth-Table Verification Of an Iterative Logic Array",
IEEE Trans. on Comput., Vol. C-25, No. 6, June 1976, PP. 605-613.

[DUP88] X. DUPERTHUY et P. RENARD,
"Conception d'une Unité Arithmétique et Logique Rapide pour un Microprocesseur 16 bits à Haut Rendement",
Rapport de Projet, ENSERG, Grenoble (France) 1988.

[EIF84] J.B. EIFERT and J.P. SHEN,
"Processor Monitoring Using Asynchronous Signatured Instruction Streams",
Proc. of 14th FTCS, Kissimmee, Florida (USA), June 1984, PP. 394-399.

[ENS88] C. A. G. ENSTONE,
"Testing Of Iterative Logic Arrays : A Review",
Elect. Desig. Autom., 1988, PP. 149-158.

[FRI73] A.D. FRIEDMAN,
"Easily Testable Iterative Systems",
IEEE Trans. on Comput., Vol. C-22, No. 12, December 1973, PP. 1061-1064.

[FUC88] W.K. FUCHS and M.-F. CHANG,
"Diagnosis and Repair of Large Memories : A Critical Review and Recent Results",
International Workshop on Defect and Fault Tolerance in VLSI Systems, Springfield, Massachusetts (USA), October 1988, PP. 6.1-1-6.1-12

[GEL87] P.P. GELSINGER,
"Design and Test of the 80386",
IEEE Design & Test of Comput., Vol. 4, No. 3, June 1987, PP. 42-50.

[GEN86] P. GENESTIER, C. JAY and G. SAUCIER,
"A reconfigurable Microprocessor for Wafer Scale Integration",
In : "Wafer Scale Integration", G.Saucier and J. Trilhe, ed., Elsevier Science Publishers, Amsterdam (North-Holland), 1986, PP.13-29.

[GEN87] P. GENESTIER,
"Conception de Microprocesseurs à Haute Rendement",
Thèse de Docteur de l'I.N.P.G, Grenoble (France), Juin 1987.

[GIR86] P. GIRARD, F.M. ROCHE and B. PISTOULET,
"Electron Beam Effects on VLSI MOS Condition for Testing and Reconfiguration",
Proc. of The IFIP-Workshop on WSI, I. N. P. Grenoble (France), Mars 1986.

[GUP88] S.K. GUPTA and D.K. PRADHAN,
"A New Framework for Designing and Analyzing BIST Techniques : Computation of Exact Aliasing Probability",
Proc. of ITC '88, Washington D.C. (USA), September 1988, pp. 329-342.

[GUY87] A. GUYOT, B. HOCHET, and J.-M. MULLER,
"A Way to Build Efficient Carry-Skip Adders",
IEEE Trans. on Comput., Vol. C-36, No. 10, October 1987, PP. 1144-1152.

[HAS82a] S.Z. HASSAN, D.J. LU, E.J. McCLUSKEY,
"Parallel Signature Analyzers, Detection Capability and Extensions",
CRC Technique Report, No. 82-20, Stanford CA (USA), December 1982, PP. 1-6.

[HAS82b] S.Z. HASSAN,
"Algebric Analysis of Parallel Signature Analyzers",
CRC Technical Report, No. 82-5, Stanford CA (USA), December 1982.

[HAY76] J.P. HAYES,
"Transition Count Testing of Combinational Logic Circuit",
IEEE Trans. on Comput., Vol C-25, No. 6, June 1976, PP. 613-620.

[HUA88] W.-K. HUANG and F. LOMBARDI,
"On an Improved Design Approach for C-Testable Orthogonal Iterative Arrays",
IEEE Trans. on Comput., Vol. 7, No. 5, May 1988, PP. 609-615.

[HWA79] K. HWANG,
"Computer Arithmetic : Principles, Architecture and Design",
Wyley, New York (USA), 1979.

[IRW88] M.J. IRWIN and R.M. OWENS,
"A Comparison of Two Digit Serial VLSI Adders",
Proc. of ICCD '88, Rye Brook, New York (USA), October 1988, PP. 227-229.

[JAY86] C. JAY,
"HSURF, un Microprocesseur Facilement Testable pour des Applications à Haute Sûreté de Fonctionnement",
Thèse de Docteur de l'USMG, Grenoble (France), Juin 1986.

[JAY88] C. JAY et M. CRASTES DE PAULET,
"Un Compilateur de Blocs de Test Intégré",
Colloque National Conception de Circuits à la Demande, Grenoble, Janvier 1988.

[KAR88] M.G. KARPOVSKY, D. NAGRAVAJARA,
"Board-Level Diagnosis by Signature Analysis",
Proc. of ITC '88, Washington D.C. (USA), September 1988, PP. 47-53.

[KAT86] M. KATEVENIS, M. BLATT,
"Switch Design for Soft-Configurable WSI Systems",
Proc. of the IFIP WG 10.5 Work. On WSI, Elsevier Science Publishers Grenoble (France), March 1986, PP. 255-270.

[KRA82] R.H. KRAMBECK, C.M. LEE and H.-F.S. LAW,
"High-Speed Compact Circuits with CMOS",
IEEE Journ. of Solid-State Circuits,Vol. SC-17, No. 3, June 1982, PP. 614-619.

[KUB84] J. KUBAN and J. SALICK,
"Testability Features of the MC68020",
Proc. of ITC '84, Philadelphia, Pennsylvania (USA), October 1984, PP. 821-836.

[KUO87] S.-Y. KUO and W.K. FUCHS,
"Fault Diagnosis and Spare Allocation for Yield Enhancement In Large Reconfigurable PLAs",
Proc. of ITC '87, Washington D.C. (USA), September 1987, PP. 944-951.

[LEH61] M. LEHMAN, and N. BURLA,
"Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units",
IRE Trans. on Elect. Comput., EC-10, NO.1, January 1961, PP. 67-91.

[LEI86] T. LEIGHTON and C. E. LEISERSON,
"A Survey of Algorithms for Integrating Wafer Scale Systolic Arrays",
In : "Wafer Scale Integration", G. SAUCIER and J. Trilhe, Ed., Elsevier Science Publishers, Amsterdam (North-Holland), 1986, PP. 177-195.

[LEV88a] R. LEVEUGLE et M. SOUEIDAN,
"Conception d'un Microprocesseur à Test en Ligne Intégré pour Applications à Haute Sûreté de Fonctionnement; Outils de CAO Appropriés",
Colloque National Conception de Circuits à la Demande, Grenoble (France), Janvier 1988.

[LEV88b] R. LEVEUGLE and M. SOUEIDAN,
"Design of an Application Specific Microprocessor",
International Workshop on Logic and Architecture Synthesis for Silicon Compilers, Grenoble (France), March 1988.

[LEV88c] R. LEVEUGLE, M. SOUEIDAN and G. SAUCIER,
"Design of an Application Specific Microprocessor with Integrated On-Line Test Facilities",
6th European Workshop on Design for Testability, Garderen (North-Holland), June 1988.

[LEV88d] R. LEVEUGLE, M. SOUEIDAN and X. DELORD,
"HSURF : A Microprocessor with Built-In Test Facilities for Highly Dependable Systems",
6th International Conference on Reliability and maintainability, Strasbourg (France), October 1988, PP.188-193.

[LEV88e] R. LEVEUGLE, M. SOUEIDAN and N. WEHN,
"Defect Tolerance in a 16-Bit Microprocessor",
International Workshop on Defect and Fault Tolerance in VLSI Systems, Springfield, Massachusetts (USA), October 1988, PP. 5.2-1-5.2-12.

[LEV88f] R. LEVEUGLE et M. SOUEIDAN,
"Conception d'un Microprocesseur à Test en Ligne Intégré pour des Applications à Haute Sûreté de Fonctionnement",
L'Onde Electrique, Vol. 68, N°6, Novembre-Décembre 1988, PP. 59-62.

[LIG86] M.M. LIGTHART, E.H.L. AARTS, and F.P.M. BEENKER,
"Design for Testability of PLAs Using Statistical Cooling",
Proc. of 23th DAC, Las Vegas, Nevada (USA), June-July 1986, PP. 339-345.

[LO85] H.Y. LO,
"An Improvement of Nonrestoring Array Divider With Carry-Save and Carry Look Ahead Techniques",
Proc. VLSI 85, Tokyo (Japan), August 1985, PP. 243-251.

[LU82] J.L. LU,
"Watchdog Processors and Structural Integrity Checking",
IEEE Trans. on Comput., Vol. C-31, No. 7, July 1982, PP. 681-685.

[MAC61] O.L. MACSORLEY,
"High-Speed Arithmetic in Binary Computers",
Proc. IRE, Vol. 49, No. 1, January 1961, PP. 61-72.

[MAH83] A. MAHMOOD, E.J. McCLUSKEY and D.J. LU,
"Concurrent Fault Detection Using a Watchdog Processor and Assertions",
Proc. of ITC '83, Philadelphia, Pennsylvania (USA), October 1983, PP. 622-628.

[MAH85] A. MAHMOOD and E.J. McCLUSKEY,
"Watchdog Processor : Error Coverage and Overhead",
Proc. of 15th FTCS, Ann Arbor, Michigan (USA), June 1985, PP. 214-219.

[MAH88] A. MAHMOOD and E.J. McCLUSKEY,
"Concurrent Error Detection Using Watchdog Processor -A Survey",
IEEE Trans. on Comput., Vol. 37, No. 2, February 1988, PP. 160-174.

[MAJ67] S. MAJERSKI,
"On Determination of Optimal Distributions of Carry Skips in Adders",
IEEE Trans. on Comput., Vol. EC-16, NO.1, February 1967, PP. 45-58.

[MAN82] T.E. MANGIR and A. AVIZIENIS,
"Fault-Tolerant Design for VLSI : Effect of Interconnect Requirements on Yield Improvement of VLSI Designs,
IEEE Trans. on Comput., Vol. C-31, No. 7, JULY 1982, PP. 609-615.

[MAN84] T.E. MANGIR,
"Sources of Failures and Yield Improvement for VLSI and Restructurable Interconnects for RVLSI and WSI",
Proc. of the IEEE, Vol. 72, No. 6, June 1984, PP. 690-708.

[MAY88] B. MAYTAL, A. DANOR, V. KARPATI, R. NASSRALLAH, Y. SIDI and E. SHIHADEH,
"Designing for High Yield : the NS32532 Microprocessor",
International Workshop on Defect and Fault Tolerance in VLSI Systems, Springfield, Massachusetts (USA), October 1988, PP. 5.1-1-5.1-7.

[MEA80] C.A. MEAD and L.A. CONWAY,
"Introduction to VLSI System",
Addison-Wesley, Reading, Massachusetts, 1980.

[MET83] L.R. METZGER,
"A 16k CMOS PROM With Polysilicon Fuses Links",
IEEE Journal of Solid-State Circuits, Vol. SC. 18, No. 5, Octobre 1983, PP. 562-567.

[MUZ82] J.C. MUZIO, D.M. MILLER,
"Spectral Techniques for Fault Detection",
Proc. of 12th FTCS, Santa Monica, California (USA), June 1982, PP. 297-302.

[NAM82a] M. NAMJOO and E.J. McCLUSKEY,
"Watchdog Processor and Capability Checking",
Proc. of 12th FTCS, Santa Monica California (USA), June 1982, PP. 245-248.

[NAM82b] M. NAMJOO,
"Techniques for Concurrent Testing of VLSI Processor Operation",
Proc. of ITC '82, Philadelphia, Pennsylvania (USA), November 1982, PP. 461-468.

[NAM83] M. NAMJOO,
"CERBERUS-16 : an Architecture for a General Purpose Watchdog Processor",
Proc. of 13th FTCS, Milano (Italy), June 1983, PP. 216-219.

[NAS88] B. NASREDDINE, E.F. KOUKA, Y. WANG, D. MARRON and J. TRILHE,
"A Reconfigurable SRAM 4.5 Mbits WSI Memory",
International Workshop on Defect and Fault Tolerance in VLSI Systems, Springfield, Massachusetts (USA), October 1988, PP. 6.2-1-6.2-14.

[PAI85] J.-F. PAILLOTIN,
"Langage LUCIE",
Manuel d'Utilisation, Grenoble (France), Juin 1985.

[PAN88] K.F. PANG, H.-W. SOONG, R. SEXTON, and P.-H. ANG,
" Generation of High Speed CMOS Multiplier-Accumulators",
Proc. of ICCD '88, Rye Brook, New York (USA), October 1988, PP. 217-220.

[PAR81] R. PARTHASARATHY and S.M. REDDY,
"A Testable Design of Iterative Logic Arrays",
IEEE Trans. on Comput., Vol. C-30, No. 11, November 1981, PP. 833-841.

[PIL85] E. PILAUD et G. SAUCIER,
"Le Concept de Test en Ligne"
L'Onde Electrique, Mai-Juin 1985.

[POI88] F. POIROT, M. CRASTES DE PAULET, C. DUFF, P. SICARD and G. SAUCIER,
"Controller Synthesis in the ASYL System",
International Workshop on Logic and Architecture Synthesis for Silicon Compilers, Grenoble (France), March 1988.

[REA85] S.M. READY, K.K. SALUJA and M.G. KARPOVSKY,
"A Data Compression for Built-In Self Test",
Proc. of 15th FTCS, Ann Arbor, Michigan (USA), June 1985, PP. 294-299.

[ROB87] J.P. ROBINSON and N.R. SAXENA,
"A Unified View of Test Compression Methods",
IEEE Trans. on Comput., Vol. C-36, No. 1, January 1987, PP. 94-99.

[ROB88] J.P. ROBINSON and N.R. SAXENA,
"Simultaneous Signature and Syndrome Compression"
IEEE Trans. on Comput. Vol C-7, No.5, May 1988, PP. 584-589.

[SAL83] K.K. SALUJA, K. KINOSHITA and H. FUJIWARA,
"An Easily Testable Design of PLAs for Multiple Faults,
IEEE trans. on Comput., Vol. C-32, No. 11, November 1983, PP. 1088-1097.

[SAM86] M.G. SAMI and R. STEFANELLI,
"Reconfigurable Architectures for VLSI Processing Arrays",
Proc. of the IEEE, Vol. 74, No. 5, MAY 1986, PP. 712-722.

[SAU84] G. SAUCIER and C. JAY,
"Design and Use of a Safety Related Microprocessor"
4th International Conference on Reliability and maintainability, Perros-Guirec (France), May 1984, PP.561-566.

[SAU85] G. SAUCIER,
"Conception de Circuits",
ENSIMAG-Cours Licence d'Informatique, Grenoble (France) 1985.

[SAU87] G. SAUCIER, M. CRASTES DE PAULET and P. SICARD,
"ASYL : A Rule-Based System for Controller Synthesis",
IEEE transactions on CAD, November 1987, PP. 1088-1097.

[SAV80] J. SAVIR,
"Syndrome Testable Design of Combinational Circuits",
IEEE Trans. on Comput., Vol. C-29, No. 6, June 1980, PP. 442-451.

[SAV81] J. SAVIR,
"Syndrome-Testing of 'Syndrome-Untestable' Combinational Circuits",
IEEE Trans. on Comput., Vol. C-30, No. 8, August 1981, PP. 606-608.

[SCH87] M.A. SCHUETTE and J.P. SHEN,
"Processor Control Flow Monitoring Using Signatured Instruction Streams",
IEEE Trans. on Comput., Vol. C-36, No. 3, March 1987, PP. 264-276.

[SEV84] M. SEVESTRE,
"Validation of a Microprocessor Based Railway Safety System",
4th International Conference on Reliability and maintainability, Perros-Guirec (France), May 1984, PP.586-589.

[SHA84] D.C. SHAVER,
"Electron Beam Customisation, Repair, and Testing of Wafer-Scale Circuits",
Solid State Techn., February 1984, PP. 135-139.

[SHE83] J.P. SHEN and M.A. SCHUETTE,
"On-Line Self-Monitoring Using Signatured Instruction Streams",
Proc. of ITC '83, Philadelphia, Pennsylvania (USA), October 1983, PP. 275-282.

[SHE84] J.P. SHEN and F.J. FERGUSON,
"The Design of Easily Testable VLSI Array Multipliers",
IEEE Trans. on Comput., Vol. C-33, No. 6, June 1984, PP. 554-560.

[SHU87] A.S. SHUBAT, J.A. PERTORIUS, and C.A.T. SALAMA,
"Expandable Arithmetic Block Macrocell",
Integration Workshop, The VLSI Journal, May 1987, PP. 47-71.

[SKL63] J. SKLANSKY and M. LEHMAN,
"Ultimate-Speed Adders",
IRE Trans. on Elect. Comput., EC-12, No. 2, April 1963, PP. 142-148.

[SMI80] J.E. SMITH,
"Measures of the Effectiveness of Fault Signature Analysis",
IEEE Trans. on Comput., Vol. C-29, No. 6, June 1980, PP. 510-514.

[SMI87] E.L. SMITT,
"Signature Analysis Testing in Large Standard-Cell ASICs",
VLSI Systems Design, May 1987, PP. 46-52.

[SOM86] F. SOMENZI and S. GAI,
"Fault Detection In Programmable Arrays,
Proc. of the IEEE, Vol. 74, No. 5, May 1986, PP. 655-668.

[SOS88] J. SOSNOWSKI,
"Detection of Control Flow Errors Using Signature and Checking Instructions",
Proc. of ITC '88, Washington D.C. (USA), September 1988, PP. 81-88.

[SOU86] M. SOUEIDAN,
"Etude et Réalisation d'un Circuit de Signature Parallèle en CMOS",
Rapport de Recherche, R. R. 609-1., IMAG Grenoble (France), Mai 1986.

[SRI81] T. SRIDHAR and J.P. HAYES,
"A Functional Approach to Testing Bit-Sliced Microprocessors",
IEEE Trans. on Comput., Vol. C-30, No. 8, August 1981, PP. 563-571.

[SRI82a] T. SRIDHAR, D.S. HO and S.M. THATTE,
"Concurrent Checking of Program Flow in VLSI Processors",
Proc. of ITC '82, Philadelphia, Pennsylvania (USA), November 1982, PP. 191-199.

[SRI82b] T. SRIDHAR, D.S. HO, T.J. POWELL and S.M. THATTE,
"Analysis and Simulation of Parallel Signature Analysers",
Proc. of ITC '82, Philadelphia, Pennsylvania (USA), November 1982, PP. 656-661.

[SUM86] G.W. SUMERLING, G.E. DIXON and A. K.J. STEWART,
"An Assessment of Non-Regular Cell Based Architecture for ULSI and WSI",
In : "Wafer Scale Integration", G. SAUCIER and J. Trilhe, Ed., Elsevier Science Publishers, Amsterdam (North-Holland), 1986, PP. 3-12.

[SUS81] A.K. SUSSKIND,
"Testing by Verifying Walsh Coefficients",
Proc. of 11th FTCS, Portland, Maine (USA) June 1981, PP. 206-208.

[SUS83] A.K. SUSSKIND,
"Testing by Verifying Walsh Coefficients",
IEEE Trans. on Comput., Vol. C-32, No.2, February 1983, PP. 198-201.

[TEE87] G.H. TEEPE and W.L. ENGL,
"A Bipolar Correlator With Redundancy",
IEEE Journ. of Solid-State Circuits, Vol. SC-22, No. 6, December 1987, PP. 1190-1195.

[TRI88] J. TRILHE,
" Wafer Scale Integration : Mythe ou réalité",
3ème Colloque National de Circuits Intégrés à la demande, Grenoble (France), Janvier 1988.

[TUN86] C.-H. TUNG and J.P. ROBINSON,
"On Concurrent Testable Microprogrammed Control Units",
Proc. of ITC '86, Philadelphia, Pennsylvania (USA), September 1986, PP. 895-900.

[VAL86] C. VAL,
"Wafer Scale Integration Packaging",
Proc. of the IFIP-Workshop on WSI, Grenoble (France), March 1986.

[VAR88] P. VARMA and Y. TOHMA,
"Aknowledge-Based Test Generator for Standard Cell and Iterative Array Logic Circuits",
IEEE Journ. of Solid-State Circuits, Vol. 23, No. 2, April 1988, PP. 428-436.

[VEN80] C.S. VENKATRAMAN, K.K. SALUJA,
"Transition Count Testing of Sequential Machines",
Proc. of 10th FTCS, Kyoto (Japan), October 1980, PP. 167-172.

[VTI88] "Manuels utilisateur du système VTI",
VLSI Technology Inc, 1988.

[WAN82] L.-T. WANG,
"Autonomous Linear Feedback Shift Register With On-Line Fault-Detection Capability",
Proc. of 12th FTCS, Santa Monica, California (USA) June 1982, PP. 311-314.

[WAN86a] L.-T. WANG, E.J. McCLUSKEY,
"Circuits for Pseudo-Exhaustive Test Pattern Generation",
Proc. of ITC '86, Philadelphia, Pennsylvania (USA), Septembre 1986, PP. 25-37.

[WAN86b] L.-T. WANG, E.J. McCLUSKEY,
"A Hybrid Design of Maximum-Length Sequence Generators",
Proc. of ITC '86, Philadelphia, Pennsylvania (USA), Septembre 1986, PP. 38-47.

[WAN86c] L.-T. WANG, E.J. McCLUSKEY,
"Feedback Shift Registers for Self-Testing Circuits",
VLSI Systems Design, December 1986, PP. 50-58.

[WEH88] N. WEHN, M. GLESNER, K. CAESAR, P. MANN and A. ROTH,
"A Defect-Tolerant and Fully Testable PLA",
Proc. of the 25th DAC, Las Vegas, Nevada (USA), June 1988, PP. 22-27.

[WES85] N. WEST and K. ESHRAGHIAN,
"Principles of CMOS VLSI Design, a System Perspective"
Addison-Wesley, Reading, Massachusetts, 1985.

[WEY87] C.-L. WEY, M.K. VAI and F. LOMBARDI,
"On The Design of a Redundant Programmable Logic Array (RPLA)",
IEEE Journal of Solid-state Circuits, Vol. SC-22, No. 1, February 1987, PP. 114-118.

[WEY88] C.-L. WEY,
"On Yield Consideration for the Design of Redundant Programmable Logic Arrays",
IEEE trans. on C.A.D., Vol. 7, No. 4, April 1988, PP. 528-535.

[WILK87] K.D. WILKEN and J.P. SHEN,
"Embedded Signature Monitoring : Analysis and Technique",
Proc. of ITC '87, Washington D.C. (USA), September 1987, PP. 324-333.

[WILK88] K.D. WILKEN and J.P. SHEN,
"Continuous Signature Monitoring : Efficient Concurrent-Detection of Processor Control Errors",
Proc. of ITC '88, Washington D.C. (USA), September 1988, PP. 914-925.

[WILL87] T.W. WILLIAMS, W. DAEHN, M. GRUETZNER and C.W. STARKE,
"Aliasing Errors With Primitive and Non-Primitive Polynomials",
Proc. of ITC '87, Washington D.C. (USA), September 1987, PP. 637-644.

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'Arrêté du 23 novembre 1988 relatif aux Etudes doctorales

VU les rapports de présentation de Messieurs

. Francis JUTAND , Professeur
. Jean-Louis LARDY , Ingénieur

Monsieur SOUEIDAN Mohammad

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité " Microélectronique "

Fait à Grenoble, le 31 mars 1989

Pour le Président de l'I.N.P.-G.
et par délégation,
le Vice-Président
P. VENNEREAU

Résumé

L'objet de cette thèse est la conception et la réalisation d'un microprocesseur reconfigurable en fin de fabrication afin de tolérer les défauts de fabrication. Il s'agit d'un microprocesseur destiné à être le cœur d'un microcontrôleur pour les applications de l'automatisme à haute sûreté de fonctionnement. Pour améliorer le rendement en fin de fabrication, des éléments redondants pouvant remplacer les éléments défectueux sont implantés sur le silicium.

Une étude particulière a concerné les unités arithmétiques et logiques; les quatre critères d'évaluation étant leur complexité, leur performance, leur testabilité et leur reconfigurabilité.

Enfin, des circuits de compaction d'information par division polynômiale implantés dans ce microprocesseur ont également fait l'objet d'une attention spéciale essentiellement en ce qui concerne leur testabilité.

Un deuxième prototype de ce microprocesseur HYETI a été implanté en technologie HCMOS3 1,2 µm et fabriqué dans le cadre du projet ESPRIT 824.

Mots clés : circuits reconfigurables, microprocesseur, UAL testables et reconfigurables, LFSR, test en ligne, test hors ligne.

Abstract

The design and the realization of an end of manufacturing reconfigurable microprocessor which tolerates manufacturing defects are presented. This microprocessor has been designed to be the core of a microcontroller dedicated to high safety automatic applications. In order to improve the manufactoring yield, redundant elements able to replace defective ones are integrated on the wafer.

Particular attention is given to the arithmetic and logic units, for which the evaluation criteria are the complexity, the performance, the testability and the reconfigurability.

Data compaction circuits based on polynomial division integrated in this microprocessor are studied more particularly from the point of view of testability .

A second prototype of the HYETI microprocessor has been manufactured with the HCMOS3 1.2 µm technology in the framework of the ESPRIT project 824.

Keywords : reconfigurable circuits, microprocessor, testable and reconfigurable ALU, LFSR, on line test, off line test,.