APLYSIE : un circuit neuro-mimétique : réalisation et intégration sur tranche

Philippe Hurat

THESE

présentée par

Philippe HURAT

pour obtenir le titre de **DOCTEUR**

de **l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**
(arrêté ministériel du 5 juillet 1984)

spécialité : microélectronique

❑❑❑

APLYSIE

Un Circuit Neuro-Mimétique

Réalisation et Intégration sur Tranche

❑❑❑

Date de soutenance : 24 février 1989

Compostion du jury : MM. François ROBERT Président
Francis JUTAND Rapporteur
Guy MAZARE Rapporteur
Max FONTET Examinateur
Paul CASPI Examinateur

Thèse préparée au sein du Laboratoire de Génie Informatique (LGI/UGM)

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 CORREC	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 agréées 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 - 198

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

Monsieur **François ROBERT** Professeur à l'ENSIMAG, qui, dès le début, a soutenu ce travail, m'a fait l'honneur d'accepter de présider le jury de cette thèse. Je lui adresse mes plus vifs remerciements.

Monsieur **Francis JUTAND**, Professeur à TELECOM Paris, responsable du département électronique, et monsieur **Guy MAZARE**, Professeur à l'ENSIMAG sont rapporteurs de cette thèse. Je les remercie d'avoir accepté cette tâche.

Je remercie aussi Monsieur **Max FONTET**, Professeur à l'Université PARIS VI et adjoint au directeur de la division informatique des laboratoires de Marcoussis (CGE), d'avoir bien voulu faire partie du jury de cette thèse.

Monsieur **Paul CASPI**, attaché de recherche au CNRS et responsable de l'Unité Génie Matériel du Laboratoire de Génie Informatique, a assuré l'encadrement de cette thèse. Je lui témoigne ma gratitude pour sa confiance et pour son apport lors de la formalisation théorique de ce travail.

Beaucoup des travaux de cette thèse ont été menés en commun avec François BLAYO. Je souhaite qu'il trouve ici l'expression de mes plus sincères remerciements pour sa précieuse collaboration.

Je remercie tous mes collègues de l'Unité Génie Matériel, de l'INPG et de l'IMAG et plus particulièrement
Catherine BELLON pour son travail d'encadrement et son aide constante,
Paul AMBLARD pour sa relecture attentive,
le service du CMP et les membres de l'USSI pour leur assistance technique et leur gentillesse.

Enfin, Sylvie, je te remercie pour ton soutien quotidien, tes nombreuses relectures et d'avoir su supporter avec patience un thèsard souvent anxieux.

aplysie n. f. (*a* priv., et gr. *plunien*, laver). Mollusque marin limaciforme sécrétant un liquide violet. Animal d'expérimentation en neurobiologie. (Nom usuel : *lièvre des mers*.) [Sous classe des opisthobranches.]

à mes parents,

à Sylvie.

Introduction

Les sciences connexionistes ont suivi un parcours original. Nées presque en même temps que l'approche traditionnelle Von Neuman, elles sont sujettes à un brusque revers dans les années 60. Pendant une vingtaine d'années, le connexionisme devient marginal. Les recherches se poursuivent mais rares sont les chercheurs qui s'avouent de l'école connexioniste.
Puis, la communauté scientifique redécouvre le connexionisme et les réseaux de neurones artificiels ou réseaux neuro-mimétiques avec l'article du physicien Hopfield [HOP82] qui a un retentissement surprenant. Cet article n'apporte pas de solution aux problèmes restés en suspens pendant 20 ans, mais sert de détonateur. Depuis, l'intérêt pour l'approche connexioniste ne cesse de croître.

Calqués sur les modèles biologiques, les réseaux neuro-mimétiques sont basés sur un parallélisme à grain fin où les cellules appelées neurones interagissent fortement. Chaque cellule participe au traitement et à la représentation des données : on a une représentation distribuée de l'information.

La programmation d'un réseau de neurones est un concept entièrement nouveau. Il ne s'agit plus de décrire exhaustivement la méthode de résolution d'un problème comme avec la programmation algorithmique classique. Ici, la programmation se fait à partir d'un ensemble significatif d'exemples, qui sert de base à l'apprentissage. On parle alors d'adaptation. L'adaptation a pour but de définir l'interaction des cellules neuroniques, qui régit le comportement du réseau.

Si l'approche connexioniste est conceptuellement très différente de celle de Von Neuman, elle ne lui est pas directement concurrente. C'est d'ailleurs, en partie, grâce à l'accroissement des performances des machines actuelles,

séquentielles ou parallèles, que les recherches dans ce domaine ont pu progresser de façon aussi spectaculaire.

Pour aller au dela de cette étape de recherche et de simulations, de véritables machines neuro-mimétiques doivent être développées. Dans ce cadre, ce rapport de thèse présente la réalisation d'un circuit neuro-mimétique et l'étude de son intégration sur tranche entière.

Dans le premier chapitre, nous allons introduire la notion de neurone formel. Puis nous nous intéressons aux différents modèles neuronaux, à leur comportement en réseau et aux applications qui en découlent.

Le deuxième chapitre présente l'analyse des machines neuro-mimétiques et des réalisations matérielles existantes, ou en cours de réalisation. Ceci permet de mieux cerner le cadre de notre étude. Nous montrons également que la conception de circuits dédiés aux réseaux de neurones n'est raisonnablement envisageable que pour des modèles théoriquement bien spécifiés. Nous justifions notre choix d'intégrer la phase de reconnaissance d'un réseau de Hopfield.

Afin de résoudre les problèmes liés à l'interconnexion complète des neurones, nous avons défini une architecture systolique présentée dans le chapitre trois.

Chaque cellule de cette architecture implante une fonction synaptique. Le quatrième chapitre présente la démarche de conception et la phase d'intégration sur le silicium d'une cellule Aplysie 1.

Comme le test de ce premier circuit a été couronné de succès, nous avons poursuivi notre travail d'intégration VLSI en concevant un réseau de 16

neurones : Aplysie 16. La conception de cette puce est décrite dans le chapitre cinq.

Nous présentons les perspectives de développement dans le chapitre six.

Le chapitre sept développe une de ces perspectives : l'intégration sur tranche entière (WSI). Les défauts de fabrication sont inhérents à l'intégration sur tranche, aussi présentons nous les différents aspects de la reconfiguration. Cette étude nous a amenés à définir un élément de commutation programmable. Nous terminons ce chapitre et ce rapport de thèse par l'étude de faisabilité de l'intégration sur tranche d'un réseau neuronique à base d'Aplysies.

I. Modèles & Applications

Le développement de l'approche neuro-mimétique est fondé sur une vision simplifiée du comportement des cellules nerveuses ou neurones.
Le corps du neurone, appelé aussi soma, dispose de deux types d'appendices :

- les dendrites reçoivent les activités des autres neurones au travers de contacts appelés synapses. Lorsque le côté pré-synaptique est actif, une faible tension est engendrée du côté post-synaptique. La somme des tensions post-synaptiques constitue le potentiel de membrane.
- l'axone constitue la sortie du neurone. Quand le potentiel de membrane dépasse une valeur seuil, un potentiel d'action, en anglais *spike*, est engendré et se propage de proche en proche le long de l'axone jusqu'aux synapses terminales.

D'après [VIL88], on peut dire que les cellules ont une entrée analogique par les dendrites et une sortie digitale par l'axone.

C'est à partir de ce modèle biologique simplifié que l'on établit le modèle des neurones formels qui sont les constituants de base des réseaux de neurones artificiels. En décrivant leurs différentes applications, nous introduirons les divers types de réseaux neuroniques.

I. LES NEURONES FORMELS

I.1. Le modèle générique

Un neurone reçoit les activités des neurones auxquels il est connecté au travers de synapses. Ces dernières ont pour rôle de définir l'interaction entre les neurones. La force de la jonction synaptique du neurone i qui reçoit l'information du neurone j est modélisée par le coefficient synaptique C_{ij}. Un neurone va évaluer son potentiel de membrane en effectuant la somme de ses

entrées pondérée par les coefficients synaptiques. Ensuite, il prend une décision et modifie son activité si nécessaire. Cette décision dépend de la somme pondérée des entrées et du seuil d'activité θ_i du neurone. Une fonction non linéaire F modélise cette prise de décision. On obtient alors le modèle générique suivant :

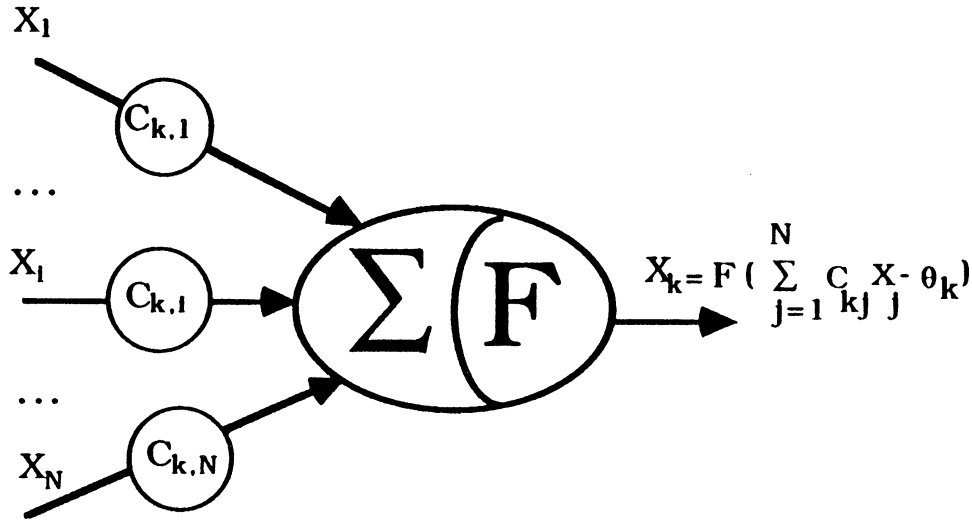


Figure I.1 Modèle générique.

Remarque : le nombre de neurones d'un réseau sera noté N.

I.2. Le modèle de Mac Culloch et Pitt

Dans le modèle initialement développé par Mac Culloch et Pitt (1943) et repris par Hopfield (1982), l'activité d'un neurone est binaire. Deux variantes peuvent être définies :

Modèle 1 : l'état 1 représente un neurone actif alors qu'un neurone inactif est représenté par un état 0. La décision est déterminée par une fonction de seuil qui compare la somme pondérée avec le seuil θ_i du neurone (voir figure I.2a) :

$$\mathbf{X_i = si} \quad \sum_{j=1}^{N} \mathbf{C_{ij} \, . \, X_j > \theta_i} \quad \mathbf{alors +1 \ \ sinon \ 0.} \qquad \mathbf{(I.1)}$$

Modèle 2 : les états actif et inactif sont représentés respectivement par +1 et -1. La fonction de décision compare la somme pondérée et le seuil (voir figure I.2b) :

$$X_i = si \quad \sum_{j=1}^{N} C_{ij} \cdot X_j - \theta_i > 0, \quad \textbf{alors +1 sinon -1.} \qquad \textbf{(I.2)}$$

Ces 2 modèles sont strictement équivalents [BRU86], [LEC87]. Tout réseau de neurones de type 1 possède un équivalent suivant le modèle 2. Soient (C_1, ..., C_N, θ) les paramètres relatifs à un neurone du modèle 1 (0, 1), les paramètres du neurone équivalent de type 2 (-1, +1) sont (C_1', ..., C_N', θ') avec :

$$C_i' = \frac{1}{2} C_i \quad \textbf{et} \quad \theta' = \theta - \frac{1}{2} \sum_{i=1}^{N} C_i \qquad \textbf{(I.3)}$$

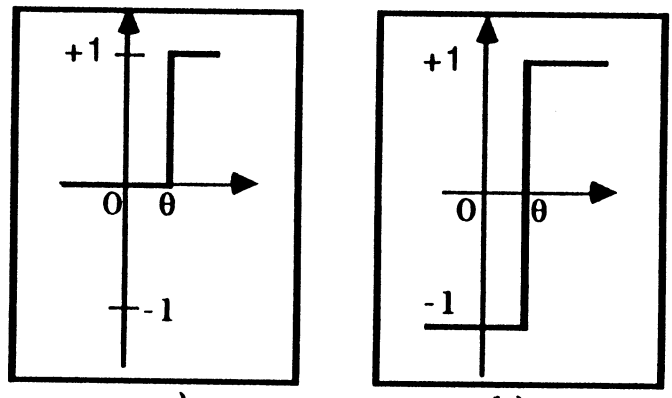


Figure I.2 Fonctions de seuil du modèle Mac Culloch et Pitt.

Ces modèles sont particulièrement bien adaptés aux applications dont les informations peuvent se représenter sous forme binaire : par exemple, le traitement d'image noir et blanc. Un pixel de l'image peut alors être directement associé à un neurone d'entrée :

- blanc ⇒ neurone inactif ⇒ état 0 ou -1,
- noir ⇒ neurone actif ⇒ état +1.

Lorsque les entrées prennent des valeurs continues ces modèles sont moins appropriés et d'autres modèles sont alors utilisés.

I.3. Les modèles logique et sigmoïde.

D'autres modèles sont obtenus en utilisant des fonctions de décision plus complexes à valeurs réelles [HOP84], [JOR86]. Deux modèles sont souvent utilisés :

- le modèle logique prend ses valeurs entre 0 et +1. La décision est modélisée par une fonction de décision continue comme le montre la figure I.3a.
- la fonction de décision est de type sigmoïde. L'activité du neurone peut prendre une valeur réelle entre -1 et +1. La fonction de décision est une fonction sigmoïde calculée par la formule suivante [LEC87] :

$$F(x) = \frac{(e^{kx} - 1)}{(e^{kx} + 1)} = th\left(\frac{kx}{2}\right) \qquad \text{(I.4)}$$

Cette fonction impaire, strictement croissante possède deux asymptotes d'ordonnée -1 et +1 (cf figure I.3b). Le paramètre k agit sur la courbure de la sigmoïde. Quand k tend vers l'infini, la fonction sigmoïde tend vers la fonction de seuil.

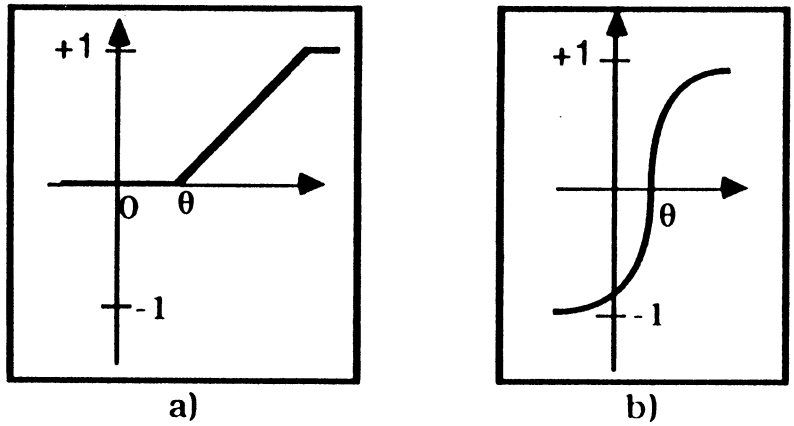


Figure I.3 Modèle logique Modèle sigmoïde

Remarque : lorsque k tend vers l'infini, la fonction sigmoïde exprimée par la formule (I.4) tend vers la fonction de seuil (-1, +1) de la figure I.2b.
Cette fonction est beaucoup plus proche de la relation existant entre l'activité et le potentiel de membrane du neurone biologique. Les autres fonctions de décision présentées auparavant, n'en sont que des approximations.

I.4. Réseau de neurones

La capacité de traitement d'un neurone formel est très limitée. Il n'en va pas de même des assemblages de telles cellules. Les groupements de neurones formels constituent des réseaux qui peuvent être de deux types :

- combinatoire,
- itératif.

Les réseaux combinatoires sont ainsi appelés par référence aux circuits combinatoires. En effet, ils n'ont pas d'état interne et la sortie ne dépend que des valeurs d'entrée.
Dans les réseaux itératifs, les sorties dépendent des valeurs d'entrée et de l'état précédent du réseau. Ils sont caractérisés par une boucle de rétro-action qui reboucle les sorties sur les entrées.

I.5. Apprentissage

Une fois le réseau défini par le choix du modèle de neurone utilisé, son utilisation se décompose en deux étapes :

- la phase d'apprentissage,
- la phase de reconnaissance.

L'apprentissage ou phase d'adaptation a pour principal but la définition des poids synaptiques. En effet, la dynamique du réseau est déterminée par :

- les coefficients synaptiques,
- la fonction de décision de chaque neurone.

La disponibilité d'une loi d'apprentissage efficace est fondamentale pour une utilisation pratique des réseaux neuroniques. D'ailleurs, les réseaux de neurones ont vécu une période sombre entre 1960 et 1980 car les règles d'adaptation faisaient cruellement défaut ou étaient très limitées. C'est avec l'émergence de nouvelles lois d'apprentissage que la vague d'intérêt actuelle pour les systèmes neuro-mimétiques est apparue.

Ces lois sont bien connues et établies pour des réseaux simples où la dynamique et l'état général du réseau sont connus d'avance. Par contre, les réseaux plus complexes, comprenant par exemple des couches cachées dont on ne peut connaître à priori l'état d'activité, nécessitent des lois plus sophistiquées. Ces dernières sont d'ailleurs en pleine évolution et sont largement étudiées.

Les lois d'apprentissage connexionistes peuvent être divisées en trois classes [HIN87] :

- l'apprentissage supervisé nécessite un superviseur pour donner au réseau les réponses attendues,
- l'apprentissage quasi-supervisé requiert seulement une évaluation numérique de la qualité de la réponse donnée par le réseau,
- l'apprentissage non supervisé construit des représentations internes et il est capable de généraliser sans aide extérieure.

Une fois la matrice synaptique spécifiée, la phase de reconnaissance peut être exécutée. Après avoir initialisé les entrées, le réseau évolue librement. Les activités des neurones sont modifiées en fonction des coefficients synaptiques pré-calculés et du modèle de neurone choisi. Quand cette évolution se stabilise, le réseau a convergé. Les activités des neurones de sortie après la convergence, composent la réponse du système.

Le modèle du réseau ainsi que la loi d'apprentissage dépendent essentiellement de l'application envisagée. Les principales utilisations des

réseaux connexionistes que nous allons détailler sont les mémoires associatives, la résolution de problèmes d'optimisation et la classification [HOP85], [RUM86a].

II. MEMOIRE AUTO-ASSOCIATIVE

La fonction d'une mémoire auto-associative est de retrouver une information, préalablement mémorisée, à partir de son contenu. Une mémoire auto-associative retrouve la totalité d'une information stockée à partir d'éléments incomplets, erronés ou imprécis : la clé de recherche. La clé est présentée à la mémoire auto-associative. Celle-ci donne en réponse l'ensemble des éléments lui correspondant. Les réponses peuvent être simples, multiples ou inexistantes.

Les réalisations matérielles actuellement développées sont relativement élémentaires. Les concepts plus sophistiqués comme la correction d'erreur ne sont pas encore implantés en interne dans la mémoire.

II.1. Principe

Un réseau de neurones peut être utilisé comme mémoire auto-associative. Tout d'abord, la clé est imposée comme état initial du réseau. Chaque neurone va avoir une activité définie en fonction de la recherche à faire. Ensuite, le réseau évolue librement. Les activités des neurones vont être modifiées en fonction des coefficients synaptiques pré-calculés. Si ceux-ci sont correctement choisis, le réseau converge vers un état stable. L'ensemble des activités des neurones après convergence détermine la réponse du système.

Une fonction d'énergie peut être associée aux états du réseau [HOP82]. Les états stables sont alors des minima locaux.

Pour obtenir un comportement de mémoire auto-associative, il faut préalablement mémoriser les informations ou prototypes. Dans un réseau de

neurones, apprendre des prototypes consiste à imposer un comportement au réseau, de telle sorte que ces prototypes sont des états stables. En fonction du modèle de neurone choisi, la phase d'apprentissage effectue la synthèse des prototypes à apprendre et produit la matrice synaptique. Si l'on considère l'espace des états des neurones, l'apprentissage vise à construire des puits d'énergie correspondant aux états stables définis par les prototypes.

Les états proches d'un état stable sont constitués d'éléments partiels du prototype. Partant d'un état initial correspondant à une information partielle, le réseau converge alors vers l'information complète.

Un prototype est un minimum local vers lequel le réseau est attiré quand son état initial est proche de ce prototype. L'information mémorisée n'est pas retrouvée par son adresse mais à partir d'une clé faisant référence au contenu de l'information. N'importe quelle clé de taille suffisante permet d'arriver au bon résultat : l'information est adressable par son contenu et non par son emplacement. La fonction de mémoire auto-associative est réalisée.

Pour avoir un comportement de mémoire auto-associative, il faut associer un prototype X à lui-même. La matrice synaptique C devra alors vérifier :

$$X = F(C.X) \qquad (I.5)$$

La construction de la matrice doit assurer la convergence de la phase de reconnaissance (recherche). Le système va donc toujours délivrer une solution. Cependant, le résultat de la recherche est sans alternative : le système ne donne qu'une seule réponse. Ceci permet de minimiser les conflits et la réponse du réseau est en quelque sorte une réponse optimale.

II.2. Réseaux de Hopfield

Les réseaux les plus couramment utilisés comme mémoires auto-associatives sont les réseaux de Hopfield. Distinguons deux variantes de ce réseau :

- le réseau asynchrone,
- le réseau synchrone.

Ces deux variantes sont assez proches car elles utilisent, toutes deux, des neurones du type Mac Culloch et Pitt. Ces neurones sont tous reliés entre eux sur la même couche comme le montre la figure I.4 et forment un réseau itératif.

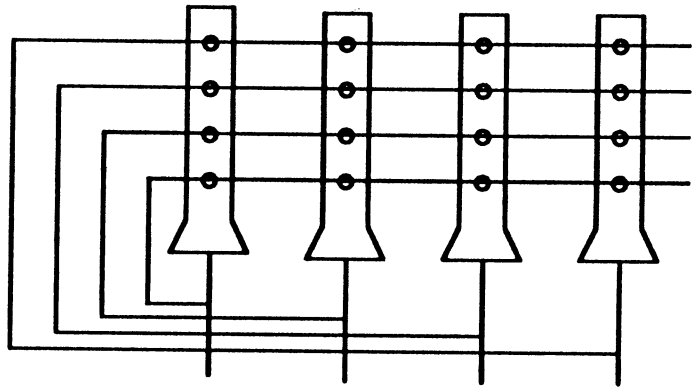


Figure I.4 Réseau de Hopfield

Comme on peut le voir sur la figure I.4, chaque nœud interagit avec les autres. Les sorties sont rebouclées sur les entrées. Ce sont des réseaux monocouche avec boucle de rétro-action.
Les deux variantes diffèrent par leur mode d'évolution.

II.2.1. Description du réseau asynchrone

Afin de coller à la réalité biologique, Hopfield [HOP82] préconise une mise à jour des neurones asynchrone et aléatoire. La décision prise par un neurone est indépendante de l'instant où les autres neurones effectuent leur mise à jour. La loi d'évolution suivie par les N neurones d'un réseau est :

$$X_i = F\left(\sum_{j=1}^{N} C_{ij} \cdot X_j\right) \qquad \text{où F est une fonction de seuil} \qquad (I.6)$$

Les conditions de convergence de cette évolution ont été énoncées par [HOP82]. La fonction d'énergie associée à l'évolution asynchrone est décroissante si la matrice est symétrique à diagonale nulle.

Cette évolution asynchrone permet de simuler les délais de propagation, et les perturbations auxquels sont soumis les vrais réseaux de neurones biologiques. C'est la seule justification de l'asynchronisme donnée par Hopfield [HOP82] .

II.2.2. Description du réseau synchrone

Cependant Hopfield lui-même [HOP84] reconnait qu'une mise à jour synchrone conserve les caractéristiques des réseaux asynchrones.
Tous les neurones mettent à jour leur activité en même temps selon une fonction de seuil : ils calculent la somme pondérée de leurs entrées puis prennent la décision simultanément.
Le réseau itère selon la loi d'évolution précédemment décrite (cf formule (I.6)) jusqu'à obtention d'un état stable. Le réseau converge au pas r, lorsque les activités des neurones restent inchangées entre deux itérations successives :

$$\forall\, i \in [1,N] \qquad X_{i,r} = X_{i,r-1} \qquad \text{(I.7)}$$

Les résultats sur la convergence du réseau asynchrone s'étendent aux réseaux synchrones lorsque la matrice est symétrique semi-définie positive. Dans ce cas, la matrice n'a plus besoin d'avoir une diagonale nulle.

Comme le réseau synchrone a les mêmes caractéristiques que le réseau asynchrone, nous ne ferons plus référence, par la suite, qu'au réseau synchrone. Pour utiliser ce réseau comme mémoire auto-associative, plusieurs lois ont été développées. Nous allons en présenter trois.

II.2.3. Loi d'apprentissage de Hebb/Hopfield

La loi de Hebb est présentée en premier afin de respecter la chronologie et parce qu'elle inspira de nombreuses variantes. La modification des poids synaptiques se fait par corrélation. Pour apprendre un nouveau vecteur X, la

modification de la matrice synaptique C se fait selon la formule suivante :

$$\Delta C = (X X^t) / N \text{ avec } C = [0] \text{ avant l'apprentissage} \quad \textbf{(I.8)}$$

Cette loi produit une matrice symétrique ($C_{I,J} = C_{J,I}$). Pour le réseau asynchrone, la diagonale de la matrice est forcée à zéro pour assurer la convergence de la phase de reconnaissance [HOP82]. Hopfield utilisa cette loi dans la formalisation de la dynamique de ses réseaux et de nombreuses applications ont été développées avec cette loi d'apprentissage.

Cependant cette loi a 2 limitations [HOP82], [LIP87] :

- le nombre de prototypes appris et correctement retrouvés est très limité. Si l'on force le réseau à apprendre trop de prototypes, la loi d'apprentissage crée des états stables parasites. Le réseau converge alors vers des états stables non appris. On peut montrer [HOP84] que si le nombre de prototypes appris est inférieur à 15% du nombre de neurones, la loi crée des puits d'attraction suffisamment profonds autour des prototypes pour que le réseau converge vers les états stables appris. Si l'on poursuit l'apprentissage, de nombreux états parasites sont créés. A la limite tous les états deviennent stables.
- la 2ème limitation vient du fait que n'importe quel prototype ne peut être appris. En effet si 2 prototypes sont trop proches, un des 2 prototypes risque d'être instable. Si le réseau effectue une reconnaissance du prototype instable, il converge vers un autre état stable. On peut parler alors de confusion. Pour éviter ce problème, on peut soumettre les prototypes à des procédures d'orthogonalisation avant de les apprendre au réseau. On peut aussi refuser d'apprendre un prototype qui risque d'être confondu avec un autre.

II.2.4. Apprentissage par la projection.

Une des limitations de la loi de Hebb est son incapacité à mémoriser des prototypes linéairement dépendants ou trop proches les uns des autres. La loi

de la projection, au contraire permet d'éviter de tenter une telle mémorisation.

L'apprentissage par la projection a été inspiré par la forte analogie existant entre les réseaux de Hopfield tels qu'on les a décrits et les "verres de Spin". Ces réseaux peuvent alors être étudiés avec les outils de la mécanique statistique [PER85], [PER86 a,b,c,d].

Pour apprendre un nouveau prototype, il faut résoudre l'équation (I.5). A cause de la non linéarité introduite par la fonction de seuil F, cette équation est très complexe à résoudre sous cette forme. La résolution est plus simple en utilisant la forme linéaire de l'équation :

$$\mathbf{X = C \, . \, X} \qquad \textbf{(I.9)}$$

Cette équation doit être vérifiée pour tous les vecteurs prototypes à apprendre. Soit Σ la matrice dont les colonnes sont les prototypes à apprendre, l'équation (I.9) à résoudre devient :

$$\Sigma = \mathbf{C}\,\Sigma \qquad \textbf{(I.10)}$$

Une solution peut être trouvée en calculant la pseudo-inverse :

$$\mathbf{C} = \Sigma\,\Sigma^{I} \quad \textbf{où} \quad \Sigma^{I} = (\Sigma^{t}\Sigma)^{-1}\Sigma^{t} \quad \textbf{est la pseudo inverse de } \Sigma \qquad \textbf{(I.11)}$$

La matrice C peut être calculée itérativement à partir d'une matrice nulle en suivant l'algorithme suivant :

$$\textbf{Soit } y = (X^{t}X - X^{t}CX) \textbf{ alors } \Delta C = \frac{(CX - X)\,(CX - X)^{t}}{y} \qquad \textbf{(I.12)}$$

La mesure "y" sert à quantifier le supplément d'information qu'apporte la mémorisation du vecteur X. Si X est linéairement dépendant ou trop proche des vecteurs déjà appris, "y" est faible et le vecteur X est rejeté : la mémorisation n'a pas lieu.

Dans le cas particulier où les prototypes sont orthogonaux, la relation (I.11) se réduit exactement à la loi de Hebb (I.8). Mais, la loi de la projection permet la mémorisation de prototypes non orthogonaux : cette loi est une généralisation de la loi de Hebb. D'autre part, la loi de la projection crée moins d'états parasites que la loi de Hebb et permet un apprentissage de meilleure qualité.

II.2.5. Delta-projection

Difficile à calculer, la loi de la projection peut être approchée de plusieurs façons [WEI88]. La loi de la delta-projection utilise l'approximation suivante :

$$\Delta C = (X - CX)X^t / N \qquad \textbf{(I.13)}$$

Cette formule (I.13) est intéressante car il a été montré que la répétition de cet apprentissage conduit à la création d'une matrice similaire à la matrice de la projection.

II.2.6. Comparaison des trois lois d'apprentissage

Les performances d'une loi d'apprentissage sont caractérisées par le nombre de prototypes appris et la capacité à corriger les erreurs.

Afin d'évaluer les performances relatives des trois lois d'apprentissage présentées, nous avons procédé à des simulations.

Pour un réseau de 128 neurones, un jeu de vecteurs tirés aléatoirement est utilisé pour l'apprentissage selon les trois lois : lois de Hebb, de la projection et de la delta-projection. Les résultats présentés sont établis sur 500 reconnaissances de vecteurs prototypes perturbés. Ces perturbations se font, de façon classique [LIP87] [STI87], en inversant chaque bit du vecteur avec une probabilité que l'on définit.

La première simulation traduit l'évolution des performances en fonction du nombre de prototypes appris : ces performances sont caractérisées par le pourcentage de réussite de la phase de reconnaissance sur les prototypes perturbés avec une probabilité de 0.1 pour chaque bit.

On a obtenu les résultats présentés par la figure I.5. Dans un premier temps, on remarque que les performances de la loi de Hebb sont comparables à celles annoncées en [HOP84]. Lorsque le nombre de vecteurs appris dépasse 15% du nombre de neurones, le réseau ne corrige plus les erreurs et n'assure donc plus la fonction de mémoire associative.

Par contre, la projection et la delta-projection sont capables de mémoriser deux fois plus de prototypes. Ces deux algorithmes ont des performances identiques, les faibles différences qui apparaissent sur la courbe sont induites par la génération aléatoire de la perturbation.

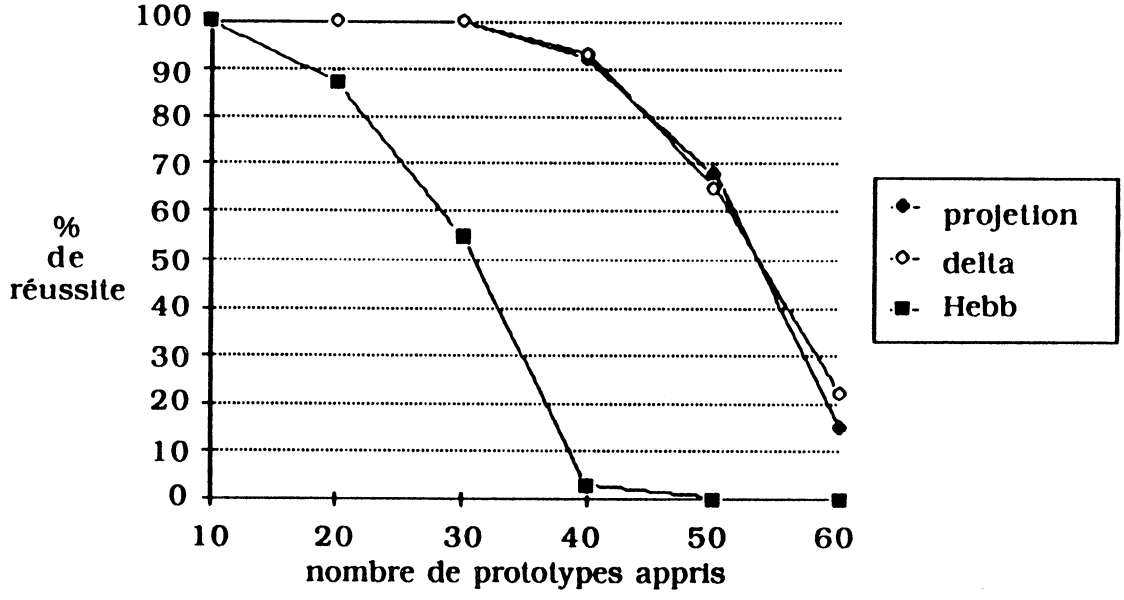


Figure I.5 Performances pour une perturbation avec une probabilité de 0.1 en fonction du nombre de prototypes appris sur 128 neurones

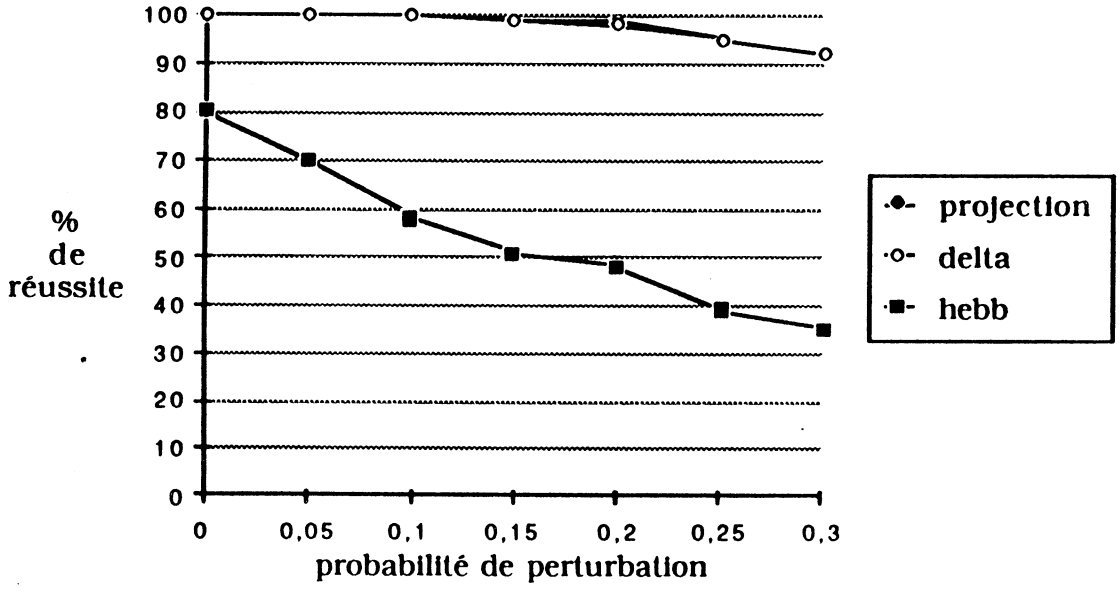


Figure I.6 Evolution des performances pour 25 prototypes appris en fonction de la probabilité de la perturbation sur 128 neurones

La deuxième simulation montre la capacité de correction d'erreurs des différentes lois. 25 vecteurs prototypes ont été appris. Les résultats de cette simulation sont rassemblés sous forme de courbes dans la figure I.6. Les conclusions tirées de cette simulation sont comparables à celles de la première.

La loi de Hebb est la loi d'apprentissage la moins performante et les lois de la projection et de la delta-projection offrent de meilleures performances de correction d'erreurs.

II.3. La machine de Boltzmann

II.3.1. Principe

La machine de Boltzmann est une généralisation des réseaux de Hopfield. Elle consiste à modifier la loi de mise à jour des neurones. Cette modification permet d'introduire la notion de recuit simulé dans les réseaux de neurones [HIN85], [FAH87], [HIN86]. Chaque neurone va modifier son activité selon le processus stochastique suivant :

$$X_i = 1 \text{ avec la probabilité } P = \frac{1}{(1 + e^{-\Delta E_i / T})} \quad \text{sinon } X_i = 0 \qquad \text{(I.14)}$$

$$\text{où } \Delta E_i = \sum_{j=1}^{N} C_{i,j}. X_j.$$

T est appelé température par analogie aux systèmes physiques.

Appliquer cette loi de façon répétitive permet d'assurer que le réseau atteindra un "équilibre thermique". Une fois cet équilibre atteint, les neurones changent encore d'état. Mais la probabilité de se trouver dans n'importe quel état reste constante. Elle obéit à une loi de distribution de Boltzmann dans laquelle le rapport entre la probabilité d'être dans l'état A et celle d'être dans l'état B ne dépend que de la différence d'énergie (E_A - E_B) :

$$\frac{P_A}{P_B} = e^{-(E_A - E_B) / T} \qquad \text{(I.15)}$$

A haute température, le réseau atteint rapidement un équilibre thermique où les états de faible énergie ne sont pas plus probables que les états de haute énergie : tous les états sont équiprobables. Par contre à basse température, les états à faible énergie sont nettement favorisés mais l'équilibre est atteint lentement. La meilleure manière d'obtenir un équilibre final à basse température est de débuter à haute température puis de réduire progressivement la température.
Cette méthode est connue sous le nom de recuit simulé, par analogie à la physique des solides. En effet, la nécessité d'abaisser lentement la température d'un liquide est caractéristique d'un processus de recuit : cette procédure conduit à un état cristallin ordonné qui correspond à un minimum global de l'énergie, alors qu'un refroidissement brutal donne naissance à un solide amorphe ou polycristallin qui correspond à un minimum local de l'énergie [SIA85], [SIA86]. Cette méthode permet donc d'atteindre les minima énergétiques avec une forte probabilité.

Pour utiliser la machine de Boltzmann comme mémoire associative, la clé de recherche définit les états initiaux des neurones. Ces neurones sont gelés c'est à dire que leur état n'est pas altéré lors de la reconnaissance. Puis le processus de reconnaissance est mis en œuvre selon la loi stochastique (I.14) en commençant à haute température et en la réduisant progressivement.

II.3.2. Loi d'apprentissage

La différence entre les réseaux de Hopfield et la machine de Boltzmann intervient au niveau de la phase de reconnaissance. Elle ne concerne pas forcément l'étape d'apprentissage.
Les lois présentées pour les réseaux de Hopfield utilisés comme mémoires associatives, peuvent donc être utilisées pour la machine de Boltzmann.
Cependant, Hinton et Sejnowski [HIN85] ont établi un mode d'apprentissage

spécialement dédié à cette machine. Il est relativement complexe car il nécessite une étude statistique de la dynamique du réseau à chaque étape du processus d'apprentissage.

III. PROBLEMES D'OPTIMISATION COMBINATOIRE

Nous venons d'étudier l'utilisation d'un réseau de neurones comme mémoire associative, rendue possible par la création d'états stables correspondant aux prototypes à apprendre. Ces états stables correspondent à des minima locaux de l'énergie dans l'espace des états. Apprendre c'est donc créer ces minima et reconnaître c'est se déplacer dans l'espace des états en minimisant l'énergie.
De ce point de vue, les réseaux de neurones semblent de bons candidats pour résoudre les problèmes d'optimisation consistant à minimiser une fonction de coût. Pour cela, il faut définir un réseau ayant comme fonction d'énergie la fonction de coût à minimiser.

III.1. Le problème du voyageur de commerce

Le problème du voyageur de commerce est un classique des problèmes d'optimisation. De par la simplicité de sa description et sa définition mathématique bien connue, ce problème est idéal pour étudier les capacités et les performances des réseaux de neurones pour résoudre des problèmes d'optimisation difficiles.
Soient un ensemble de villes (A,B,C...) et la table des distances qui les séparent 2 à 2, il s'agit de trouver le circuit fermé, le plus court possible qui passe une fois et une seule dans chaque ville et revient à son point de départ. Une tournée est définie par une séquence de villes (ex. BAGE...DB) et la longueur d'une tournée est définie par $d = d_{BA} + d_{AG} + d_{GE} + ... + d_{DB}$. Trouver la solution optimale de ce problème est très complexe. Nous sommes en présence d'un problème NP complet. A l'heure actuelle, tous les algorithmes connus

résolvant ce problème sur un ordinateur séquentiel, voient leur temps de calcul croître de façon exponentielle avec le nombre de villes.

III.2. Représentation du problème

A partir des distances entre villes, le réseau doit fournir une liste ordonnée des villes traversées. Une représentation du problème sur réseau de neurones a été définie [HOP85] : la position de chacune des P villes dans la tournée est représentée sur P neurones par un codage " 1 parmi P". Pour 5 villes, les activités des neurones associés à la deuxième ville de la tournée seront 0 1 0 0 0.

Pour P villes, une tournée complète est représentée par P ensembles de P neurones. Il faut donc une matrice PxP de neurones. La tournée BCADE sera représentée par :

	1	2	3	4	5
A	0	0	1	0	0
B	1	0	0	0	0
C	0	1	0	0	0
D	0	0	0	1	0
E	0	0	0	0	1

Figure I.7 Représentation d'une solution du voyageur de commerce

Pour P villes, il existe P! tournées possibles. Cependant à partir d'une tournée donnée, P tournées identiques peuvent être engendrées en changeant la ville de départ et deux fois plus si l'on change le sens de parcours. Par exemple BCADE est équivalent à EBCAD, CADEB et BEDAC. Il n'y a donc que (P!) / (2P) chemins réellement différents.

III.3. "Programmation" du problème

Il s'agit de trouver des coefficients synaptiques qui définissent une fonction d'énergie équivalente à la fonction de coût à minimiser (ici la longueur du chemin). Mais surtout, il faut s'assurer que le réseau convergera vers une solution valide. Pour définir une loi permettant de produire les coefficients amenant à une "bonne" solution, il faut analyser la méthode de représentation. Un état final du réseau de P^2 neurones est considéré comme valide si :

- deux villes différentes ne peuvent pas être visitées en même temps ⇒ un seul neurone actif par ligne,
- une ville n'intervient qu'une seule fois par tournée ⇒ un seul neurone actif par colonne,
- les P villes participent à la tournée ⇒ P neurones actifs dans le réseau.

Dans un réseau de neurones standard, l'état du neurone i est représenté par la variable X_i. Ici, nous allons utiliser une notation différente. Les P neurones affectés à une ville Y seront représentés par $X_{Y,j}$ avec $j \in [1,P]$.

La loi d'apprentissage définie par Hopfield et Tank [HOP85] découle des contraintes imposées pour la validité des réponses et des données du problème. Pour calculer les poids synaptiques, on évalue quatre termes :

$$\begin{aligned} C_{Yi,Zj} = & - A\,\delta_{YZ}\,(1 - \delta_{ij}) && \text{"inhibition latérale par ligne",} \\ & - B\,\delta_{ij}\,(1 - \delta_{YZ}) && \text{"inhibition latérale par colonne",} \\ & - C && \text{"inhibition globale",} \\ & - D\,d_{YZ}\,(\delta_{j,i+1} + \delta_{j,i-1}) && \text{"données spécifiques du problème".} \end{aligned} \quad \textbf{(I.16)}$$

Les trois premiers termes de la formule (I.16) assurent que toutes les tournées valides ont une énergie faible. Le dernier apporte les informations spécifiques au problème traité : les distances entre villes.

Avec une telle loi, l'énergie totale d'un état du réseau correspond à la longueur du chemin et les états représentant le plus court chemin sont les états d'énergie faible.

III.4. Résultats

Hopfield a traité le problème du voyageur de commerce pour 10 villes. Il a utilisé un réseau de Hopfield asynchrone de 100 neurones de type Mac Culloch et Pitt. On peut obtenir des résultats similaires avec une machine de Boltzmann.

Une fois les coefficients calculés par la loi définie ci-dessus, Hopfield simula le fonctionnement de ce réseau pour 20 états initiaux différents. Le réseau convergea 16 fois vers une solution valide (80% de réussite). 50% des essais donnèrent comme résultat 1 des 2 plus courts chemins.

D'autres problèmes d'optimisation ont été étudiés comme l'allocation de tâches [TAN87], la recherche dans les arbres binaires [SAY86] et d'autres problèmes nécessitant l'optimisation de fonctions [GRZ86], [JEF86].

Hopfield et Tank ont aussi poursuivi leurs investigations sur le problème du voyageur de commerce avec 30 villes (900 neurones). Comparés aux résultats obtenus pour le problème des 10 villes, les résultats publiés pour cet essai sont moins probants. Alors qu'un algorithme classique comme celui de Lin-Keninghan donne un chemin de longueur 4.26, le réseau de Hopfield fournit fréquemment une réponse inférieure à 7 et occasionnellement inférieure à 6. Cependant, sur les 10^{30} chemins possibles, le réseau donne ses réponses parmi les 10^{8} meilleurs chemins soit un facteur de sélection de 10^{23}.

La principale difficulté réside dans l'ajustement des paramètres de la loi d'apprentissage. En effet, il faut trouver un compromis entre favoriser le plus court chemin et n'obtenir que des tournées valides. Si on donne plus d'importance aux paramètres A, B et C de la formule (I.16), le réseau délivrera, à coup sûr, des tournées valides mais les solutions obtenues seront médiocres. Par contre si le paramètre D est prépondérant, les tournées valides seront très rares mais de longueur proche de l'optimal.

IV. LES CLASSIFIEURS - MEMOIRES HETERO-ASSOCIATIVES

IV.1. Réseau de Hopfield

Nous venons de présenter dans les chapitres précédents, l'utilisation des réseaux de Hopfield pour des tâches d'optimisation combinatoire et de mémoires auto-associatives. Ce type de réseaux permet aussi de faire de la classification.

IV.1.1. Principe

L'approche retenue par Personnaz consiste à transformer en mémoire hétéro-associative, une mémoire auto-associative sur un réseau de Hopfield. Un champ supplémentaire est ajouté aux prototypes à apprendre. Ce champ représente la classe du prototype. L'étude présentée [PER85], [PER86b] porte sur la reconnaissance de chiffres décimaux manuscrits. Les réseaux de neurones étant particulièrement adaptés pour traiter des informations binaires, les chiffres manuscrits sont dans un premier temps digitalisés sous la forme d'une matrice de pixels 32x32.

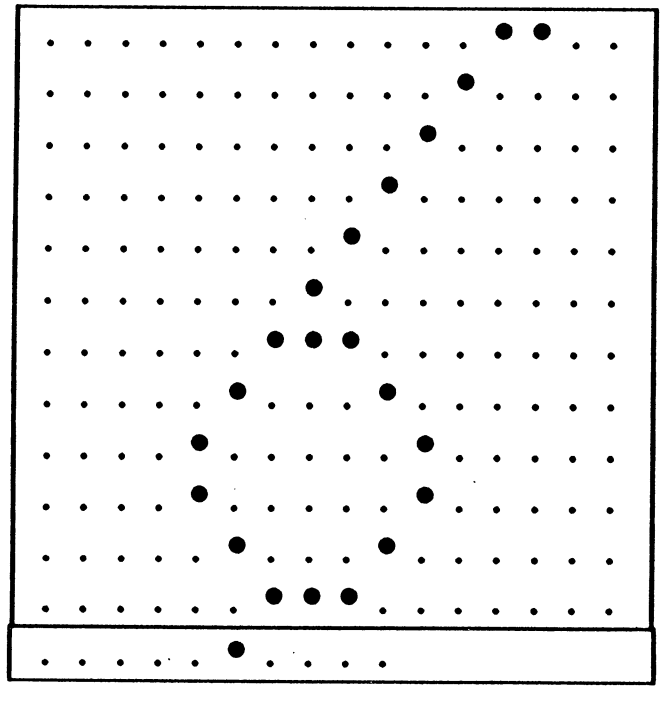


Figure I.8 Représentation du chiffre 6.

Pour l'apprentissage de l'ensemble des prototypes, un champ est ajouté pour coder le chiffre représenté par la matrice de pixels comme le montre la figure I.8. La dernière ligne représente le code "1 parmi n" du chiffre appris.
Une fois l'apprentissage effectué, des chiffres digitalisés sont présentés au réseau avec un champ de codage vierge. Le réseau converge naturellement vers la forme du chiffre le plus proche et complète le champ de codage si l'état stable atteint est un prototype appris.

IV.1.2. Résultats

L'apprentissage consiste à mémoriser les écritures les plus courantes des chiffres. 44 chiffres manuscrits ont été mémorisés avec leur champ d'identification. Une fois l'apprentissage effectué par la loi de la projection, 250 chiffres ont été reconnus par le réseau. Les représentations présentées au réseau n'étaient pas des prototypes et le champ de codage était vierge.
Dans 80% des cas, le réseau a complété correctement le champ de classification. Cependant le réseau n'est pas infaillible car il commet une erreur dans 10% des cas. Dans les 10% restant, le réseau donne une réponse incohérente : le champ de classification après convergence ne correspond pas à une classe apprise (par exemple champ vide ou comprenant plusieurs bits à 1) [PER85], [PER86b].
Des résultats présentés, on peut remarquer :

- que certains états parasites servent d'état "poubelle" en produisant un code non valide.
- que les états stables atteints ne sont pas tous des prototypes appris. Pourtant le champ de classification est correct.

Ces résultats sont intéressants car ils mettent en évidence les capacités de généralisation du réseau et l'utilisation des états stables parasites créés par le réseau.

IV.2. Réseau de Hamming

Ce réseau est ainsi appelé car il fait appel à la distance de Hamming pour définir la classe d'un élément. Ce réseau calcule la distance de Hamming entre l'exemplaire traité et le représentant de chaque classe. Il choisit la classe qui minimise cette distance.

Implanter une telle fonctionnalité nécessite deux niveaux successifs. Chaque niveau assure un rôle bien défini [LIP87] :

- le 1er niveau calcule la distance de Hamming (d) entre l'entrée à N composantes et chacune des M classes. Elle fournit au niveau suivant N-d,
- le 2ème niveau extrait le maximum de N - d.

Dans le premier niveau, les représentants de chaque classe sont mémorisés dans les poids : l'entrée i est connectée à la cellule interne j par le poids synaptique suivant :

$$C = \frac{1}{2}\Sigma^t \quad \textbf{où } \Sigma \textbf{ est la matrice des prototypes} \qquad \textbf{(I.17)}$$

Avec des neurones du type logique et un seuil θ de N/2, le premier niveau évalue bien la différence entre N et la distance de Hamming d.

Le deuxième niveau doit extraire le maximum des activités des neurones du premier niveau. Deux solutions sont possibles :

- un réseau itératif : c'est un réseau de Hopfield (cf figure I.4) avec des neurones du type logique et un seuil θ nul. Les poids de ce réseau sont fixes :

  Si i=j alors $C_{i,j} = 1$ **Sinon** $- \varepsilon$ **(I.18)**

  ε doit être inférieur à $\frac{1}{M}$ pour que le réseau converge [LIP87].

- un réseau combinatoire MAXNET : (cf figure I.9) c'est un réseau composé de $Log_2(N)$ couches avec des neurones du type logique (carrés) et du type Mac Culloch et Pitt (rond). Y_i sera actif si X_i est la valeur maximale des

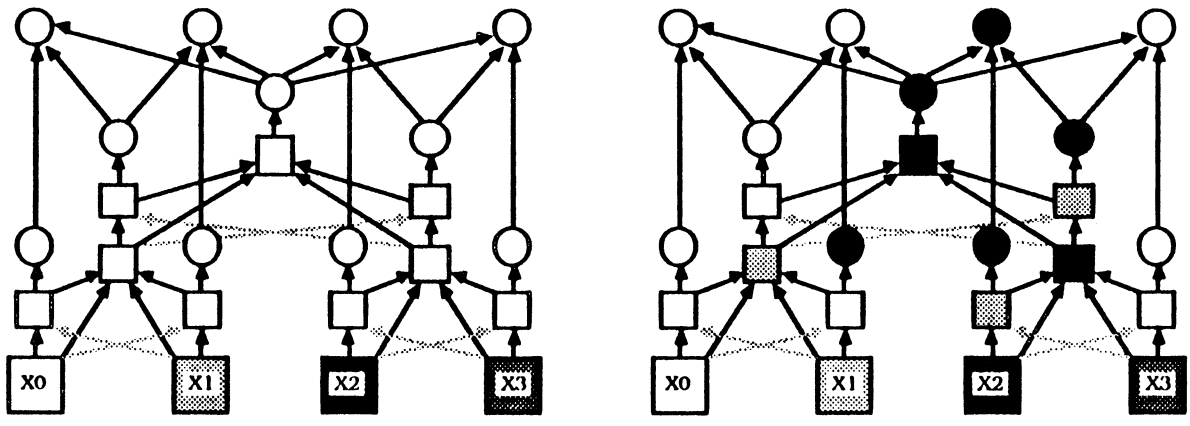
entrées (cf figure I.9). Les poids sont fixes :

- flèches gris clair : - 1,
- flèches noires latérales : + 0.5,
- flèches noires verticales : + 1.

Sur la figure l'activité des neurones est indiquée par la couleur : plus un neurone est actif, plus sa teinte est foncée.

Le réseau de Hamming nécessite moins de connexions qu'un réseau de Hopfield équivalent. L'apprentissage est trivial. Par contre, ce réseau nécessite des neurones de types différents.

Remarque : les réseaux permettant d'extraire le maximum sont d'une grande utilité, particulièrement dans le cas de l'apprentissage compétitif (cf IV.5.1).



état initial état final

Figure I.9 Evolution du Maxnet

IV.3. Perceptron monocouche

IV.3.1. Principe

Ce modèle fut parmi les premiers développés et il suscita beaucoup d'intérêt, lors de sa découverte, pour ses capacités de classifieur.

Le Perceptron utilise un réseau à une couche sans boucle de rétro-action. Les

neurones sont du type Mac Culloch et Pitt avec des valeurs +1 ou -1. Nous allons expliquer le principe de cette machine sur l'exemple simple d'un classifieur à 2 classes.

Dans ce cas, le Perceptron ne comprend qu'un seul neurone. Si les entrées appartiennent à la classe A (respectivement B) le nœud de sortie prendra la valeur +1 (respectivement -1). Le réseau définit donc des zones de décision. Une zone correspond à l'ensemble des variables d'entrée déterminées par le réseau comme appartenant à une même classe. Le comportement du réseau peut alors être illustré par la représentation graphique des zones de décision dans l'espace multidimensionnel des variables d'entrées.

Le Perceptron partage cet espace en 2 zones de décision séparées par un hyperplan. Des lois d'apprentissage vont donc tenter de définir cet hyperplan.

IV.3.2. Loi d'apprentissage de Rosenblatt

Au départ, les poids synaptiques sont initialisés avec une faible valeur aléatoire. Les prototypes aux composantes réelles sont présentés au réseau qui fournit une réponse. Si cette réponse est identique à la réponse attendue Y, les poids synaptiques sont inchangés et on passe à l'apprentissage du prototype suivant. Par contre si la réponse donnée diffère de la réponse attendue Y, les poids synaptiques sont modifiés comme suit :

$$\Delta C = \eta (Y X^t) \qquad \textbf{(I.19)}$$

Cet apprentissage est du type supervisé. Le gain η permet d'apprendre un nouveau vecteur sans "oublier" les vecteurs déjà appris. Ce paramètre doit être ajusté afin de permettre une mise à jour significative des poids sans pour autant perdre les informations concernant les entrées précédentes.

Cet algorithme permet d'obtenir un classifieur si les entrées sont séparables par un hyperplan. Dans le cas contraire l'apprentissage ne va pas converger car la limite entre les deux zones de décision va perpétuellement osciller.

IV.3.3. Loi d'apprentissage de Widrow et Hoff - LMS

L'oscillation peut être limitée si le modèle de Mac Culloch et Pitt est abandonné au profit du modèle logique [LIP87]. De plus la loi (I.19) est modifiée pour utiliser un critère quadratique d'erreur. Cette loi d'apprentissage quasi-supervisée est décrite ci-dessous :

$$\Delta C = \eta (CX - Y)X^t \qquad \textbf{(I.20)}$$

Cet algorithme développé par Widrow et Hoff [WID88] est aussi appelé LMS (LMS = Least Mean Squares = moindres carrés). Il repose sur la minimisation d'un critère quadratique.

A l'origine, cet algorithme d'apprentissage a été développé pour la machine Adaline de Bernard Widrow. Adaline (ADAptive LInear NEuron) est très proche du modèle du Perceptron.

IV.3.4. Extensions et limitations

Le modèle du Perceptron à deux classes peut être étendu à M classes en utilisant M structures de base. Les classes sont alors codées en 1 parmi M par les M neurones de sortie.

De même, la Madaline (Many Adalines) est construite à base de plusieurs Adalines connectées aux même entrées par des poids différents [WID88].

Cependant ni l'algorithme de Rosenblatt ni l'algorithme LMS ne permettent de résoudre tous les problèmes de classification. Seules les classes linéairement séparables, c'est à dire qui peuvent être séparées par un hyperplan, sont discriminées par de telles méthodes. Ces limitations sont intrinsèques aux séparateurs linéaires. Le OU exclusif est un célèbre exemple de ce genre de difficultés, et il servit à Minski et Papert pour illustrer les limitations du Perceptron.

Sur un Perceptron à 2 classes, on veut classer 2 entrées binaires X et Y selon la fonction du OU exclusif. Soit la classe A correspondant à ($X \neq Y$) et la classe

B pour (X = Y), la représentation graphique de la figure I.10 montre bien qu'un hyperplan (une droite dans le plan) ne peut définir des zones de décision (hachures) correspondant aux classes A et B.

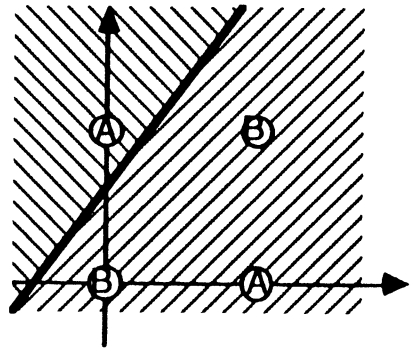


Figure I.10 Problème du OU exclusif

IV.4. Réseaux multicouche

IV.4.1. Principe

L'idée du réseau multicouche est très simple. Il s'agit de connecter plusieurs Perceptrons les uns à la suite des autres. Le nombre de nœuds par couche est variable et le type des neurones peut éventuellement différer suivant la couche.

Déjà dans les années 60, des Perceptrons à 3 couches avaient été étudiés. Mais, malgré les nombreuses tentatives, aucune loi d'apprentissage efficace n'avait été mise au point. D'après [HIN87], ces tentatives s'étaient concentrées sur la définition du pré-traitement à effectuer par les couches intermédiaires pour résoudre un problème. Elles n'avaient pas considéré les couches intermédiaires comme de véritables couches cachées. Les algorithmes étaient trop dirigés. L'algorithme de la rétro-propagation du gradient a été le premier à considérer véritablement les couches intermédiaires comme des couches cachées.

Cet algorithme d'apprentissage quasi-supervisé a été conçu pour des réseaux combinatoires structurés en plusieurs couches successives. La première reçoit les stimuli d'entrée et la dernière fournit la sortie (réponse) du système. Les couches intermédiaires sont appelées aussi couches cachées car elles n'ont pas de relation directe avec l'extérieur. On obtient la structure suivante :

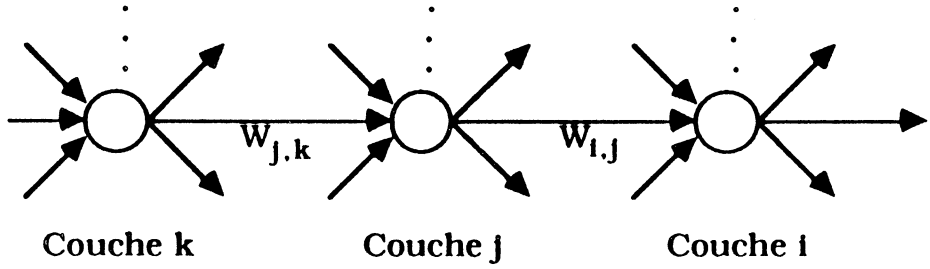


Figure I.11 Réseau multicouche

Apprendre sur un tel réseau n'est pas chose aisée car, lors de l'apprentissage, les activités des couches cachées sont inconnues. L'algorithme de rétro-propagation a été la première tentative d'apprentissage sur un tel réseau à être couronnée de succès.

IV.4.2. Rétro-propagation du gradient

Cet algorithme, étudié indépendamment par [LEC87] et [RUM86b], est une généralisation de l'algorithme LMS (I.20) pour les réseaux multicouche comprenant des couches cachées.

Au départ, les poids synaptiques sont initialisés avec une faible valeur aléatoire. Un prototype aux composantes réelles est présenté au réseau sur la première couche. Cette entrée est propagée à travers toutes les couches par la loi de propagation suivante :

$$X_i = F\left(\sum_j C_{i,j} \cdot X_j \right) \qquad \text{(I.21)}$$

où F est la fonction sigmoïde (I.8).

L'ensemble des activités des neurones de la dernière couche fournit la réponse S du système. La sortie désirée Y est alors présentée aux cellules de la couche de sortie. L'algorithme calcule les dérivées partielles de l'erreur en sortie par rapport à tous les poids du réseau. Ensuite, le gradient de la fonction d'erreur est calculé à partir des gradients γ_i évalués de couche en couche à partir de la couche de sortie en utilisant les poids à l'envers : rétro-propagation du gradient. Le calcul des gradients se fait selon la formule (I.22).

$$\gamma_i = 2\,F'\left(\sum_j C_{i,j} \cdot X_j\right) \cdot (Y_i - S_i) \quad \textbf{pour les cellules de la couche de sortie,}$$

(I.22)

$$\gamma_j = F'\left(\sum_k C_{j,k} \cdot X_k\right) \cdot \left(\sum_i C_{i,j} \cdot \gamma_i\right) \quad \textbf{pour les cellules des autres couches.}$$

Ensuite les poids sont modifiés sur toutes les couches en utilisant les gradients calculés par (I.22) selon la formule (I.23) où η est le gain.

$$\Delta C_{i,j} = \eta\,\gamma_i\,X_j \qquad \textbf{(I.23)}$$

IV.4.3. Résultats

L'avantage d'un tel apprentissage est sa capacité à résoudre des problèmes non linéairement séparables. On peut résoudre le plus petit problème de ce type : le OU exclusif (Voir figure I.12).

D'autres problèmes plus complexes ont été étudiés comme la synthèse de fonctions booléennes, le diagnostic médical, le jeu du Tic-Tac-Toe [LEC87], etc...

Les principales difficultés restent la définition de la structure du réseau ainsi que l'ajustement des paramètres de l'apprentissage.

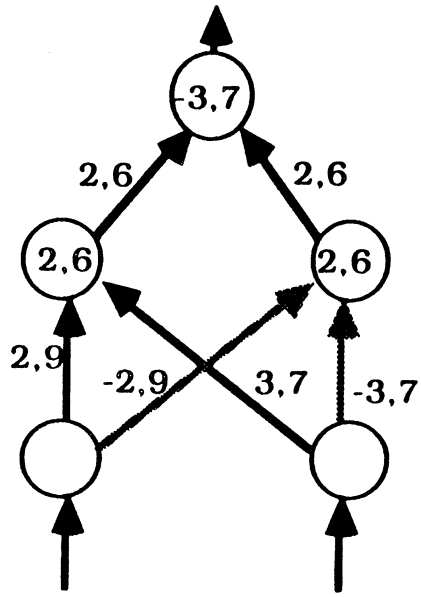


Figure I.12 Une solution du problème du OU exclusif

Remarque : l'algorithme de la rétro-propagation du gradient sur réseaux multicouche permet aussi d'obtenir une mémoire auto-associative. Les résultats obtenus ne sont pas meilleurs que ceux obtenus par les structures monocouche, excepté dans le cas précis où la loi de perturbation est connue. L'apprentissage peut alors être adapté à cette perturbation et donner de meilleur résultats que les réseaux à une couche [LEC87].

IV.5. Auto-organisation

Presque tous les algorithmes déjà présentés sont du type supervisé. L'algorithme LMS et sa généralisation aux réseaux multicouche (rétro-propagation du gradient) se distinguent par un apprentissage quasi-supervisé. Ces apprentissages moins dirigés permettent la construction des précieuses représentations internes, dans les couches cachées. Celles-ci sont responsables du succès de l'algorithme mais la présence d'un superviseur externe est toujours nécessaire pour permettre l'évaluation numérique de la qualité de la réponse donnée par le réseau.

La suppression de ce superviseur nous amène à l'auto-organisation où l'apprentissage n'est plus supervisé que par les entrées.

Il n'est pas surprenant de faire intervenir la notion d'auto-organisation dans les réseaux de neurones. En effet, le cerveau biologique peut adapter ses connaissances et ses caractéristiques en fonction de l'environnement dans lequel il évolue. La structure générale du cerveau est déterminée génétiquement. Mais, à un niveau inférieur, les structures s'organisent en fonction de l'environnement. On suppose que son adaptation s'effectue par modification des connexions synaptiques inter-neurones [AMA83].
Il ne s'agit pas ici de reprendre la bataille "inné/acquis" mais de mettre en évidence le rôle que joue l'auto-organisation dans les modifications des structures fines de la connectique du cerveau.

IV.5.1. Apprentissage compétitif

L'apprentissage compétitif s'utilise dans les réseaux multicouche. Outre les connexions des réseaux multicouche classiques (voir figure I.11), les éléments d'une même couche sont localement reliés par groupe. Dans chaque groupe, les éléments sont entièrement reliés par des connexions inhibitrices. Il y a donc une compétition entre cellules d'un même groupe pour devenir actives. Plus une cellule répond à un stimulus en étant active, plus elle tente de rendre inactives les autres cellules de son groupe. Seul le vainqueur de chaque groupe peut modifier ses poids synaptiques [RUM86a].
Avec un apprentissage compétitif, il va donc y avoir une véritable lutte entre les cellules pour avoir le droit d'adapter leurs poids synaptiques [HIN87].
Cet apprentissage a été utilisé sous différentes formes [RUM86a] comme nous allons le voir ci-après.

IV.5.2. Loi de Hebb non supervisée

La loi de Hebb non supervisée a été utilisée par Fukushima pour le Cognitron [FUK75], [MIY86].

Le Cognitron est un réseau combinatoire à plusieurs couches. Le modèle des neurones est assez complexe et diffère sensiblement du modèle générique présenté en début de ce chapitre.
Dans le Cognitron, par analogie avec le cerveau biologique, les connexions synaptiques se modifient en fonction de l'expérience acquise.
Cette adaptation se fait par renforcement des connexions synaptiques sur un critère précis. L'idée clé de cette adaptation est de modifier un poids synaptique en fonction de la corrélation existant entre les activités pré et post-synaptiques : la connexion synaptique entre un neurone i donné et un neurone j de la couche suivante sera renforcée si j est le plus actif de son entourage.

Cette version non supervisée de la loi de Hebb appliquée aux réseaux multicouche permet d'obtenir certaines analogies avec les propriétés du cortex visuel [AMA83]. Linsker adopta une démarche similaire [LIN86], [LIN88].

Cette adaptation non supervisée est une illustration de l'apprentissage compétitif. Après plusieurs présentations des prototypes, le réseau s'auto-organise sans l'aide d'un superviseur.

IV.5.3. Classifieur de Carpenter/Grossberg

Ce classifieur va former des classes au fur et à mesure de son apprentissage sans superviseur. Le premier prototype présenté va être le représentant de la première classe. Le deuxième prototype est comparé avec le représentant de la première classe. Si la distance qui les sépare est inférieure à un seuil prédéfini, la nouvelle entrée est associée à la première classe, sinon une nouvelle classe est créée. Le deuxième prototype devient alors le représentant de la classe qu'il a créée [GRO76], [CAR86].

L'augmentation du nombre de classes pendant l'apprentissage dépend de deux facteurs [LIP87] :

- la mesure utilisée pour évaluer la distance entre un nouveau prototype et les représentants des classes déjà existantes,
- le seuil qui permet de décider de la création d'une nouvelle classe.

Le seuil doit être choisi correctement afin d'avoir un apprentissage correct sans augmenter de façon considérable le nombre de classes créées. Une fois de plus, la difficulté vient de l'ajustement des paramètres.

IV.5.4. Auto-organisation de Kohonen

Kohonen a essayé de comprendre les raisons de l'existence des cartes d'activités que l'on trouve dans le cortex. Par exemple, en Electro-Encéphalo-Graphie : pourquoi une région du cerveau répond-elle préférentiellement à certains stimuli.

Il a réussi à mimer ce phénomène en considérant un réseau dont les neurones ont des connexions d'excitation et d'inhibition fixes entre eux, et plastiques avec l'extérieur. Lorsqu'on envoie des stimuli externes, il se forme dans ce réseau des réponses préférentielles à ces stimuli.

Il s'agit bien d'un phénomène d'auto-organisation car on ne peut prévoir quelle région du réseau répondra préférentiellement à un stimulus plutôt qu'à un autre. Il s'agit en fait de l'auto-organisation d'un système instable qui évolue dans le sens du renforcement de différences de comportement initialement aléatoires.

Fondé sur ce principe, un réseau appliqué à la reconnaissance de la parole a été réalisé sous forme de carte coprocesseur. Ce réseau contient une couche où les neurones de sortie sont disposés sur un maillage carré et sont localement interconnectés. Un ensemble d'entrées est présenté au réseau et les poids s'auto-adaptent selon une loi simple. La loi de modification présentée

dans les nombreuses publications de Kohonen permet de créer les cartes d'activités préférentielles en fonction des entrées [KOH88].

Contrairement au classifieur de Carpenter/Grossberg, le nombre de classes est limité: la présence de bruit ne risque pas de provoquer une explosion du nombre de classes. Ce réseau est donc plus résistant au bruit [LIP87].

V. CONCLUSION

Avant de conclure, la table présentée figure I.13 permet de résumer les différents modèles présentés et leurs applications.

	Mémoire auto-associative	**Optimisation**	**Classifieur**
Hopfield	Hebb Projection Delta-projection	Hopfield&Tank	Hebb Projection Delta-projection
Multi-couche	Rétro Propagation		Rétro Propagation
Auto organisation			Compétitif

Figure I.13 Récapitulation des lois d'apprentissage en fonction des applications et du type de réseau

D'un point de vue conceptuel, l'auto-organisation est très attrayante. C'est un des principes fondamentaux du cerveau biologique qui sert de base à ce modèle. Malheureusement, l'apprentissage sur ce type de réseaux artificiels est souvent long et délicat. D'une part, l'apprentissage dépend fortement de l'état initial du réseau. D'autre part, le succès de l'adaptation est conditionné par l'ajustement des paramètres. Par exemple, le choix de la mesure et du seuil qui servent de critères d'évaluation est décisif. La principale difficulté est inhérente à l'auto-organisation. Elle provient de l'absence de superviseur. Il est difficile de contrôler ce qui est appris et par conséquent difficile de l'exploiter [LEC87].

La rétro-propagation du gradient est la première loi efficace adaptée aux réseaux multicouche. Elle a permis le renouveau de ce type de réseau et elle lève la principale limitation liée aux réseaux monocouche : la résolution de problèmes non linéairement séparables.
Cependant certaines difficultés demeurent. En pratique, la plus sérieuse réside dans la lenteur de convergence de l'apprentissage [HIN87]. L'apprentissage du problème trivial du OU exclusif nécessite au minimum 50 présentations des 4 configurations. La vitesse de convergence dépend de nombreux facteurs [LEC87] :

- la configuration initiale des poids,
- l'ordre de présentation des vecteurs,
- les paramètres de la fonction de seuil,
- le gain η ...

Cet inconvénient devient rédhibitoire pour des applications grandeur nature où le nombre de cellules est conséquent. De plus, l'ajustement des paramètres est d'autant plus délicat que l'application est importante car le temps de simulation croît proportionnellement.

D'autre part, un certain flou persiste dans le choix de la structure du réseau à adopter pour une application donnée. Le nombre de couches ainsi que le nombre d'éléments par couche restent difficiles à optimiser en fonction de la tâche à réaliser.
Enfin, tous les mécanismes du processus d'apprentissage ne sont pas encore bien appréhendés. De nombreuses études sont en cours afin d'améliorer la formalisation et la compréhension de cette approche.

Par contre, la théorie des réseaux de Hopfield est parfaitement maîtrisée. Les réseaux de Hopfield demeurent un outil privilégié pour les applications neuroniques. Les qualités ainsi que les limitations de ces réseaux itératifs sont très bien connues. Des physiciens ont parfaitement établi les principes de ces réseaux à l'aide des outils de la physique statistique par analogie avec la théorie des "verres de Spin" [HOP82], [PER85], [PER86a]. D'autre part, des études menées par des mathématiciens [CAS88], [COT88] ont permis de formaliser plus complètement ce type de réseaux. Les performances des procédures d'apprentissage sont bien connues et la convergence de la phase de reconnaissance a été prouvée. Ainsi, le réseau de Hopfield est bien établi et universellement reconnu.
Le modèle de neurone utilisé pour ce réseau est le plus simple : modèle de Mac Culloch et Pitt. Ce modèle présente donc une bonne intégrabilité.
Le champ d'application de ces réseaux est vaste car il couvre la résolution de problèmes d'optimisation, les mémoires auto-associatives et la classification.
Les problèmes d'optimisation sont difficiles à résoudre car ce sont des problèmes NP complets. Dans la résolution de ces problèmes, les algorithmes neuroniques ne sont pas concurrentiels face aux heuristiques classiques [BAU86].
Pour l'instant, outre l'aspect novateur, le principal avantage des réseaux de neurones réside dans leur rapidité à résoudre ces problèmes complexes sur

des machines dédiées. Cette vitesse de résolution est un avantage important mais pas forcément décisif. En effet, si la qualité de la réponse est le seul critère d'évaluation, les algorithmes classiques sont alors largement supérieurs. Cependant, si on considère le rapport qualité/coût, leur supériorité n'est plus aussi flagrante.

En effet, pour certaines applications, il est préférable d'obtenir une "bonne" solution rapidement plutôt que de calculer une solution optimale de façon excessivement coûteuse [JOS87]. On retrouve ici un des domaines d'excellence du cerveau biologique et c'est dans ce contexte que les performances des réseaux de neurones peuvent être exploitées au mieux. De toute façon, au vu de la sophistication et de la complexité des techniques classiques, on peut considérer que les systèmes neuroniques en sont encore à leurs balbutiements.

Ces réseaux peuvent aussi être utilisés comme classifieurs. Mais dans ce cas, la théorie met en évidence une limitation importante : seuls les problèmes linéairement séparables peuvent être résolus. Cependant, il a été montré en [PER85], [PER86a,b,c,d] que ces réseaux avaient des capacités de généralisation intéressantes.

Enfin, l'utilisation des réseaux de Hopfield comme mémoires associatives a fait ses preuves. C'est l'application la plus courante. Nous avons détaillé des lois d'apprentissage qui permettent des applications efficaces sur ces réseaux et les résultats publiés mettent bien en évidence leurs performances.

La qualité de la formalisation théorique, l'existence de lois d'apprentissage efficaces, le vaste champ d'application couvert par ce réseau et la simplicité du modèle font, actuellement, du réseau de Hopfield un excellent candidat à l'intégration VLSI. Nous avons donc choisi d'étudier l'intégration d'un tel réseau. La phase de reconnaissance de ce réseau est d'ailleurs l'objet de la majorité des études d'intégration de réseaux de neurones artificiels.

II. Définition de l'étude

Dans ce chapitre, nous nous proposons d'étudier les réalisations matérielles existantes et de dégager les motivations qui nous ont amenés à réaliser les circuits présentés dans ce travail.
Les réalisations matérielles neuroniques sont nombreuses et peuvent être regroupées en deux classes :

- les coprocesseurs de simulation et les architectures neuroniques,
- les circuits intégrés spécifiques.

Nous présenterons quelques architectures dédiées mais nous nous attacherons plus particulièrement à la dernière approche car elle est pleine de promesses.

Cependant, avant d'entreprendre le développement d'un circuit intégré VLSI, il faut bien étudier la validité et le bien-fondé d'une telle entreprise. Réaliser un circuit intégré nécessite beaucoup de moyens. C'est un projet qui s'étale sur de nombreux mois. Outre les délais de conception, il faut tenir compte des délais liés à la fabrication et au test. La conception d'un circuit demande des investissements tant au niveau personnel que financier. La décision de concevoir un circuit intégré doit donc être motivée.
De plus, un circuit intégré n'est pas un dispositif évolutif comme un logiciel. Une fois spécifié et réalisé, un circuit ne peut pas être modifié et mis à jour à faible coût. Si les logiciels peuvent facilement s'adapter à de nouvelles lois d'apprentissage ou à de nouveaux modèles, les circuits intégrés n'ont pas la capacité de suivre aussi aisément les évolutions théoriques. Ces derniers doivent être basés sur des théories bien établies et stables afin de ne pas risquer d'être obsolètes dès leur fabrication. Aussi, nous avons choisi d'étudier l'intégration d'un réseau de Hopfield.

Une revue des réalisations matérielles existantes nous permettra de spécifier le cadre de notre étude. Enfin, la spécification fonctionnelle du circuit et les

problèmes liés à l'intégration à grande échelle nous conduiront à étudier, de façon générale, les architectures systoliques.

I. LES REALISATIONS NEURONIQUES

Les algorithmes neuroniques de reconnaissance sont relativement simples mais ils nécessitent beaucoup de calculs. Par exemple, 2 millions de multiplications et d'additions sont nécessaires pour évaluer un pas de reconnaissance dans un réseau de Hopfield contenant 1000 neurones. Certes, ces opérations n'ont pas besoin d'être d'une grande précision mais les simulations réalistes requièrent une puissance de calcul comparable à celle des super-calculateurs. Les simulations sur des ordinateurs classiques du type Von Neuman sont donc très lentes et impropres au développement d'applications neuroniques.

Les calculs neuro-mimétiques étant intrinsèquement parallèles, de grands progrès sont attendus avec l'avènement des ordinateurs parallèles. Les simulations sur des machines comme les Hypercubes sont plus aisées et plus performantes. Elles permettent la mise au point d'algorithmes complexes comme la rétro-propagation du gradient pour la reconnaissance de la parole [ROB88]. Mais, n'étant pas spécialisées, ces machines ne sont pas encore idéales.

La lenteur des simulations sur des architectures classiques a aussi conduit au développement d'architectures dédiées à la simulation de réseaux de neurones. A base de composants discrets, ces machines peuvent être intégrées comme coprocesseurs dans un environnement classique : micro/mini-ordinateur. Dans cette catégorie, on peut remarquer les deux réalisations de la

firme Texas Instrument :

- le système GRIFFIN est composé d'un ensemble de cartes à micro-processeur Intel 80188 [GAR88],
- ODDYSEY, basée sur un processeur de traitement du signal [PEN86], forme un coprocesseur de calcul matriciel. Une carte ODDYSEY permet de simuler un réseau de 512 neurones avec un rendement de 10^4 mises à jour de neurones par seconde.

Une approche similaire consiste à développer des architectures neuroniques proprement dites à base de composants discrets. Vu le parallélisme naturel des réseaux de neurones, les architectures parallèles sont bien adaptées comme l'architecture systolique hétérogène de la machine CRASY [GUE87]. Dans la version actuelle, 100 vecteurs de 128 composantes (128 neurones) peuvent être reconnus chaque seconde.

Ces machines sont utilisées comme "ateliers de simulation". Elles sont facilement paramétrables et s'adaptent à différents modèles et diverses lois d'apprentissage. Dans la spécification des algorithmes d'apprentissage, l'ajustement des paramètres est une tâche délicate qui conditionne largement le succès d'une application. Il est quasiment inconcevable de l'effectuer sur des machines conventionnelles. Les architectures dédiées permettent de combler cette carence et elles facilitent la mise au point de nouveaux modèles connexionistes.

Mais quand les modèles sont bien spécifiés et bien connus, comme le réseau de Hopfield, on peut aller plus loin dans l'intégration : la conception de circuits intégrés VLSI dédiés aux réseaux de neurones devient possible.
Les premiers prototypes de circuits intégrés font leur apparition. Les approches sont toutefois très diverses.

Elles diffèrent par :

- leur technologie : analogique / digitale,
- leur principe de mémorisation,
- leur programmation.

Les réalisations analogiques se décomposent en deux groupes.

La première approche est basée sur les principes spécifiés par Hopfield dans ses premiers articles [HOP82], [HOP84]. Il s'agit d'utiliser des amplificateurs opérationnels pour intégrer les neurones, les connexions inter-neurones étant réalisées par des résistances. Les valeurs des résistances sont choisies pour que la conductance représente les coefficients synaptiques ($R = 1/|C_{i,j}|$). Tout le problème réside donc dans l'intégration de ces résistances. Elles doivent atteindre de grandes valeurs (de l'ordre de 10^6 à 10^{11} Ω) afin de limiter la consommation du circuit intégré [GRA86]. Les procédés technologiques classiques ne permettent pas l'intégration de ces résistances [HUB86].

Des investigations se font actuellement sur la possibilité d'intégrer ces résistances avec des matériaux comme les oxydes de bismuth [SPE86] ou le silicium amorphe [THA86], [GRA86] et [HUB86]. Ces derniers utilisent un procédé technologique classique CMOS plus un procédé de micro-fabrication qui permet l'implantation des résistances. Des réseaux de Hopfield de 22 neurones puis de 512 neurones ont été réalisés selon cette technologie. Comme les résistances ne peuvent prendre que des valeurs positives, une loi d'apprentissage spéciale est nécessaire pour définir les poids synaptiques binaires de ce réseau. Une fois la matrice spécifiée, la programmation des résistances se fait une fois pour toutes par déposition de silicium amorphe.

D'autres tentatives, plus souples quant à la programmation des coefficients, sont basées sur le principe des EPROM. Des transistors à grille flottante sont utilisés pour représenter les coefficients synaptiques [RUE87]. Mais il est difficile de les programmer de façon précise et d'éviter leur dégradation au cours du temps. Pour résoudre ces problèmes, des dispositifs à transfert de

charges (CCD) peuvent être utilisés. La réalisation du M.I.T. [SAG86] intègre un réseau de Hopfield avec cette méthode. Les coefficients stockés ont entre 4 et 8 bits et peuvent être programmés électriquement. Cependant, d'après [RUC88], seule une intégration en 3D permet une intégration efficace des CCD. Toutes ces réalisations sont difficiles à programmer même quand les coefficients sont adaptables après fabrication.

La deuxième classe de réalisations analogiques consiste à créer des poids synaptiques variables avec une technologie VLSI classique, comme celle développée pour les circuits numériques. Les synapses ne sont plus réalisées à partir de résistances mais par des dispositifs générateurs de courant. Cette approche a été étudiée par H.P. Graf [GRA87], [GRA88]. Les résistances sont remplacées par deux points mémoires qui contrôlent la génération de courant (voir figure II.1).

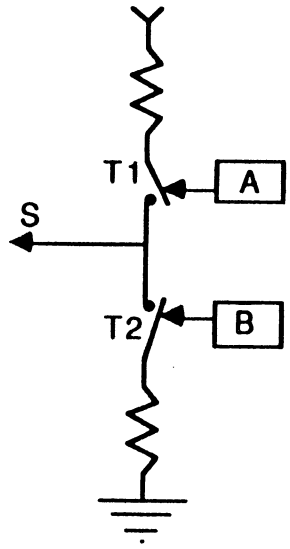


Figure II.1 Synapse selon [GRA87]

Les points mémoires A et B permettent la définition de coefficients synaptiques. 3 états sont possibles (-1, +1, 0) :

- A=0 et B=1 : l'interrupteur T1 est ouvert et T2 fermé. La synapse est un puits de courant. Le coefficient synaptique vaut -1,

- A=1 et B=0 : l'interrupteur T1 est fermé et T2 ouvert. La synapse est une source de courant. Le coefficient synaptique vaut +1,
- A=0 et B=0 : les deux interrupteurs T1 et T2 sont ouverts. Le coefficient synaptique vaut 0.

Un circuit comprenant 54 neurones a été réalisé en technologie CMOS.
Le réseau est facilement programmable en mémorisant les coefficients synaptiques dans les points mémoires des synapses. Mais la faible précision de leur mémorisation impose l'utilisation de lois d'apprentissage spécifiques et limite le champ d'application de tels réseaux. De plus, la somme analogique des courants manque de précision.
Une amélioration a été apportée par [VER88] pour combler ce dernier inconvénient mais les coefficients restent ternaires (-1, +1, 0).

La démarche de [TAR88] se démarque nettement des autres approches analogiques par sa façon de représenter les activités des neurones et les potentiels synaptiques ($\Sigma C_{i,j} . X_j$). Deux procédés sont étudiés.
La première technique procède par modulation de la longueur d'impulsion pour le calcul synaptique. Une résistance programmable permet de contrôler la décharge du signal d'entrée d'un inverseur. Il délivre alors une impulsion de longueur proportionnelle à la vitesse de décharge. La longueur de l'impulsion représente le produit ($C_{i,j} . X_j$) si l'entrée représente l'activité pré-synaptique X_j et si le coefficient synaptique $C_{i,j}$ contrôle la résistance programmable de décharge. Le poids synaptique est alors mémorisé sous forme capacitive sur la grille du transistor qui fait office de résistance. Afin de limiter la taille de ce transistor, il a été choisi d'utiliser un procédé de rafraîchissement proche de celui des mémoires dynamiques (DRAM).
Actuellement, les poids synaptiques sont stockés dans une mémoire extérieure au circuit. Un convertisseur digital/analogique permet d'envoyer les valeurs nécessaires au rafraîchissement des poids. La conversion doit être

suffisamment précise pour que les poids synaptiques puissent être exprimés par une tension comprise dans un intervalle de 1 Volt, afin que le transistor de décharge fonctionne correctement.

Le deuxième procédé utilisé par [TAR88] est plus proche du modèle biologique : les activités des neurones sont définies par des trains d'impulsions et les calculs synaptiques sont directement réalisés sur cette représentation.
Un neurone comprend un générateur d'impulsions réalisé par :

- un anneau d'inverseurs (ou "ring oscillator") qui délivre un signal carré,
- un dispositif simple qui transforme le signal carré en impulsions.

L'anneau d'inverseurs comprend une porte NAND qui permet de le programmer en fonction du potentiel de membrane : l'oscillation peut être stoppée si le potentiel de membrane devient négatif.
Ce potentiel de membrane arrive au neurone sous la forme de deux signaux impulsionnels, l'un inhibiteur, l'autre excitateur. Ces signaux traversent un intégrateur qui définit l'activité du neurone et contrôle la porte NAND de l'anneau d'inverseurs comme le montre la figure II.2.

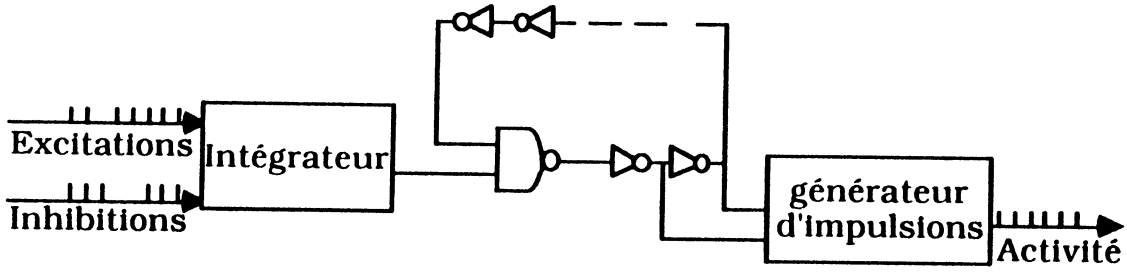


Figure II.2 Neurone selon [TAR88]

Dans chaque synapse, le coefficient synaptique est mémorisé dans un registre de M bits et l'activité pré-synaptique est tronquée, proportionnellement à ce coefficient. Par exemple, avec un poids de 0.5, une synapse laisse passer la moitié des impulsions représentant l'activité pré-synaptique. Le résultat du

produit ($C_{i,j}$. X_j) est un train d'impulsions qui est dirigé sur le fil excitateur ou inhibiteur du neurone selon le signe du poids synaptique.
Un circuit comprenant 64 synapses a été réalisé en technologie CMOS. Chaque synapse occupe une surface inférieure au dixième de mm^2. Les neurones sont actuellement intégrés sous forme de composants discrets afin de perfectionner la mise au point du système.

Toutes les réalisations matérielles ne sont pas basées sur une technologie analogique. Quelques prototypes sont réalisés ou en cours d'étude en technologie digitale.

Dans une équipe de l'université d'Edimbourg, l'intégration en technologie digitale d'un accélérateur de calcul synaptique a été réalisée [BUT88]. Il est composé d'une matrice de processeurs élémentaires intégrant chacun une synapse fonctionnant sur le principe du "bit-serial". Chaque processeur synaptique (i,j) calcule un bit du produit $C_{i,j}$ X_j et le transmet à la cellule (i,j+1).
Le modèle de neurone est une approximation du modèle sigmoïde par une fonction en escalier à 5 marches. Une activité est alors codée sur 3 bits pour : -1, -0.5, 0, +0.5 et +1. L'opérateur arithmétique reste simple car ajouter la moitié du coefficient revient à ajouter sa valeur décalée d'un rang. Codé sur 8 bits, le coefficient est ajouté au bit de somme partielle par un simple additionneur/soustracteur 1 bit.
Une matrice de 3 x 9 processeurs synaptiques a été intégrée en technologie CMOS dans un boîtier 64 broches. Chaque synapse occupe une surface de l'ordre de O.17 mm^2.
Ce circuit est à priori cascadable horizontalement et une architecture de 12 x 9 processeurs a été intégrée sur carte avec son environnement de mémoire et de contrôle. Un réseau de 256 neurones est "simulé par partie"

dans l'accélérateur synaptique 12 x 9 en 1ms. On atteint donc une puissance de calcul de 2 millions de mises à jour de neurones par seconde.

Dans l'étude de [WEI88], un processeur intègre la fonction d'un neurone complet, c'est à dire les coefficients synaptiques qui lui sont rattachés et la fonction de décision. Le modèle des neurones est du type Mac Culloch et Pitt. Mais, ce circuit à la particularité d'intégrer un algorithme d'apprentissage : la delta-projection présentée dans le chapitre précédent.
Le réseau fonctionne en deux phases :

- pendant l'apprentissage, les prototypes sont insérés dans le réseau par décalage, puis les poids sont mis à jour selon la loi de la delta-projection (I.13). Les prototypes sont présentés plusieurs fois jusqu'à obtention de leur stabilité.
- en mode de reconnaissance, le vecteur à reconnaître est présenté au réseau par décalage et le réseau itère jusqu'à obtention de la convergence.

Intégrer l'apprentissage augmente la complexité de la cellule neurone qui se compose de :

- une mémoire pour stocker les N coefficients. Cette mémoire est implantée sous forme d'un registre à décalage afin de minimiser le contrôle,
- un additionneur/soustracteur,
- un dispositif de détection de la convergence de l'apprentissage (stabilité des poids synaptiques),
- un dispositif de détection de la convergence de la phase de reconnaissance.

L'architecture envisagée est un réseau systolique linéaire : structure en anneau, comme le montre la figure II.3.

Un registre série/parallèle permet d'effectuer les Entrées/Sorties.

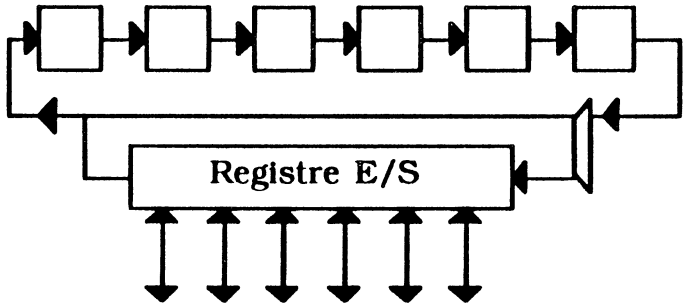


Figure II.3 Architecture en anneau de [WEI88]

Un circuit est en cours d'étude pour intégrer un neurone comprenant 64 poids synaptiques. Ce neurone servira de base pour un réseau complet de 64 neurones.

Toujours en technologie digitale, une intégration de réseaux multicouche est en cours d'étude [FAU88]. La structure de base est un réseau de cellules asynchrones communiquant par messages. Chaque cellule comprend deux parties :

- un partie routage qui assure l'acheminement des messages,
- une partie calcul qui implante la fonction neurone.

Les cellules communiquent par voisinage au travers de tampons. Un message envoyé par la cellule (i,j) à la cellule (i',j') se compose de deux éléments :

- la position relative de la cellule réceptrice par rapport à la cellule émettrice : delta_X = i'-i et delta_Y=j'-j,
- l'information proprement dite.

La partie routage des cellules permet d'acheminer ce message par l'intermédiaire des autres cellules, comme le montre la figure II.4. Ainsi toutes les cellules peuvent être logiquement connectées de façon souple et programmable.

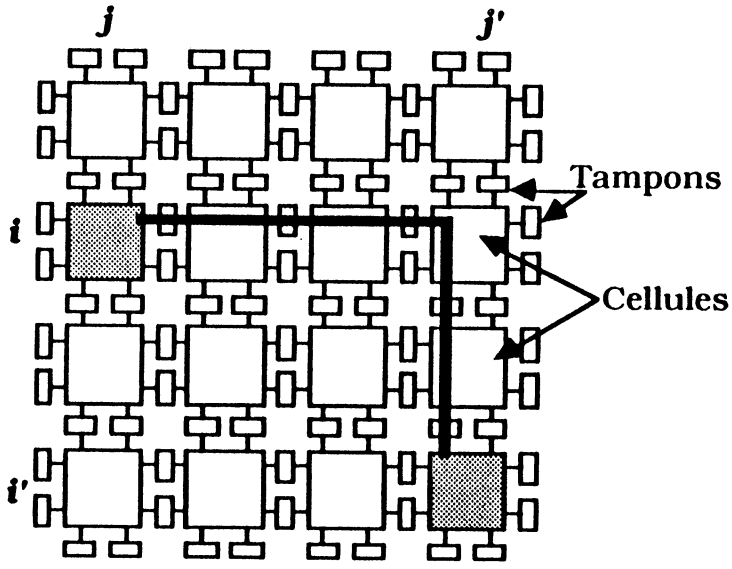


Figure II.4 Réseau de cellules asynchrones

Pendant la phase de reconnaissance, une cellule représentant le neurone j mémorise les poids synaptiques $C_{i,j}$ et communique aux cellules i de la couche suivante $C_{i,j} . X_j$ (flèche noire de la figure II.5). Pendant la phase d'apprentissage, la cellule i calcule le gradient γ_i et le transmet aux cellules j de la couche précédente qui mettent à jour leurs coefficients synaptiques (flèche grise).

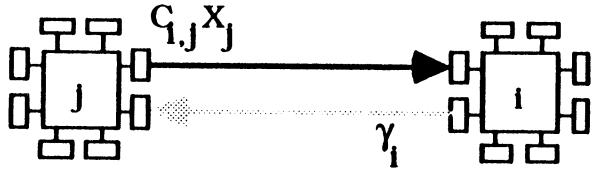


Figure II.5 Fonctions neurones

L'originalité du réseau réside dans son système de communication qui a déjà été mis en œuvre pour d'autres applications [COR88], [OBJ88], [LAT88]. Il permet de résoudre élégamment les problèmes liés aux connexions dans les réseaux de neurones. Le nombre de couches et le nombre d'éléments par couches est, aussi, facilement programmable.

La principale difficulté de l'intégration VLSI est de limiter la complexité de la cellule. Le modèle de neurone utilisé par le réseau implanté est du type sigmoïde. Une implantation directe de la formule sigmoïde est impossible et la

fonction doit être interpolée. L'algorithme de rétro-propagation du gradient nécessite des calculs relativement complexes comme le calcul de dérivées.

On peut aussi signaler des tentatives de réalisations optiques. Elles sont généralement basées sur de la mémoire holographique et des phénomènes physiques particuliers comme la photo-réfractivité, ou des dispositifs spéciaux comme les miroirs déformables. Les mémoires holographiques semblent de bonnes candidates pour la réalisation de mémoires associatives. Cette approche semble pleine de promesses, mais aucune réalisation entièrement optique n'a été développée à ce jour.

En conclusion, bien que des études soient menées sur l'intégration de lois d'apprentissage, on peut noter que tous les circuits effectivement réalisés à l'heure actuelle n'intègrent que la phase de reconnaissance. Pour une application donnée, une loi d'apprentissage adaptée peut fournir la matrice de coefficients adéquate. Lorsque le réseau est programmable, ces poids synaptiques peuvent être mémorisés. La phase de reconnaissance peut alors être effectuée avec des performances maximales.

II. CHOIX DE LA TECHNOLOGIE

Parmi les réalisations VLSI de circuits neuroniques, la technologie analogique semble avoir la faveur des concepteurs. La raison de ce choix peut être historique. En effet, le modèle établi par Hopfield trouve une implantation immédiate en technologie analogique. Cette technique permet d'intégrer de grands réseaux sur un seul circuit et de résoudre de façon élégante les problèmes liés aux interconnexions et aux calculs synaptiques.
Par contre, la programmation de ces réseaux est souvent limitée. Pour être opérationnels, les réseaux doivent pouvoir mémoriser les coefficients

synaptiques. Si cette mémorisation doit être effectuée lors de la fabrication, le circuit obtenu n'est plus alors dédié qu'à une seule application. On perd alors l'intérêt principal des réseaux de neurones : ils ne peuvent plus couvrir un vaste champ d'application et ne sont plus adaptables à différents problèmes. Les coefficients, synthèse de l'apprentissage, doivent donc être programmables.

Généralement, les solutions analogiques déjà réalisées n'implémentent que le signe des coefficients synaptiques (+1,0,-1). Ces coefficients, de précision réduite, conservent les propriétés générales des réseaux mais affaiblissent considérablement leurs performances ou requièrent des lois d'apprentissage spécifiques.

Face au problème de dégradation des performances en fonction de la précision des coefficients, nous avons été amenés à effectuer des simulations. Ces simulations ont été effectuées sur MicroVax® VMS à partir de programmes Pascal que nous avons développés.

Les résultats de ces simulations montrent, sur la figure II.6, l'influence de la précision des coefficients sur la capacité à identifier un vecteur prototype. Ces résultats ont été obtenus en effectuant 10 reconnaissances différentes avec une matrice synaptique calculée par la loi de la projection [PER85] [PER86a] sur des vecteurs prototypes choisis aléatoirement.

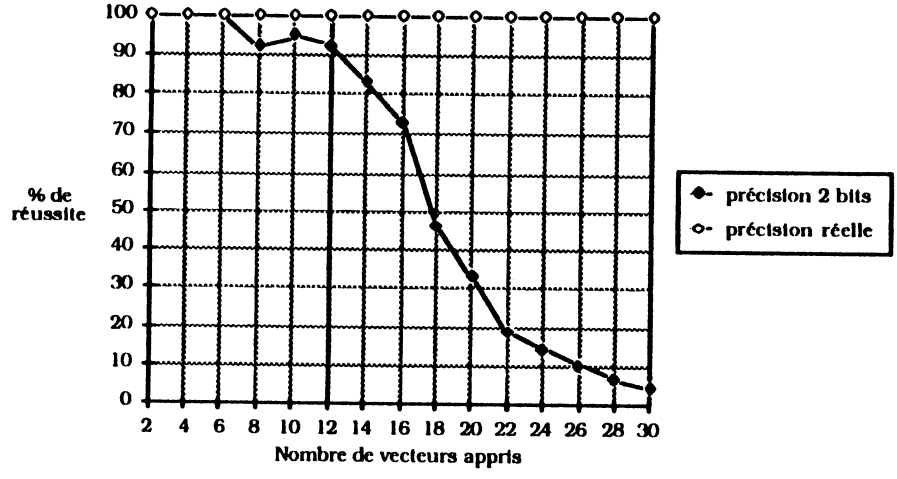


Figure II.6 Pourcentage de réussite de reconnaissance des vecteurs prototypes sur un réseau de 128 neurones.

Cette courbe nous montre qu'une trop faible précision provoque l'oubli des prototypes appris. La fonction de mémoire associative n'est donc plus assurée. On peut en conclure qu'une précision de 2 bits pour les coefficients synaptiques est largement insuffisante.

Outre la représentation sous forme d'impulsions qui est tout à fait originale, le projet de [TAR88] présente l'intérêt de mémoriser, sous forme digitale, les poids synaptiques. Les coefficients sont donc programmables et leur plus grande précision permet d'obtenir une reconnaissance plus performante. Cependant, l'intégration mixte digitale/analogique pose des problèmes dus à la sensibilité aux variations des tensions d'alimentation.

De plus, ces circuits n'échappent pas aux différents inconvénients liés à la technologie analogique. Les instabilités électriques, leur sensibilité au bruit sont autant d'inconvénients qui viennent s'ajouter aux difficultés d'interfaçage avec un environnement digital classique.
Ensuite, les circuits analogiques actuellement réalisés sont mono-chip. La taille du réseau est donc limitée par la taille d'un circuit intégré.
Enfin, la difficulté à disposer d'une filière technologique appropriée aux circuits analogiques conduit souvent à fabriquer ces circuits sur des filières dédiées aux circuits digitaux.
On retrouve alors tous les problèmes classiques des circuits analogiques réalisés en technologie digitale [SIV86], [TAR88] :

- non homogénéité des caractéristiques,
- dimensions approximatives des dispositifs,
- problèmes liés aux fuites de courant.

Les circuits digitaux sont beaucoup moins sensibles à ces problèmes.

D'un autre côté, les implantations digitales sont peu nombreuses. Pourtant elles permettent une programmation aisée et précise, et répondent bien aux exigences de souplesse.

Une architecture digitale permet :

- de modifier les coefficients synaptiques en fonction de l'application traitée,
- de mémoriser les coefficients de façon suffisamment précise pour garder des performances optimales,
- de pouvoir intégrer facilement le circuit dans les environnements habituels comme les micro ou mini-ordinateurs,
- d'utiliser les filières technologiques classiques, donc de pouvoir intégrer de façon fiable et à moindre coût,
- de pouvoir interconnecter simplement plusieurs circuits afin de réaliser des réseaux de grande taille, si l'architecture est extensible.

Notre choix s'est donc porté vers la technologie digitale. Nous avons présenté dans la section précédente deux projets qui étudient aussi l'intégration de réseaux de neurones entièrement digitaux. Le circuit présenté par [BUT88] est un "accélérateur" de calcul synaptique. Les coefficients sont calculés et chargés par la machine hôte. Le circuit reçoit le vecteur à reconnaître et retourne le résultat du produit matrice-vecteur. L'ordinateur hôte doit appliquer la fonction de décision sur ces résultats et prendre la décision de poursuivre ou non le calcul pour ce vecteur. Le rôle de la machine hôte est donc prépondérant.

Le projet de [WEI88] propose un circuit beaucoup plus autonome. Il permet d'effectuer la reconnaissance d'un vecteur avec des coefficients synaptiques calculés par le circuit lui-même. Intégrer un algorithme d'apprentissage sur un circuit peut paraître attrayant. Cependant, une fois intégré, l'algorithme d'apprentissage est figé. La loi choisie par [WEI88] est la loi de la delta-projection qui permet de calculer les coefficients pour obtenir un

comportement de mémoire associative. Certaines applications comme la résolution de problèmes d'optimisation combinatoire, qui nécessitent des lois d'apprentissage totalement différentes, ne peuvent pas être réalisées avec la loi câblée.

De plus, ce réseau intègre une fonction neurone par processeur. Chaque processeur va donc contenir l'ensemble des synapses qui lui sont associées. La cellule actuelle, conçue pour contenir 64 synapses, servira à construire un réseau de 64 neurones au maximum. L'architecture choisie n'est pas extensible au delà de cette limite.

Au vu des réalisations décrites ci-dessus, il apparait intéressant d'intégrer un réseau qui n'effectue que la reconnaissance mais libère au maximum la machine hôte. Il pourrait alors se comporter comme un véritable co-processeur neuronique. Les coefficients synaptiques doivent être programmables et avoir une précision suffisante pour couvrir un vaste champ d'application. De plus, comme la taille d'un circuit est limitée, l'architecture choisie doit être modulaire afin de permettre la construction de grands réseaux par assemblage de plusieurs circuits.

III. PROBLEMES LIES A L'INTEGRATION A GRANDE ECHELLE

Comme nous l'avons vu dans le chapitre précédent, les réseaux monocouche sont fortement interconnectés. Dans un réseau de Hopfield, les neurones sont tous reliés entre eux : l'interconnexion est complète.

Lorsqu'on étudie l'intégration d'un tel réseau, cette interconnexion est très pénalisante. Une implantation directe est bien sûr envisageable. Cette solution a déjà été choisie aussi bien pour une technologie digitale [BUT88] que pour une technologie analogique [GRA86], [SAG86], [SIV86], [RUC88], [VER88].

Cependant la longueur des connexions nécessaires pose des problèmes lors de

l'intégration. Les capacités importantes de ces lignes ralentissent sensiblement le fonctionnement de grands réseaux, surtout en technologie digitale. Selon la technologie utilisée, le routage de ces lignes peut aussi faire perdre beaucoup de place. Avec une telle implantation, un neurone doit envoyer son activité directement à tous les autres neurones. La cellule neurone a donc un "fan-out" de N, ce qui est inconcevable dès que N dépasse la dizaine. Il faut donc prévoir des dispositifs d'amplification très importants et assurer une distribution correcte des activités des neurones. De plus, en mode synchrone, des signaux peuvent mal se propager sur des lignes à grande capacité. Certains phénomènes, comme le décalage des signaux, peuvent alors se produire. Pour les éviter il faut ralentir le circuit ou concevoir avec précaution les lignes d'interconnexion afin que les retards dûs à la propagation soient sans conséquence grave.

Si on raisonne au niveau de l'intégration sur tranche entière, ces inconvénients deviennent réellement critiques : la probabilité de panne est proportionnelle à la longueur des connexions. Il est donc préférable de les réduire. Dans un prochain chapitre consacré à l'intégration sur tranche, les méthodes de reconfiguration de lignes globales seront étudiées plus en détail. Mais il est de toute façon plus facile de reconfigurer plusieurs petits éléments qu'un grand.

IV. ARCHITECTURES SYSTOLIQUES

Il est nécessaire de minimiser la longueur des interconnexions pour obtenir une intégration plus sûre, plus efficace et plus aisée. Il est évident que le parallélisme intrinsèque des réseaux de neurones conduit naturellement à des architectures parallèles. Parmi celles-ci, les architectures systoliques ont de nombreux avantages pour l'implantation envisagée.

D'où l'idée de développer une circulation systolique qui implante une

interconnexion fonctionnelle complète sur des connexions physiquement locales.

IV . 1 . Principe des architectures systoliques

La notion d'architecture systolique a été introduite par H.T. Kung [KUN80], et ainsi nommée car son fonctionnement rappelle le comportement rythmique du cœur, c'est à dire l'alternance Diastole/Systole Réception/Refoulement. Dans une machine systolique, chaque processeur joue le rôle du cœur : il reçoit régulièrement des données en entrée, leur applique un court traitement et émet des données en sortie. Le processeur assure donc un flux régulier de données dans le réseau.
On distingue différentes géométries de communication comme le montrent les figures II.7, II.8 et II.9.

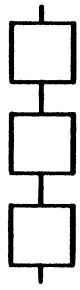
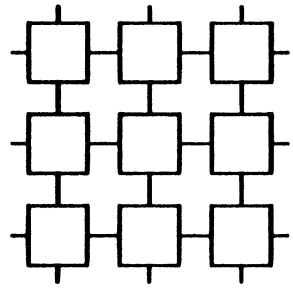


Figure II.7 Réseau linéaire



Réseau carré

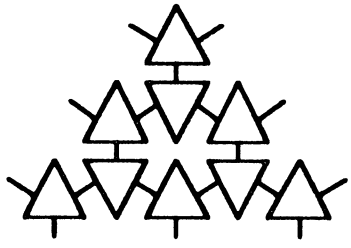
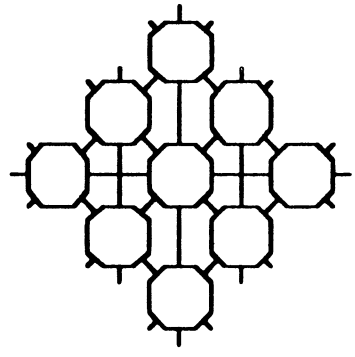


Figure II.8 Réseau triangulaire



Réseau hexagonal

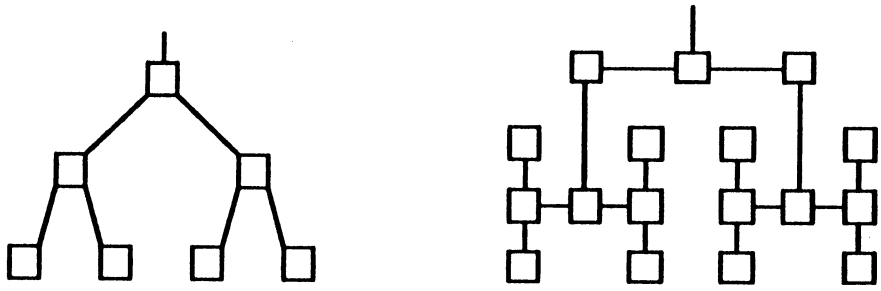


Figure II.9 Réseaux en arbre

Chaque géométrie est adaptée à la résolution d'une classe de problèmes [KUN80] :

- la convolution, le produit matrice-vecteur, certaines opérations de tri peuvent être traités sur un réseau linéaire,
- le réseau carré est idéal pour le traitement d'image, certains calculs matriciels,
- La décomposition LU, la multiplication de matrices se calculent avec un réseau hexagonal,

etc...

L'ensemble de ces réseaux permet de couvrir un vaste champ d'application. Mais sur une structure donnée et pour un problème donné, plusieurs solutions sont possibles. Kung dénombra 7 solutions différentes pour le calcul systolique de la convolution sur un réseau linéaire [KUN82]. Les solutions diffèrent principalement par la circulation des données.

Une donnée peut être :

- résidante : elle est mémorisée dans la cellule pour tous les calculs,
- émise : elle se déplace de cellule en cellule,
- diffusée : elle est communiquée à toutes les cellules en même temps.

Face à cette diversité de solutions, faciliter l'implantation d'algorithmes systoliques semble nécessaire et des méthodes de synthèse automatique de réseaux systoliques ont été développées.

Le système DIASTOL est issu de cette approche systématique [QUI84]. Il fonctionne en trois étapes. En premier lieu, l'algorithme à implanter est décrit sous forme d'un système d'équations récurrentes uniformes. Ensuite, il s'agit de choisir une fonction de temps T compatible avec les dépendances introduites par les équations qui décrivent l'algorithme systolique. Enfin, il faut définir une fonction d'allocation A qui alloue des processeurs différents pour des tâches concurrentes. Ce système fournit au concepteur un environnement qui lui permet de trouver rapidement et sûrement différentes alternatives architecturales pour un problème donné.

IV . 2 . Le produit matrice-vecteur systolique

Nous rappellons que la phase de reconnaissance consiste en un calcul récurrent d'un produit entre la matrice synaptique (C) et le vecteur des activités pré-synaptiques (X). Les activités binaires sont représentées par -1 ou +1. La fonction de décision des neurones est alors une fonction de seuil (F) qui prend le signe de la somme pondérée comme le montre l'équation (II.1).

$$X_i = F\left(\sum_{j=1}^{N} C_{i,j} \cdot X_j\right) \qquad \text{(II.1)}$$

La récurrence se poursuit jusqu'à obtention de la convergence, c'est à dire lorsque les activités ne changent plus entre deux pas de récurrence successifs (r et r+1) :

$$\forall\, i \in [1,N] \quad (X_{i,r} = X_{i,r+1}) \qquad \text{(II.2)}$$

Le produit matrice-vecteur fait partie de la classe des problèmes facilement résolus sur réseaux systoliques. Deux architectures sont particulièrement bien adaptées :

- les réseaux linéaires,
- les réseaux carrés.

Nous allons étudier différentes solutions possibles pour effectuer le produit matrice-vecteur :

- Solution 1 : réseau linéaire où le vecteur est résidant. A chaque processeur est associée une composante du vecteur. La matrice traverse le réseau Chaque cellule calcule $C_{i,j} . X_j$, l'ajoute à la somme partielle qui vient de sa voisine et transmet le résultat comme le montre la figure II.10. Les composantes du vecteur résultat sortent en série au bout du réseau. En régime stationnaire, le réseau fournit une composante tous les pas, soit un vecteur complet tous les N pas.

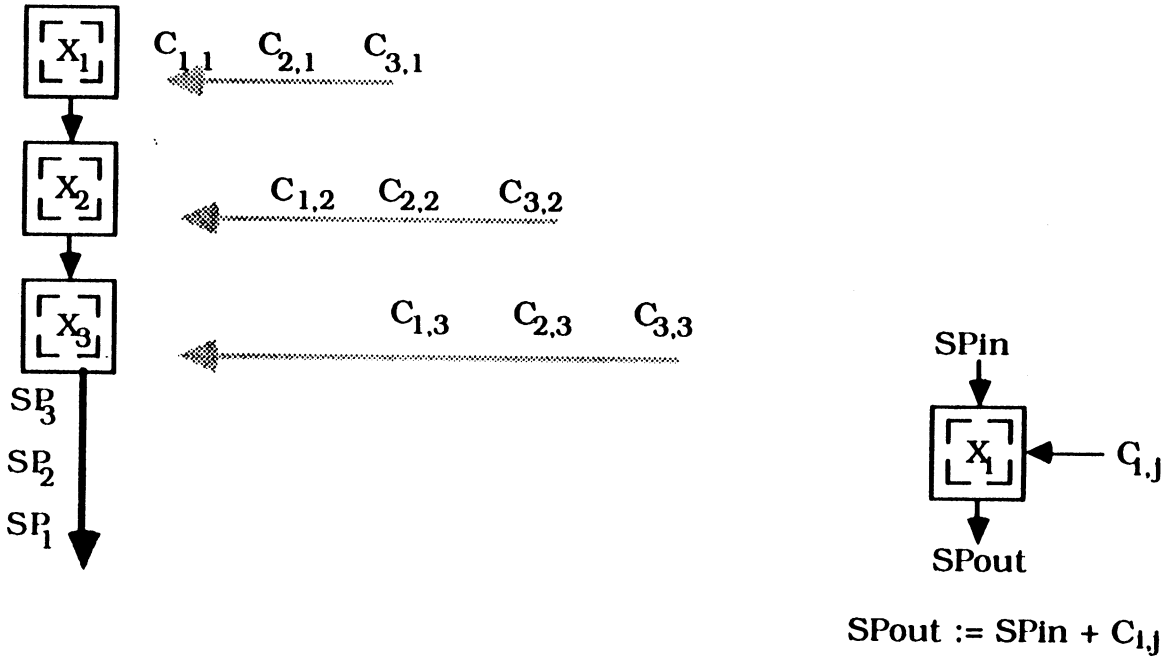


Figure II.10 Solution 1

- Solution 2 : réseau linéaire où les éléments de la matrice sont résidants. A chaque processeur est associée une ligne de la matrice et le vecteur traverse le réseau comme le montre la figure II.11. Les coefficients sont placés selon une permutation circulaire afin que chaque cellule calcule $C_{i,j} . X_j$. La somme partielle reste dans la cellule. En régime stationnaire, le vecteur résultat SP est émis, en parallèle, tous les N pas.

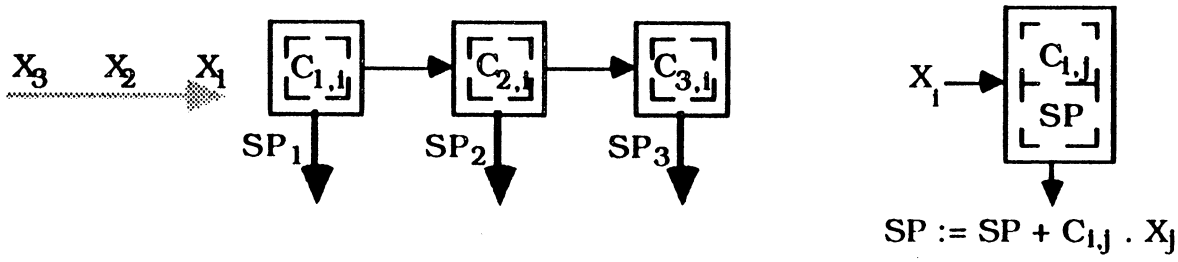


Figure II.11 Solution 2

- Solution 3 : réseau carré où les éléments de la matrice sont résidants. A chaque processeur est associé un élément de la matrice et le vecteur se propage à travers le réseau comme le montre la figure II.12. Chaque cellule calcule $C_{i,j} . X_j$, l'ajoute à la somme partielle entrante et communique le résultat à sa voisine. En mode stationnaire, N composantes SP sont calculées tous les pas.

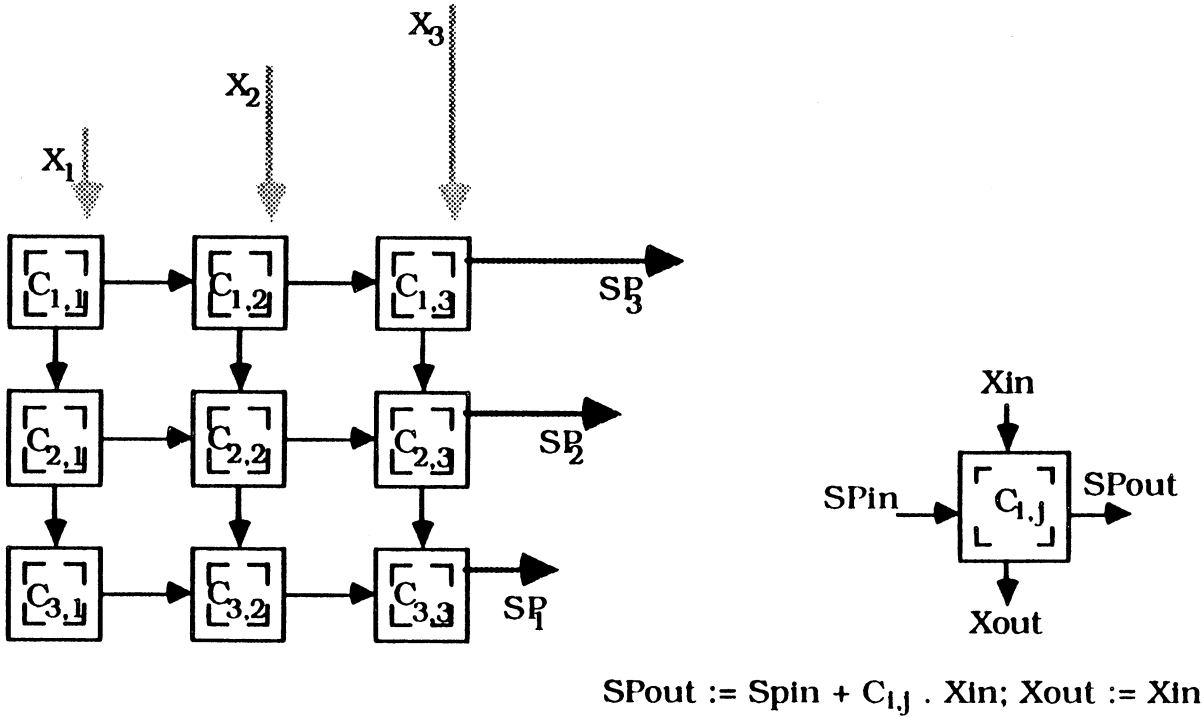


Figure II.12 Solution 3

Pour implanter la fonction de reconnaissance neuronique, il faut effectuer le produit (matrice synaptique) x (vecteur des activités pré-synaptiques). Dans

une architecture systolique dédiée à la reconnaissance, la matrice doit être résidante afin de pouvoir mémoriser dans le réseau la matrice synaptique. La solution 1 n'est donc pas adaptée à l'application envisagée.

Les solutions 2 et 3 sont intéressantes car les éléments de la matrice sont résidants et le vecteur est émis. Nous allons les étudier, plus en détail, en vue de l'intégration VLSI.

IV . 3 . Etude en vue de l'intégration des solutions proposées

Les architectures systoliques, en général, sont de très bonnes candidates à l'intégration VLSI ou WSI. Leur contrôle simple, leur modularité, leur extensibilité, la régularité et la faible complexité de chaque processeur sont autant d'avantages pour l'intégration.

Le produit matrice-vecteur peut atteindre un taux d'occupation de 100% sur un réseau linéaire [URQ84]. Mais, le réseau carré atteint aussi un taux d'occupation maximal et offre de meilleures performances.

Les processeurs des solutions 2 et 3 ne sont pas d'une grande complexité. Cependant, l'intégration d'un réseau de grande taille sur une seule puce est impossible. Il faut donc étudier l'extensibilité de ces solutions. Dans la solution 2, le nombre des coefficients dans chaque processeur est fixé et limite la taille maximale du réseau. Si un processeur est conçu pour N coefficients, la taille maximale du réseau est de N neurones. Par contre, la solution 3, en intégrant un processeur par synapse, permet d'augmenter plus facilement la taille du réseau. Chaque puce contient un sous-réseau, et un grand réseau peut être construit par juxtaposition de puces identiques.

Dans ce chapitre, nous avons défini les caractéristiques du circuit réalisé :

- le modèle de Hopfield a été choisi comme base de cette étude pour la qualité de sa formalisation théorique, l'existence de lois d'apprentissage efficaces, le vaste champ d'application couvert par ce réseau,
- il s'agit de définir un circuit digital autonome effectuant la phase de reconnaissance. Pour garder toute l'efficacité du modèle, le réseau est programmable,
- une architecture systolique est bien adaptée au problème.

Dans le chapitre suivant, nous décrivons l'architecture qui a été définie.

III. Définition Architecturale

Dans ce chapitre, une architecture dédiée à la phase de reconnaissance d'un réseau de Hopfield est définie. Ce réseau est du type monocouche où les neurones, du modèle Mac Culloch et Pitt, sont complètement interconnectés. La phase de reconnaissance calcule le produit récurrent de la matrice synaptique par le vecteur des activités pré-synaptiques.
Dans un premier temps, nous donnons les spécifications architecturales du réseau avant de décrire complètement la cellule de base.

I. UNE ARCHITECTURE SYSTOLIQUE POUR UN RESEAU DE HOPFIELD

Nous avons vu dans le chapitre précédent, qu'une architecture systolique est susceptible de résoudre le problème lié à l'interconnexion complète des neurones. L'architecture développée doit tenir compte de l'aspect récurrent de l'algorithme de reconnaissance. L'originalité de notre solution réside dans l'implantation de cette récurrence sur une architecture systolique.

I.1. Première approche : réseau systolique classique

Dans le chapitre précédent, nous avons présenté trois solutions pour le calcul du produit récurrent matrice-vecteur. La solution retenue est un réseau carré qui est facilement extensible et où la matrice des coefficients synaptiques est résidante.
L'idée de départ a donc été d'associer un élément de calcul par synapse. Ainsi un réseau de N neurones est constitué de N^2 cellules synaptiques contenant chacune un coefficient synaptique. Par la suite, la cellule $C_{i,j}$ sera identifiée à son coefficient $C_{i,j}$.
L'algorithme de reconnaissance peut être implanté sur une telle structure à condition de définir une circulation de données adéquate [BLA87].

Si l'on considère que les poids synaptiques sont déjà positionnés dans le tableau de cellules, le produit matrice-vecteur peut s'effectuer en faisant traverser la matrice synaptique C par le vecteur à reconnaître X. Comme les informations se propagent sous la forme d'un flux de données qui traverse de part en part le réseau systolique, nous pouvons parler d'architecture systolique à flux.

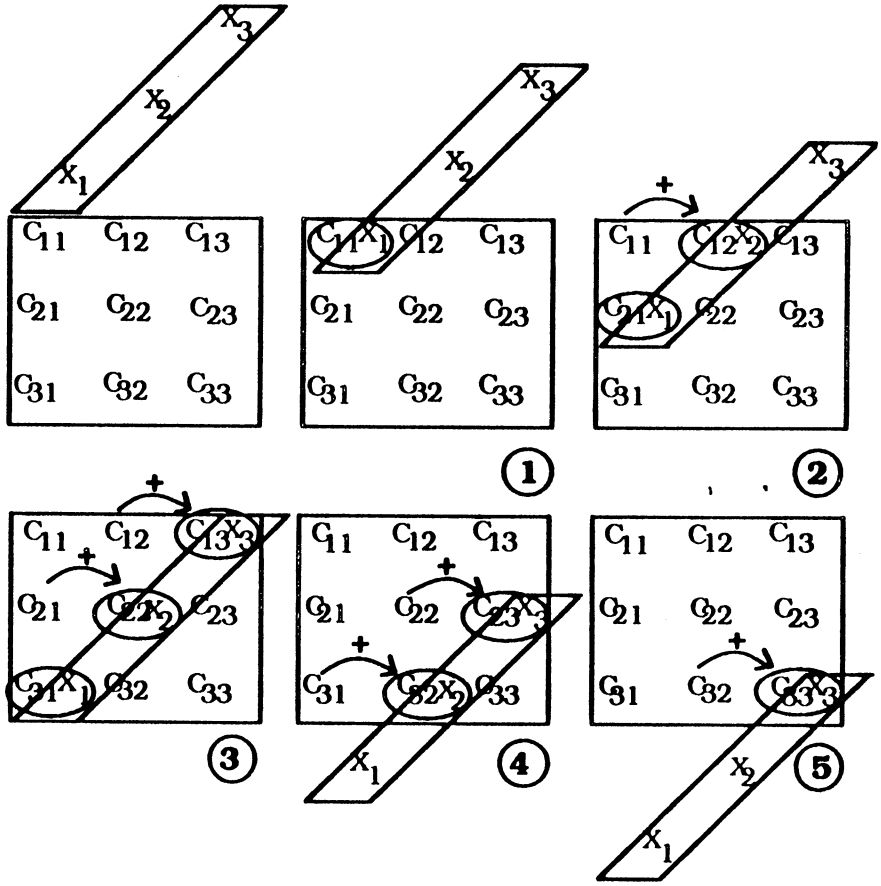


Figure III.1 Architecture systolique à flux

Chaque cellule (i,j) calcule le produit $C_{i,j}.X_j$, ajoute ce résultat à celui qu'elle reçoit de sa voisine de gauche puis transmet la somme partielle calculée à sa voisine de droite.

Dans cette architecture systolique à flux, la circulation des données se fait selon deux axes:

- chaque composante du vecteur X parcourt la matrice verticalement du Nord au Sud.

- les sommes partielles de chaque ligne parcourent la matrice horizontalement d'Ouest en Est.

On peut constater que le calcul complet d'une somme partielle nécessite N pas de calcul (dans la figure III.1 N=3). L'évaluation de toutes les sommes partielles requiert 2.N pas.

Cette étape constitue la fin d'un pas de récurrence. En effet, les sommes partielles qui apparaissent sur le bord Est du réseau sont seuillées et deviennent les nouvelles composantes du vecteur X. Pour poursuivre le calcul, le résultat doit être remis à disposition du réseau. On doit alors relier le bord Est au bord Nord. Il faut reboucler les résultats sur les points d'entrée et donc passer d'un flux de données à une circulation de données.

Cependant cette architecture ne résout pas le problème crucial des interconnexions. Comme le montre la figure III.2, une implantation directe n'est pas satisfaisante du point de vue du routage (connexions en gras).

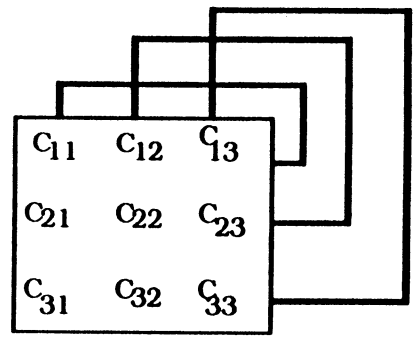


Figure III.2 Implantation directe

I.2. Deuxième approche : réseau systolique à réinjection

Lors de l'intégration d'Aplysie, nous avons amélioré la circulation de données précédemment décrite en réduisant les interconnexions extérieures au réseau. Une fois de plus, l'idée a été de transformer une liaison directe en une circulation systolique par voisinage. Les données circulent à l'intérieur du réseau en "rebondissant" sur les bords : modèle à réinjection de données. Trois

règles ont été suivies pour passer du modèle systolique à flux au modèle à réinjection :

- règle de séquencement : le séquencement des calculs est celui présenté dans le modèle à flux. Voir figure III.1,
- règle systolique : un processeur effectue son calcul sur ses données internes et celles délivrées par ses cellules voisines. Les données arrivent dans une cellule au moment où elles sont nécessaires au calcul,
- régle de récurrence : à la fin d'un pas de récurrence, les nouvelles composantes du vecteur X sont en position pour remplacer les anciennes.

La circulation présentée ci-dessous nécessite deux conditions initiales :

- les coefficients synaptiques $C_{i,j}$ sont dans leur cellule respective,
- les composantes du vecteur $X_{i,r}$ sont dans les cellules diagonales $C_{i,i}$. L'indice r indique le pas de récurrence : si l'on débute la reconnaissance, r est nul.

Le calcul d'un pas de récurrence se fera en 2N étapes. L'exemple développé dans les pages suivantes, consiste en 3 neurones (N=3) donc 9 cellules et le calcul se fera donc en 6 étapes :

Etape 1 : (cf figure III.3) pour respecter la règle de séquencement, on ne procède qu'au calcul de la somme partielle $SP_{1,1}=C_{1,1}.X_{1,r}$. Aucun traitement n'est effectué sur $X_{2,r}$. et $X_{3,r}$.

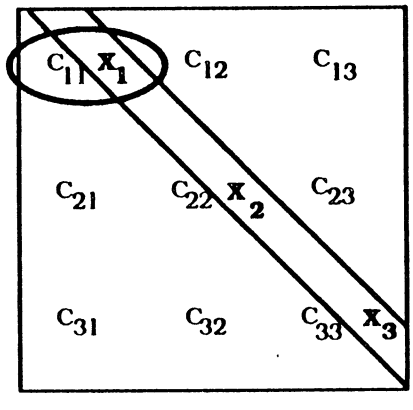


Figure III.3 Etape 1

Etape 2 : afin de respecter la règle de séquencement, les calculs devront s'effectuer dans les cellules $C_{1,2}$ et $C_{2,1}$. En vertu de la règle systolique, les données devront se propager préalablement comme suit :

- $X_{1,r}$ est transmis vers le Sud afin d'être présent dans la cellule $C_{2,1}$ pour le calcul de $SP_{2,1}=C_{2,1}.X_1$.
- La somme partielle $SP_{1,1}$ calculée par $C_{1,1}$ se propage horizontalement vers la cellule $C_{1,2}$.
- De même, $X_{2,r}$ doit être présenté à la cellule $C_{1,2}$ et doit donc être propagé vers le Nord afin de calculer $SP_{1,2}=SP_{1,1}+C_{1,2}.X_2$.
- De plus, à l'étape suivante, $C_{1,3}$ devra disposer de $X_{3,r}$. Pour anticiper cette étape, il faut, d'ores et déjà, le faire "monter" en transitant par la cellule $C_{2,3}$.

L'étape 2 peut être accomplie comme le montre la figure suivante.

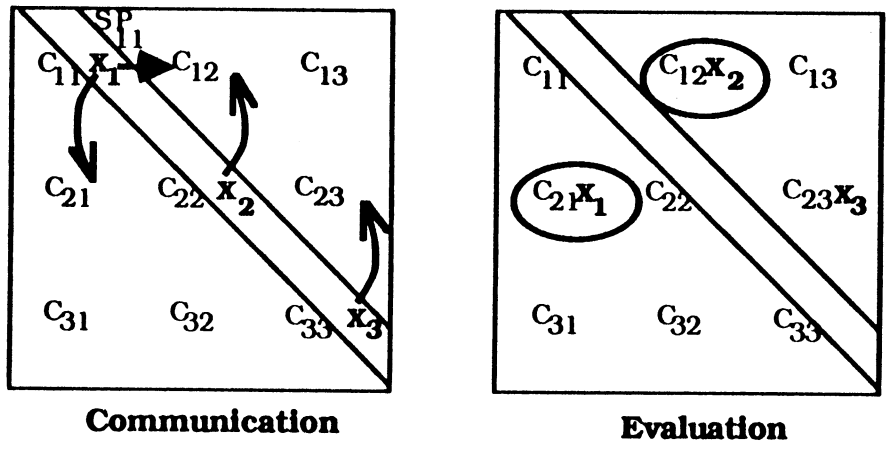


Figure III.4 Etape 2

Etape 3 : à cette étape, les calculs s'effectuent dans les cellules $C_{1,3}$, $C_{2,2}$ et $C_{3,1}$. Comme précédemment, les données se propagent pour être disponibles dans ces cellules :

- $X_{1,r}$ est transmis vers le Sud afin d'être présent dans la cellule $C_{3,1}$.
- Il faut propager horizontalement les sommes partielles calculées par $C_{1,2}$ et $C_{2,1}$ vers leur voisine Est.

- De même, $X_{2,r}$ doit être présenté à la cellule $C_{2,2}$ et doit donc être propagé vers le Sud.
- Enfin, $X_{3,r}$ est transmis vers le Nord à la cellule $C_{1,3}$.

Au pas suivant, nous avons donc l'évolution :

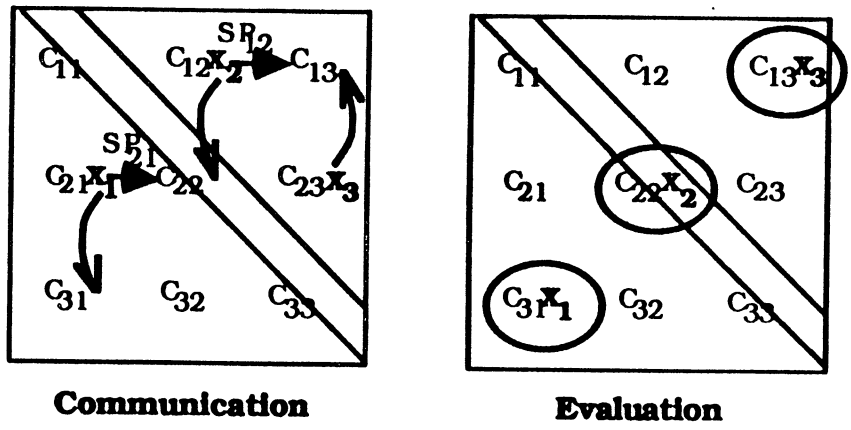


Figure III.5 Etape 3

Etape 4 : à la fin de l'étape 3, le calcul de $SP_{1,3}$ est achevé, alors que le calcul des autres sommes partielles est encore en cours.

De façon à obtenir la nouvelle composante $X_{1,r+1}$, une fonction de seuil F est appliquée à $SP_{1,3}$. En vertu de la règle de récurrence, cette nouvelle composante doit rejoindre la position initiale de $X_{1,r}$ qui est la cellule diagonale $C_{1,1}$. Pour cela, on va mettre à profit le temps dont on dispose avant la fin du calcul de la dernière somme partielle $SP_{3,3}$: $X_{1,r+1}$ est "réinjectée" dans le réseau. Sur la figure III.6 décrivant l'étape 4, la composante réinjectée $X_{1,r+1}$ est notée X'_1.

De plus, il faut effectuer la comparaison $X_{1,r} = X_{1,r+1}$ pour chaque composante afin de détecter la convergence de la reconnaissance. D'après la règle systolique, il faut amener l'ancienne et la nouvelle valeur dans une même cellule. On remarque que la nouvelle composante $X_{1,r+1}$ est calculée au moment où la valeur de $X_{1,r}$ apparaît au Sud du réseau. Cette dernière va, elle

aussi, être réinjectée dans le réseau afin de se trouver dans la cellule diagonale au même moment que la nouvelle composante.

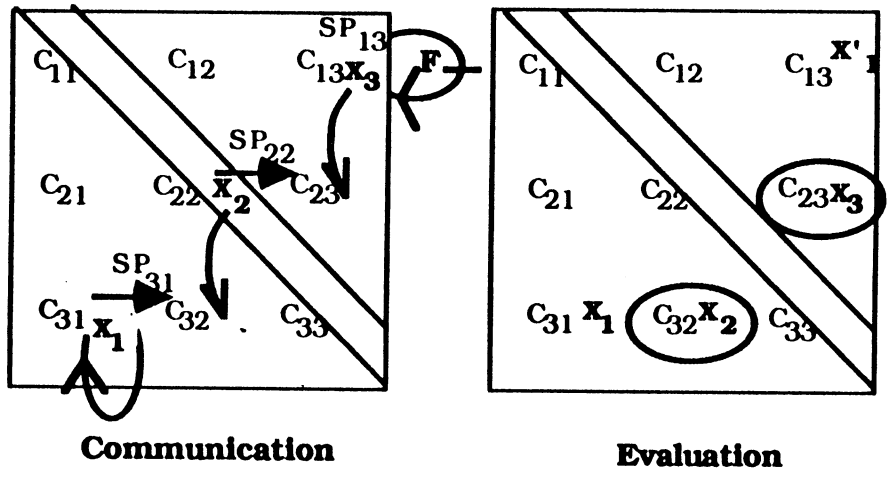


Figure III.6 Etape 4

L'étape 4 se distingue donc des 3 premières étapes par la réinjection de $X_{1,r+1}$ sur le bord Est et de $X_{1,r}$ sur le bord Sud.

Etape 5 : la nouvelle composante $X_{2,r+1}$ (notée X'_2 dans la figure III.7) et l'ancienne composante $X_{2,r}$ sont réinjectées. De plus, $X_{1,r+1}$ (notée X'_1) et $X_{1,r}$ continuent leur déplacement.

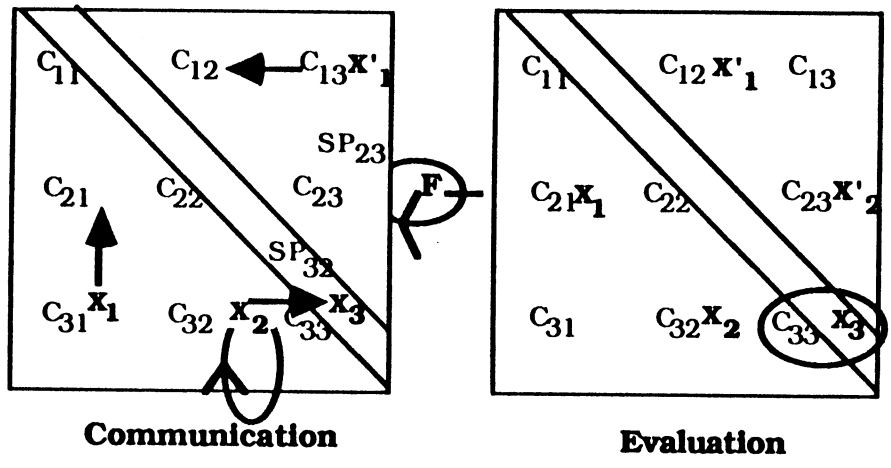


Figure III.7 Etape 5

Etape 6 : la progression continue comme précédemment décrite. La nouvelle composante $X_{3,r+1}$ (notée X'_3) et l'ancienne composante $X_{3,r}$ sont réinjectées.

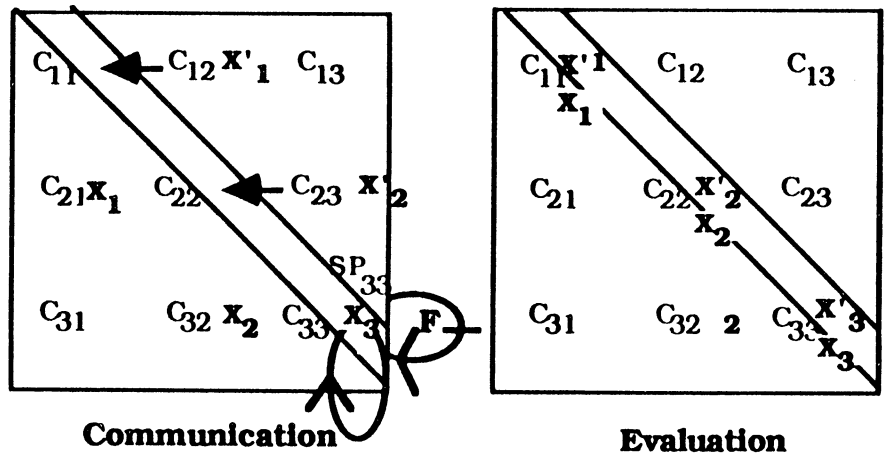


Figure III.8 Etape 6

A la fin de l'étape 6, la fin du pas de récurrence est atteinte. La règle de récurrence a bien été respectée : les nouvelles composantes sont situées dans les cellules diagonales. Comme prévu, grâce à la réinjection des anciennes composantes sur le bord Sud, les anciennes et nouvelles composantes du vecteur sont présentes simultanément dans les cellules diagonales. On peut alors procéder à l'évaluation de la convergence locale $CL_l = (X_{l,r}=X_{l,r+1})$.

La figure III.9 montre l'architecture du réseau implantant la circulation de données présentée.

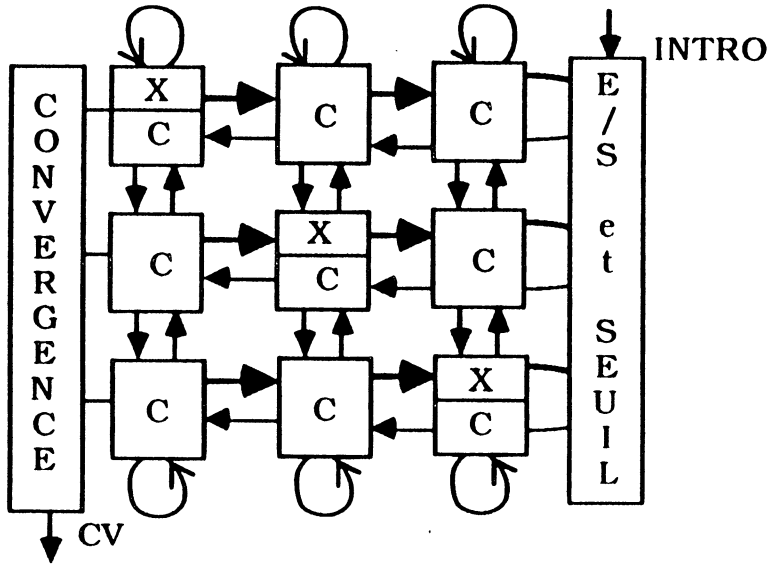


Figure III.9 Architecture du réseau

I.3. Convergence, Entrée/Sortie, rebouclage et seuil

Pour élaborer la convergence globale il faut calculer le "ET" des convergences locales :

$$CV = \bigwedge_{i=1,N} (CL_i) = \bigwedge_{i=1,N} (X_{i,r} = X_{i,r+1}).$$

Le résultat de la comparaison CL_i va se propager horizontalement vers l'Ouest à partir des cellules diagonales. Un dispositif systolique (voir figure III.10) situé sur le bord Ouest permet de calculer CV en N pas. Les convergences partielles sont notées CV_i, CV_N étant le signal de convergence CV.

- au premier pas, CL_1 arrive sur le bord Ouest et le dispositif de convergence la transmet au Sud : $CV_1 = CL_1$,
- au second pas, CL_2 arrive sur le bord Ouest. le dispositif de convergence peut alors calculer $CV_2 = (CV_1 \wedge CL_2)$, et le propager vers le sud.
- au $N^{ième}$ pas, CL_N arrive sur le bord Ouest. Le dispositif de convergence calcule $CV = CV_N = (CV_{N-1} \wedge CL_N)$.

Pendant que s'élabore la convergence globale sur le bord Ouest (N pas de calcul), le réseau anticipe le calcul du pas de récurrence suivant. Si CV est nul il faut poursuivre la récurrence et le calcul se poursuit sans perdre de temps. Dans le cas contraire, la reconnaissance a convergé : le vecteur a été reconnu. L'anticipation provoque un pas de récurrence supplémentaire qui ne change pas les composantes du vecteur car la condition de convergence est atteinte.
Au moment où le signal CV est calculé, la première somme pondérée correspondant à la première composante du vecteur reconnu atteint le bord Est. Un vecteur reconnu doit être extrait du réseau pour laisser la place à un nouveau vecteur à reconnaître. Le dispositif d'Entrée/Sortie situé sur le bord Est du réseau (Voir figure III.10) permet, par un jeu de multiplexeurs, d'effectuer soit le rebouclage, soit une Entrée/Sortie. Pour extraire un vecteur

reconnu, INTRO doit être mis à 1. Le dispositif s'étalant sur tout le coté Est du réseau fait office de "fermeture Eclair®" et permet l'Entrée/Sortie de toutes les composantes. La "fermeture Eclair®" supprime le rebouclage lorsqu'une composante X_i arrive sur le bord Est et permet l'entrée de la $i^{ème}$ composante d'un nouveau vecteur à reconnaître.

La fonction de seuil est appliquée sur le bord Est. Son implantation sera détaillée plus tard en même temps que celle de la "fermeture Eclair®".

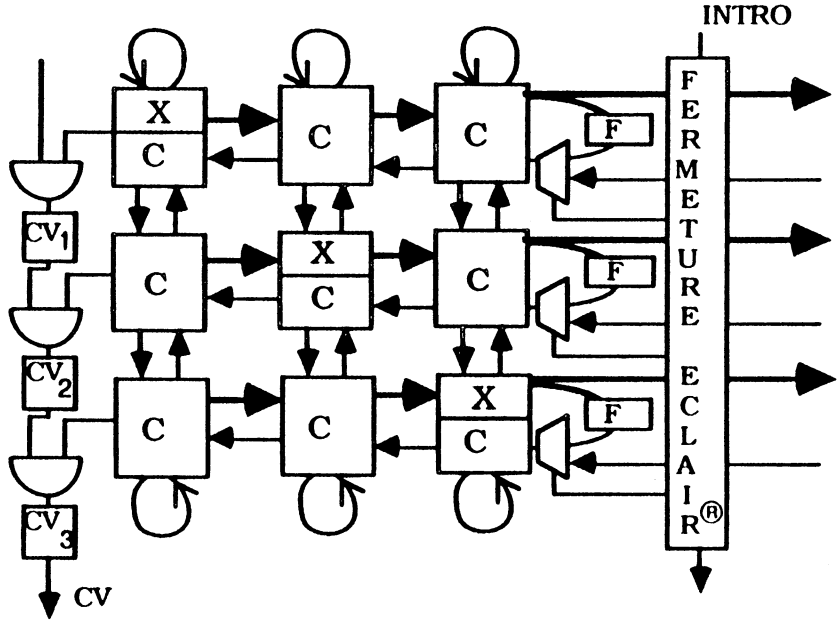


Figure III.10 Convergence et Entrée/Sortie

I.4. Validation de l'architecture fonctionnelle

Un résumé de cette circulation de données pour un réseau de N neurones s'impose.

- t=0 : les composantes de X sont dans la diagonale. Le premier pas de récurrence commence. L'indice r vaut 0.
- t=1 à N : les composantes $X_{i,0}$ circulent verticalement en commençant vers le Nord et en rebondissant sur le bord Nord. Les cellules calculent $C_{i,j}X_j$ et ajoutent ce résultat à la somme partielle calculée par leur voisine de l'Ouest. Les sommes partielles circulent donc d'Ouest en Est.

- t=N : la première somme partielle atteint le bord Est. Elle est seuillée puis réinjectée dans le réseau. De même, $X_{1,0}$ rebondit sur le bord Sud.
- t=N+1 à 2N : les autres sommes partielles et les composantes de X subissent le même traitement. Les sommes partielles qui circulent d'Ouest en Est sont seuillées lorsqu'elles atteignent le bord Est puis réinjectées dans le réseau. Les anciennes composantes rebondissent sur le bord Sud et remontent jusque dans les cellules diagonales.
- t=2N : chaque cellule diagonale $C_{I,I}$ contient l'ancienne et la nouvelle composante. La convergence locale CL_I peut être évaluée et l'affectation $X_{I,r}=X_{I,r+1}$ peut avoir lieu afin de passer au pas de récurrence suivant. L'indice r vaut 1.
- t=2N+1 à 3N : la circulation des $X_{I,2}$ et des sommes partielles s'effectue comme dans le cas t=1 à N. En plus les convergences locales se propagent vers l'Ouest où le dispositif de convergence effectue le calcul systolique de la convergence globale CV.
- t=3N : à l'Ouest le signal de convergence est calculé alors qu'à l'Est apparait la première composante du vecteur. Si le signal CV vaut 1 (respectivement 0), le dispositif d'E/S doit procéder à l'Entrée/Sortie de la première composante (respectivement au rebouclage). Pour cela il suffit de relier INTRO à CV.
- t=3N+1 à 4N : les sommes partielles sont seuillées et extraites du réseau (respectivement rebouclées) au fur et à mesure qu'elles atteignent le bord Est par la "fermeture Eclair®".
- t=4N : la reconnaissance d'un nouveau vecteur commence (respectivement un nouveau pas de récurrence commence : r=2).

I.4.1. Spécifications de la cellule

Pour assurer une telle circulation, il faut pouvoir différencier et stocker les composantes de X qui "montent" et celles qui "descendent", ainsi que

mémoriser les sommes partielles. D'autre part, les cellules doivent assurer la réinjection des nouvelles composantes après application du seuil et le transit de la convergence locale. La cellule devrait donc contenir 6 registres:

D : valeur descendante de X_r,

M : valeur montante de X_r,

NV: nouvelle valeur X_{r+1} réinjectée

CL: convergence locale,

SP: somme partielle,

C : coefficient local de la matrice synaptique.

Cependant, on peut remarquer que le registre NV est nécessaire pour les cellules supérieures à la diagonale (appelées CSD comme le montre la figure III.11) mais inutile pour les cellules inférieures ou égales à la diagonale (appelées CID + CD). En effet, une nouvelle composante rebondit sur la diagonale sans passer dans le registre NV des CD et n'atteint jamais les CID. D'autre part, le résultat d'une comparaison CL_l se propage horizontalement à partir des CD, transite par les CID et ne passe pas par les CSD. Une simplification architecturale peut alors être effectuée : les registres NV et CL peuvent être fusionnés en un seul registre appelé NV par la suite. Chaque cellule doit alors contenir 5 registres et NV mémorise soit la nouvelle valeur X_{r+1} réinjectée si la cellule est une CSD soit la convergence locale si c'est une CID ou une CD.

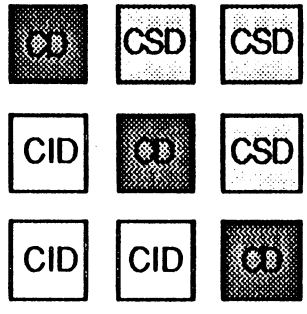


Figure III.11 Distinction entre les différents types de cellules

Chaque cellule assure 2 fonctions comme les figures III.2 à III.7 le montrent :

- évaluation,
- communication.

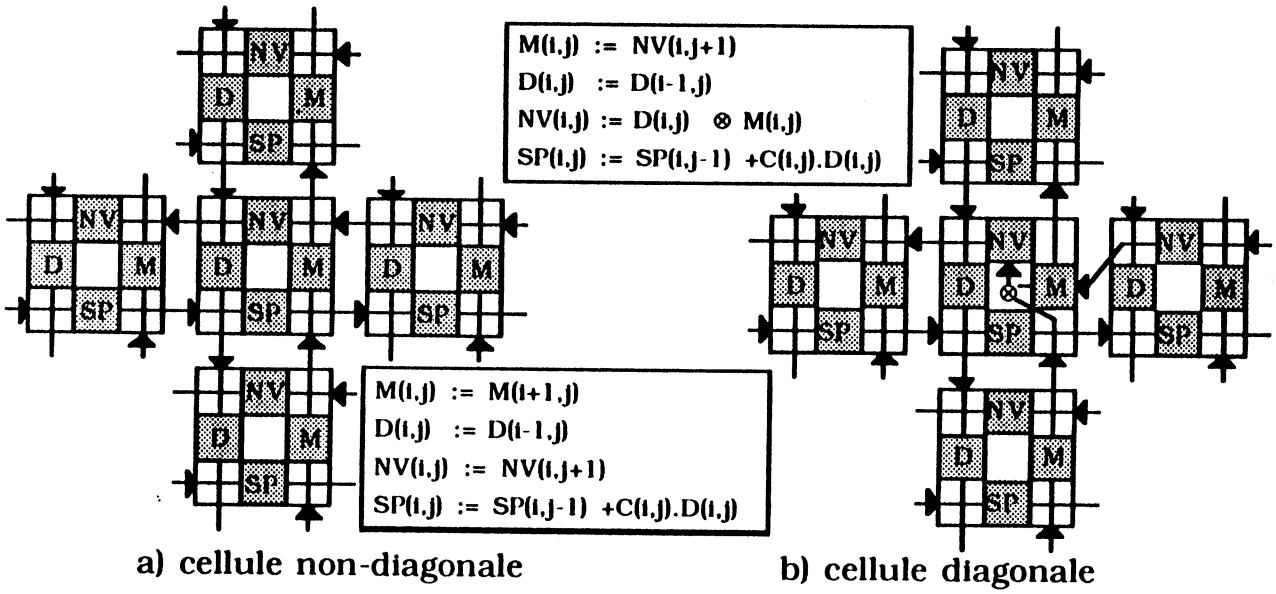
Le calcul effectué par chaque cellule i,j est :

$SP_{i,j} = SP_{i,j-1} + D_{i-1,j} \cdot C_{i,j}$

avec $SP_{i,0}=0$, $D_{0,i} = M_{1,j}$.

En ce qui concerne la communication, on dégage deux types de comportements :

- les cellules non diagonales assurent des communications orthogonales comme le montre la figure III.12a.
- les cellules implantées sur la diagonale du réseau ont un rôle différent. Elles assurent le calcul de la convergence locale et l'affectation $X_{i,r} := X_{i,r+1}$. Les communications s'effectuent selon le schéma de la figure III.12b.



a) cellule non-diagonale b) cellule diagonale

Figure III.12 Communication

I.4.2. Validation de l'architecture

Nous avons décrit dans le paragraphe précédent la circulation de données. Il faut s'assurer plus formellement que le calcul souhaité est bien effectué par l'architecture proposée.

La spécification du circuit peut être exprimée par la formule récurrente suivante :

$$X_{i,r} = F\left(\sum_{j=1}^{N} C_{i,j} \cdot X_{j,r-1}\right) \qquad \text{(III.1)}$$

- i et j décrivent la localisation d'un élément de la matrice des coefficients,
- r est le pas de récurrence,
- t représente le pas de calcul,
- N est l'ordre de la matrice,
- F est la fonction de seuil,
- X est le vecteur à reconnaître.

Pour valider l'architecture proposée, nous décrivons, dans un premier temps, son fonctionnement par les équations (III.2), (III.3), (III.4) et (III.5) ci-dessous.

L'équation suivante exprime le calcul des sommes partielles SP:

$$SP_{i,j,t} = SP_{i,j-1,t-1} + C_{i,j} \cdot D_{i-1,j,t-1} \qquad \text{(III.2)}$$

Ensuite nous traduisons l'évolution de NV. NV assure la réinjection des nouvelles composantes dans les cellules de la diagonale supérieure et le transit de la convergence locale dans la diagonale inférieure. Si j=N, il s'agit d'une cellule du bord Est. C'est dans ces cellules que le calcul des sommes partielles s'achève, que la fonction de seuil est appliquée et que NV mémorise la valeur des nouvelles composantes : on change de pas de récurrence.

D'autre part, dans les cellules diagonales, la comparaison $X_{i,r}=X_{i,r+1}$ est effectuée. Cette comparaison est calculée par la fonction booléenne conjonction, notée ⊗. On obtient donc l'équation suivante pour NV :

$$NV_{i,j,t} = \text{si } j=N \text{ alors } F(SP_{i,N-1,t-1})$$
$$\text{sinon si } i \neq j \text{ alors } NV_{i,j+1,t-1}$$
$$\text{sinon } (M_{i+1,j,t-1} \otimes NV_{i,j+1,t-1}) \qquad \text{(III.3)}$$

Nous décrivons ensuite le fonctionnement de M qui prend la valeur de NV sur la diagonale ou, au bord Sud, la valeur de D réinjectée :

$$\mathbf{M_{i,j,t} = si\ i=j\ alors\ NV_{i,j+1,t-1}\ sinon\ si\ i\neq N\ alors\ M_{i+1,j,t-1}\ sinon\ D_{N,j,t-1}} \qquad \textbf{(III.4)}$$

Une fois le côté Nord atteint, les X_i redescendent pour servir aux calculs des SP :

$$\mathbf{D_{i,j,t} = si\ i\neq 1\ alors\ D_{i-1,j,t-1}\ sinon\ M_{1,j,t-1}} \qquad \textbf{(III.5)}$$

Démontrons que ce système effectue bien le calcul voulu (III.1).

Démonstration :

En itérant (III.3), jusqu'à j=N, nous obtenons:

$$\mathbf{NV_{i,j,t} = NV_{i,N,t-N+j}} \qquad \textbf{(III.6)}$$

En appliquant (III.4) plusieurs fois jusqu'à i=j, nous obtenons:

$M_{i,j,t} = M_{j,j,t-j+i}$

$= NV_{j,j,t-j+i}$ en appliquant (III.4) si i=j

$= NV_{j,N,t-N+j-j+i}$ d'après (III.6)

Nous obtenons donc:

$$\mathbf{M_{i,j,t} = NV_{j,N,t-N+i}} \qquad \textbf{(III.7)}$$

En appliquant (III.5) si i=0 nous obtenons:

$D_{1,j,t} = M_{1,j,t-1}$

$= M_{j,j,t-j}$ en appliquant (III.4) jusqu'à i=j

$= NV_{j,j+1,t-j-1}$ d'après (III.4) si i=j

$= NV_{j,N,t-j-N+j}$ d'après (III.6)

Nous avons donc:

$$\mathbf{D_{1,j,t} = NV_{j,N,t-N}} \qquad \textbf{(III.8)}$$

En appliquant (III.5) jusqu'à i=0 nous obtenons:

$$D_{i,j,t} = D_{1,j,t-i+1}$$

D'après (III.8), nous avons donc:

$$\mathbf{D_{i,j,t} = NV_{j,N,t-N-i+1}} \qquad \textbf{(III.9)}$$

En appliquant (III.9) sur l'équation (III.2) nous obtenons:

$$SP_{i,j,t} = SP_{i,j-1,t-1} + C_{i,j} \cdot NV_{j,N,t-N-i+1}$$

Nous avons donc :

$$\mathbf{SP_{i,j,t} = SP_{i,j-1,t-1} + C_{i,j} \cdot NV_{j,N,t-N-i+1}} \qquad \textbf{(III.10)}$$

A partir de (III.10) nous obtenons:

$$\mathbf{SP_{i,N,t} = \sum_{i=1}^{N} C_{i,j} \cdot NV_{j,N,t-N-i-(N-j)+1}} \qquad \textbf{(III.11)}$$

En appliquant (III.3) si j=N à (III.11), nous trouvons:

$$\mathbf{NV_{i,N,t} = F\left(\sum_{j=1}^{N} C_{i,j} \cdot NV_{j,N-1,t-2N+j-i}\right)} \qquad \textbf{(III.12)}$$

Si nous considérons que l'origine du temps se situe au moment de l'introduction de la première composante du vecteur à reconnaître et qu'un pas de récurrence dure un temps de 2.N, alors la composante i du vecteur X au pas de récurrence r apparait sur le bord Est du réseau au temps de calcul 2.r.N+i. Nous avons donc:

$$\mathbf{X_{i,r} = NV_{i,N,2rN+i}} \qquad \textbf{(III.13)}$$

D'après (III.12) et (III.13), nous obtenons alors:

$$\mathbf{X_{i,r} = F\left(\sum_{j=1}^{N} C_{i,j} \cdot NV_{j,N-1,2(r-1)N+j}\right)} \qquad \textbf{(III.14)}$$

D'après la définition (III.13) de X nous obtenons:

$$\mathbf{X_{i,r} = F\left(\sum_{j=1}^{N} C_{i,j} \cdot X_{j,r-1}\right)} \qquad \textbf{(III.15)}$$

CQFD

Nous venons de démontrer que le réseau modélisé par les équations (III.2), (III.3), (III.4) et (III.5) effectue bien le calcul voulu (III.1). Il faut aussi démontrer que la détection de convergence s'effectue correctement. Nous rappelons que la convergence locale CL_i élaborée dans les cellules diagonales est transmise sur le coté Ouest par le registre NV des cellules inférieures à la diagonale.

En appliquant (III.3) sur (III.13) jusqu'à j=i+1 on obtient :

$$X_{i,r} = NV_{i,i+1,2rN+i+N-i-1}$$

$$\mathbf{X_{i,r} = NV_{i,i+1,2rN+N-1}} \qquad \textbf{(III.16)}$$

En reprenant (III.16) au pas de récurrence précédent on obtient :

$$X_{i,r-1} = NV_{i,i+1-1,2(r-1)N+N-1}$$

$$= M_{i,i-1,2(r-1)N+N} \qquad \text{d'après (III.4) pour i=j}$$

$$= M_{1,i-1,2(r-1)N+N+i-1} \qquad \text{d'après (III.4) jusqu'à i=1}$$

$$= D_{1,i-1,2(r-1)N+N+i} \qquad \text{d'après (III.5) pour i=1}$$

$$= D_{N,i-1,2(r-1)N+N+i+(N-1)} \qquad \text{d'après (III.5) jusqu'à i=N}$$

$$= D_{N,i,2rN+i-1} = M_{N,i,2rN+i} \qquad \text{d'après (III.4) pour i=N}$$

$$= M_{i+1,i,2rN+i+(N-(i+1))}$$

$$\mathbf{X_{i,r-1} = M_{i+1,i,2rN+N-1}} \qquad \textbf{(III.17)}$$

En appliquant (III.16) et (III.17) à (III.3) pour i=j, on obtient avec t=2rN+N:

$$\mathbf{NV_{i,i,t} = (X_{i,r-1} \otimes X_{i,r}) \quad avec\ t=2rN+N} \qquad \textbf{(III.18)}$$

En appliquant (III.3) sur (III.18) jusqu'à j=1 on obtient :

$$\mathbf{NV_{i,1,t+i-1} = (X_{i,r-1} \otimes X_{i,r}) \quad avec\ t=2rN+N} \qquad \textbf{(III.19)}$$

La convergence locale est évaluée au temps t=2rN+N. L'élaboration de la convergence globale se fait sur le bord Ouest par une fonction "ET" systolique :

$$\mathbf{CV_{i,t} = CV_{i-1,t-1} \wedge NV_{i,1,t-1}} \qquad \textbf{(III.20)}$$

D'après (III.19) et (III.20) on a :

$$\mathbf{CV_{i,t+i} = CV_{i-1,t+i-1} \wedge (X_{i,r-1} \otimes X_{i,r}) \quad avec\ t=2rN+N} \qquad \textbf{(III.21)}$$

A partir de (III.21) et (III.19) nous obtenons :

$$CV_{N,t+N} = \bigwedge_{i=1,N} (X_{i,r-1} \otimes X_{i,r}) \quad \text{avec } t=2rN+N \qquad \text{(III.22)}$$

La convergence globale du vecteur au pas de récurrence r est évaluée par le dispositif de convergence qui délivre le signal CV au temps 2(r+1)N i.e. au moment où va sortir au bord Est la nouvelle composante $X_{1,r+1}$.

Nous venons de montrer formellement que le réseau défini effectue bien le calcul voulu. Il permet d'effectuer la procédure de reconnaissance et de détecter la convergence.

I.5. Amélioration des performances

I.5.1. Traitement "pipeline"

Dans une architecture parallèle, il est fondamental d'analyser le taux d'activité des processeurs. Avant d'aller plus loin dans la description de l'architecture, il faut s'assurer de l'efficacité de l'implantation.

Un pas de récurrence de l'algorithme calculé, consiste en N sommes de N termes chacune. N^2 calculs sont donc nécessaires. Notre réseau comporte N^2 cellules et effectue un pas de récurrence en 2N pas. Le taux d'activité est donc très faible : 1/(2N). En observant les figures III.1 à III.7, on peut s'apercevoir que toutes les cellules n'effectuent pas de calcul significatif à chaque pas.
Cependant, nous allons montrer que plusieurs vecteurs peuvent être traités les uns après les autres selon un mode "pipeline". Si nous reprenons la démonstration du paragraphe précédent, nous pouvons introduire un indice supplémentaire v qui représente le rang d'introduction du vecteur dans le

réseau. Les équations qui décrivent le fonctionnement du réseau ne sont pas modifiées. Si nous considérons qu'un pas de récurrence dure un temps 2.N, alors la composante i du v^{ème} vecteur X au pas de récurrence r apparait sur le bord Est du réseau au temps de calcul 2.r.N+i+v si v est inférieur à 2N. L'équation (III.13) devient alors:

$$X_{i,r,v} = NV_{i,N,2rN+i+v} \quad \text{avec } v \leq 2N \qquad \textbf{(III.23)}$$

D'après (III.12) nous obtenons alors:

$$X_{i,r,v} = F\left(\sum_{j=1}^{N} C_{i,j}\, NV_{j,N-1,2(r-1)N+j+v}\right) \qquad \textbf{(III.24)}$$

D'après la définition (III.13) de X nous obtenons:

$$X_{i,r,v} = F\left(\sum_{j=1}^{N} C_{i,j}\, X_{j,r-1,v}\right) \quad \text{avec } v \leq 2N \qquad \textbf{(III.25)}$$

De même, nous pouvons montrer que le calcul de la convergence s'effectue correctement avec v vecteurs dans le réseau si v est inférieur à 2N. On obtient de (III.19):

$$NV_{i,1,t+i-1+v} = (X_{i,r-1,v} \otimes X_{i,r,v}) \quad \text{avec } t=2rN+N \quad \text{et} \quad v \leq 2N \qquad \textbf{(III.26)}$$

On obtient alors en appliquant (III.28) sur (III.20) :

$$CV_{N,t+N+v} = \bigwedge_{i=1,N} (X_{i,r-1,v} \otimes X_{i,r,v}) \quad \text{avec } t=2rN+N \quad \text{et} \quad v \leq 2N \qquad \textbf{(III.27)}$$

Ce qui démontre que 2N vecteurs peuvent être traités en parallèle dans le réseau. Le taux d'activité atteint alors la valeur maximale de 100%.

I.5.2. Gestion des Entrées/Sorties

Reconnaître 2N vecteurs en parallèle signifie qu'un vecteur X_v peut converger alors qu'il reste 2.N-1 vecteurs dans le réseau, n'ayant pas encore convergé.

Cette situation conduit à l'élaboration de plusieurs méthodes de gestion des Entrées/Sorties pour la réinsertion de nouveaux vecteurs :

- gestion locale : dès qu'un vecteur a convergé, il faut l'extraire. Cette opération s'effectue en N unités de temps : temps nécessaire au calcul de la convergence globale sur le bord Ouest et à l'arrivée de la première composante du vecteur reconnu sur le bord Est. On insère alors un nouveau vecteur qui prend la place de celui que l'on vient de sortir. Pour implanter une telle méthode, il suffit de relier le signal CV au signal INTRO. Cette gestion est optimum car elle effectue qu'un seul pas de récurrence supplémentaire sur un vecteur reconnu (temps d'évaluation de la convergence globale). Malheureusement, elle présente l'inconvénient de ne pas respecter en sortie l'ordre d'entrée des vecteurs. Cette particularité peut être rédhibitoire pour certaines applications comme la reconnaissance de la parole ou de l'écriture. L'environnement hôte doit alors rétablir l'ordre des vecteurs.
- gestion globale : plus simple, elle consiste à attendre la convergence des 2.N vecteurs présents dans le réseau. Dans ce cas, il faut ajouter un dispositif de comptage des vecteurs ayant convergé par scrutation du signal CV, et délivrer un signal (INTRO) si l'on atteint la valeur 2.N. Les vecteurs sont alors extraits et 2N nouveaux vecteurs sont introduits dans le réseau. Cette méthode n'est pas optimale. En effet, on continue à effectuer des calculs sur des vecteurs dont la reconnaissance à déjà convergée. Mais l'ordre d'insertion des vecteurs est respectée lors de la sortie des résultats.
- gestion probabiliste : cette méthode consiste à attendre un nombre de pas d'itération suffisant pour assurer que la convergence de tous les vecteurs est un événement quasi-probable. Le dispositif de convergence n'intervient donc plus dans la décision d'effectuer une Entrée/Sortie. Les vecteurs sont extraits lorsqu'ils sont supposés avoir convergé. Pour mettre en œuvre la

gestion probabiliste, il faut trouver une borne supérieure du nombre d'itérations. La borne fournie par les études théoriques [CAS88] est très largement supérieure à celle déduite des résultats expérimentaux. On a donc le choix entre perdre du temps ou risquer d'extraire des vecteurs n'ayant pas convergé. Mais cette gestion présente l'avantage de minimiser le contrôle.

La programmation du signal INTRO permet au dispositif d'Entrée/Sortie, présenté à la figure III.10, d'implanter les trois gestions précédentes.

Le choix entre ces trois gestions doit se faire avec précaution. En effet, les applications qui nécessitent la conservation, en sortie, de l'ordre d'entrée des vecteurs, doivent éviter la gestion locale. Cette contrainte particulière augmente la sophistication de l'interface "réseau/ordinateur hôte". Sans contrainte sur l'ordre des vecteurs, la gestion locale est intuitivement plus avantageuse. Cependant, si le réseau est plus rapide que l'interface, il doit être ralenti afin de laisser le temps à celle-ci de récupérer les données. Or, le choix d'une gestion locale implique un étalement des opérations d'Entrée/Sortie alors qu'une gestion globale ou probabiliste permet de concentrer dans le temps ces opérations qui dépendent des performances de l'environnement.

Si les dispositifs externes sont plus lents que le réseau, une gestion locale peut ralentir plus souvent le réseau et donc être moins avantageuse qu'une gestion globale. A la gestion probabiliste, on préfère la gestion globale car elle est généralement plus rapide. Mais, pour réduire les risques de temps de convergence infini, la gestion globale peut être supervisée par une gestion probabiliste utilisant la borne maximale théorique. On retrouve alors la notion de chien de garde.

Le choix de la gestion d'Entrée/Sortie dépend donc essentiellement de l'application et de la sophistication de l'environnement du réseau.

I.6. Extensibilité de l'architecture

L'architecture présentée est extrêmement régulière. Un réseau systolique carré présente une modularité intrinsèque qui permet la construction de plus grands réseaux par juxtaposition de sous-réseaux comme le montre la figure III.13.

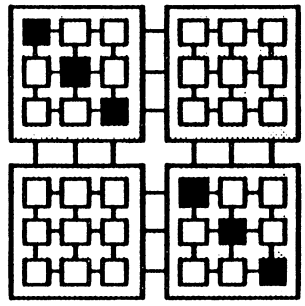


Figure III.13 Composition d'un réseau

Cependant, les cellules diagonales diffèrent des autres cellules par leur mode de communication (cf figure III.12). Seules les cellules diagonales des sous-réseaux constituant la diagonale du réseau doivent donc se comporter comme telles.

Il faut noter que cette particularité peut nuire à la régularité de l'architecture. Nous verrons qu'une conception judicieuse permet d'obtenir des sous-réseaux programmables (contenant une diagonale ou non) et donc de conserver une structure régulière.

I.7. Autres solutions architecturales

L'ensemble d'équations (III.2), (III.3), (III.4) et (III.5) est un Système d'Equations Récurrentes Uniformes (SERU). En effet, nous n'avons que des translations indépendantes des indices. Nous pouvons alors appliquer la méthode de [QUI84] et [JOI87]. Cette méthode de synthèse automatique de réseaux systoliques fonctionne en deux étapes : il faut choisir une fonction de temps T compatible avec les dépendances introduites par les équations, et définir une fonction d'allocation A qui alloue des processeurs différents pour

des tâches concurrentes. Pour obtenir le réseau précédemment décrit, les fonctions T et A sont triviales :

T(i,j,t) = t, A(i,j,t) = (i,j).

Cependant, on peut trouver d'autres fonctions qui, à partir du même système, aboutissent à des implantations différentes.

L'architecture développée précédemment intègre une cellule par élément de la matrice synaptique. Cette matrice dans le cas d'un réseau de Hopfield a la particularité d'être symétrique. On peut tirer parti de cette caractéristique en groupant les cellules qui ont des coefficients symétriques.
Il suffit alors de replier la partie inférieure de la matrice des cellules sur la partie supérieure. On obtient alors un réseau triangulaire comme le montre la figure III.14.

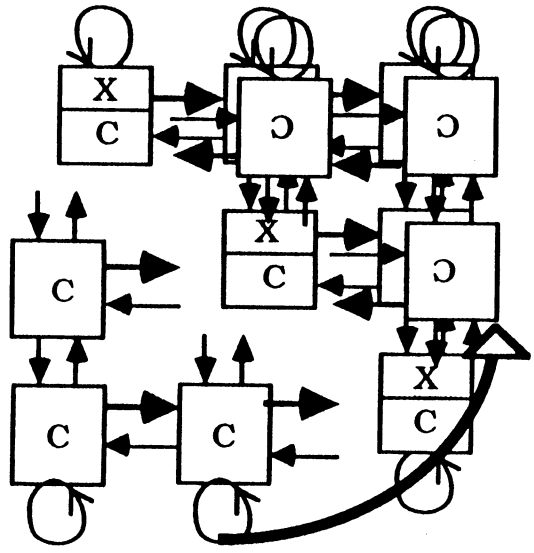


Figure III.14 Réseau triangulaire

Une cellule regroupe alors les calculs concernant $C_{i,j}$ et $C_{j,i}$. Les fonctions de temps et d'allocation qui produisent un tel réseau sont :

T(i,j,t) = Si i≥j alors 2t sinon 2t+1.

A(i,j,t) = Si i≥j alors (i,j) sinon (j,i).

Dans un premier temps, il faut montrer que la fonction de temps respecte les dépendances introduites par le système d'équations : si (i,j,t) dépend de (i',j',t') alors il faut que T(i,j,t)>T(i',j',t').

Démonstration :

Les dépendances du système d'équations peuvent être résumées par le schéma suivant :

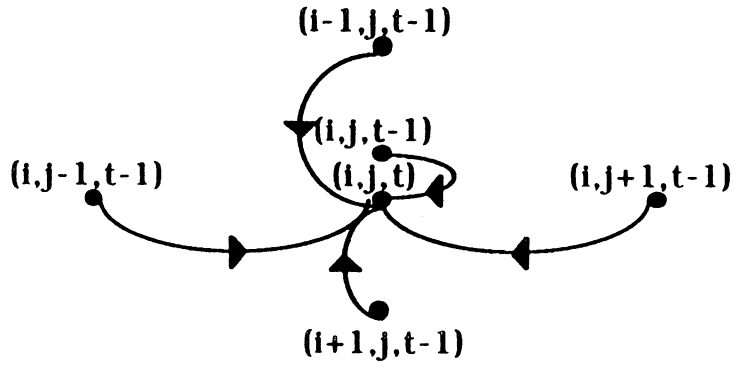


Figure III.15 Dépendances introduites par le SERU

2 cas se présentent :

- Si i ≥ j ⇒ T(i,j,t) = 2t et t' = t-1 d'après la figure ci-dessus
  ⇒ T(i,j,t) > T(i',j',t') car T(i',j',t') = 2(t-1) ou T(i',j',t') = 2(t-1)+1.
- Si i < j ⇒ T(i,j,t) = 2t+1 et t' = t-1
  ⇒ T(i,j,t) > T(i',j',t') car T(i',j',t') = 2(t-1) ou T(i',j',t') = 2(t-1)+1.

CQFD.

Dans un second temps, il faut montrer que ces fonctions n'allouent pas le même processeur à des tâches concurrentes.

Prouvons que pour (i,j,t) ≠ (i',j',t'), si A(i,j,t) = A(i',j',t') alors T(i,j,t) ≠ T(i',j',t').

Démonstration :

Si (i,j,t) ≠ (i',j',t') et A(i,j,t) = A(i',j',t') alors

1er cas : (i = i' et j = j') ⇒ t ≠ t' ⇒ T(i,j,t) ≠ T(i',j',t')

2ème cas : (i = j' et j = i') alors

Si i = j ⇒ i = i' = j =j' et on se retrouve dans le 1er cas.

Si i > j ⇒ i' < j' ⇒ {T(i,j,t) = 2t et T(i',j',t') = 2t'+1} ⇒ T(i,j,t) ≠ T(i',j',t')

Si i < j ⇒ i' > j' ⇒ {T(i,j,t) = 2t+1 et T(i',j',t') = 2t'} ⇒ T(i,j,t) ≠ T(i',j',t')

CQFD.

Cette implantation a pour avantage de pouvoir utiliser la symétrie de la matrice. Les registres nécessaires au stockage de la matrice sont donc presque réduits de moitié. Mais un problème se pose pour la normalisation de la matrice. Dans le cas du réseau carré, nous verrons au paragraphe II.1.2 de ce chapitre qu'une normalisation par ligne est possible : tous les coefficients d'une même ligne sont divisés par la plus grande valeur absolue d'une ligne de la matrice de coefficients (MAXLIG). Cette normalisation par ligne n'est pas utilisable pour le réseau triangulaire car elle casse la symétrie : elle doit se faire sur toute la matrice comme décrit ci-dessous, où MAXMAT est une fonction délivrant la plus grande valeur absolue des éléments de la matrice C :

```
Pour i:=1 à N faire
   Pour j:=1 à N faire
      C(i,j) := C(i,j)/MAXMAT( |C| ).
```

Comparons cette normalisation avec la normalisation par ligne utilisée pour le réseau carré :

∀ i ∈[1,N] MAXMAT(|C|) ≥ MAXLIG(|C(i)|) ⇒

∀ i ∈[1,N] C(i,j)/MAXMAT (|C|) ≤ C(i,j)/MAXLIG(|C(i)|) ⇒

∀ i ∈[1,N] Prec [C(i,j)/MAXMAT(|C|)] ≥ Prec [C(i,j)/MAXLIG(|C(i)|)]

Cette normalisation introduit donc une plus grande erreur que la normalisation par ligne et nécessite plus de bits pour le stockage du coefficient synaptique dans la cellule. La place gagnée est donc très réduite. De

plus, il est peu aisé d'implanter la topologie d'un réseau triangulaire. Enfin, lors de la composition d'un réseau par connexion de plusieurs sous-réseaux (chaque sous-réseau étant un circuit intégré par exemple), un problème d'homogénéité se pose : deux types de sous réseaux doivent être utilisés comme le montre la figure III.16.

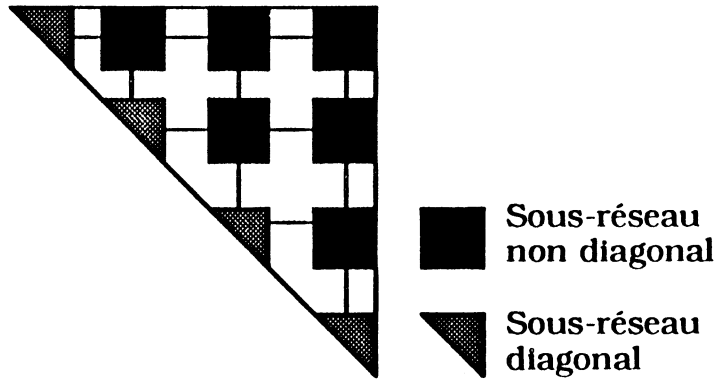


Figure III.16 Composition d'un réseau triangulaire

Cette architecture triangulaire ne présente donc aucun avantage décisif.

Une autre possibilité architecturale est de réduire la matrice de N x N processeurs en un tableau de N processeurs. Les calculs effectués sur une ligne horizontale de notre architecture se font alors dans une cellule comme le montre la figure III.17.

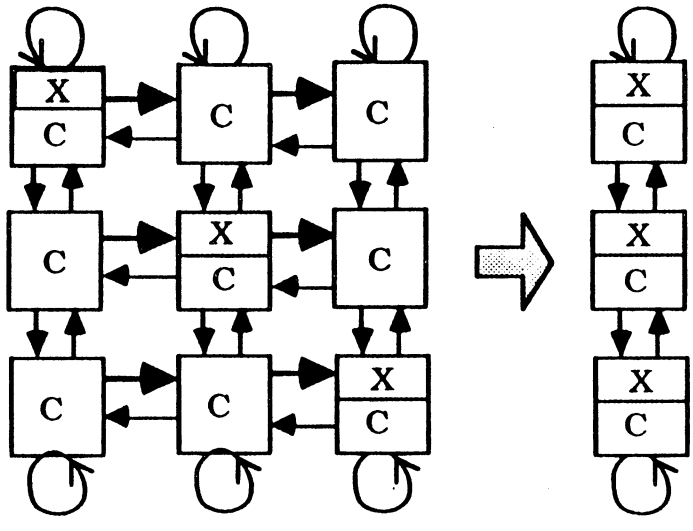


Figure III.17 Projection horizontale

La même méthode que précédemment peut être appliquée avec les fonctions $T(i,j,t) = N.(t-1)+j$ et $A(i,j,t) = i$.

La preuve de "si (i,j,t) dépend de (i',j',t') alors T(i,j,t) > T(i',j',t')" est aisée. D'après les dépendances montrées par la figure III.15, on a :

$$t' = t-1 \text{ et } j' \leq j+1 \Rightarrow T(i',j',t') = N.(t'-1)+j' \leq N.(t-2)+j+1$$
$$= N.(t-1)+j+1-N$$
$$\leq N(t-1)+j \text{ car } j'-j \leq N-1$$
$$= T(i,j,t).$$

CQFD.

Il reste à montrer que "pour $(i,j,t)\neq(i',j',t')$ si $A(i,j,t)=A(i',j',t')$ alors $T(i,j,t)\neq T(i',j',t')$".

Démonstration :

$(i,j,t) \neq (i',j',t')$ et $A(i,j,t) = A(i',j',t') \Rightarrow i=i'$ et ($j\neq j'$ ou $t\neq t'$) $\Rightarrow$

1er cas : $t\neq t'$ et $j=j' \Rightarrow T(i,j,t)-T(i',j',t') = N.(t-t') \neq 0$ car $t\neq t'$

2ème cas : $j\neq j'$ et $t=t' \Rightarrow T(i,j,t)-T(i',j',t') = j - j' \neq 0$ car $j\neq j'$

3ème cas : $j\neq j'$ et $t\neq t' \Rightarrow T(i,j,t)-T(i',j',t') = N.(t-1)+j - N.(t'-1)-j'$
$= N.(t-t')+j-j'$

Montrons par l'absurde que $N.(t-t')+j-j'$ est non nul :

$N.(t-t')+j-j'=0 \Rightarrow N(t-t') = j'-j$

$\Rightarrow N|t-t'| = |j'-j|$ ce qui est impossible car $|t-t'| \geq 1$ et $|j'-j| \leq N-1$

CQFD.

On obtient alors une architecture proche de celle adoptée par [WEI88]. Cette architecture se heurte à un problème d'extensibilité comme on l'a vu dans le chapitre précédent.

L'architecture triangulaire n'apporte aucun avantage décisif, l'architecture linéaire n'est pas facilement extensible : nous conservons donc le réseau carré précédemment défini qui est facilement extensible car chaque processeur intègre une synapse.

II. ARCHITECTURE LOGIQUE DE LA CELLULE

Lors de la spécification architecturale, la cellule a été définie en termes de registres, calculs et communication sur un plan fonctionnel. Nous allons maintenant détailler ces éléments, ce qui nous amène à décrire complètement l'architecture de la cellule.

II.1. Format des opérandes

Dans le réseau SIMD que nous venons de définir, chaque cellule effectue le même calcul :

$$SP_{i,j} = SP_{i,j-1} + D_{i-1,j} \cdot C_{i,j} \text{ avec } SP_{i,0} = 0 \text{ and } D_{0,j} = M_{1,j}. \quad \text{(III.28)}$$

Comme nous l'avons vu précédemment, les réseaux de Hopfield sont utilisés avec des neurones issus du modèle de Mac Culloch et Pitt. Leurs valeurs peuvent être -1 ou +1. L'opération se résume alors à :

$$SP_{i,j} = SP_{i,j-1} \pm C_{i,j}. \quad \text{(III.29)}$$

Nous avons choisi de coder l'activité des neurones comme suit :

- 1 pour coder (-1),
- 0 pour coder (+1).

Avec un tel codage, appliquer la fonction de seuil se résume à prendre le bit de signe.

Les registres NV, D et M sont donc des registres un bit et l'unité arithmétique et logique est réduite à un simple additionneur/soustracteur.

II.1.1. Coefficient synaptique

Le coefficient synaptique stocké dans le registre C, est fourni par une règle d'apprentissage sous forme d'un réel flottant. Afin de réduire la complexité de l'UAL et des registres de calcul, les composantes de la matrice des coefficients sont ramenées à un format fixe sur p bits après avoir subi la normalisation suivante, où MAXLIG est une fonction délivrant la plus grande valeur absolue sur une ligne:

```
Pour i:=1 à N faire
    Pour j:=1 à N faire
        C(i,j) := C(i,j)/MAXLIG(|C(i)|)
```

De cette normalisation par ligne, il résulte :

- une perte de la symétrie,
- une perte de précision sur la matrice.

La perte de la symétrie est sans conséquence sur la phase de reconnaissance. En effet, dans le cas du modèle envisagé, la fonction de seuil est la fonction Signe. Elle a la propriété suivante :

$$\mathbf{F}(a.X) = \mathbf{F}(X) \quad \text{Si } a>0 \quad \Rightarrow \quad X_I = \mathbf{F}\left(\sum_{J=1,N} C_{I,J}X_J / MAXLIG(|C(i)|)\right)$$

$$\Rightarrow \quad X_I = \mathbf{F}\left(1/MAXLIG(|C(i)|) \cdot \sum_{J=1,N} C_{I,J}X_J\right)$$

$$\Rightarrow \quad X_I = \mathbf{F}\left(\sum_{J=1,N} C_{I,J}X_J\right)$$

Une précision suffisante sur les coefficients synaptiques est essentielle afin de leur conserver les propriétés données par la loi d'apprentissage. La courbe présentée à la figure 6 du chapitre précédent nous a montré que la perte de précision peut entraîner un "oubli" important des prototypes appris lors de l'apprentissage. La détermination du nombre de bits suffisant à mémoriser le coefficient synaptique est délicate, car elle dépend de la loi d'apprentissage utilisée et de l'ensemble des prototypes appris. Les résultats des simulations

effectuées avec la loi de la projection sont très variables et difficiles à interpréter. Toutefois, la perte de précision est sans conséquence tant que:

$$\mathbf{p \geq Log_2(N)+1} \qquad \textbf{(III.30)}$$

Une telle précision a permis d'obtenir des performances équivalentes à celles obtenues avec une matrice à coefficients réels double précision.

II.1.2. Registre sommes partielles

D'après (III.29), le format du registre SP est directement lié au format du registre C et au nombre de neurones du réseau car la somme partielle est, au maximum, la somme totale des valeurs absolues de tous les coefficients d'une ligne de la matrice synaptique (N coefficients par ligne). Soit UTIL (C) le nombre de bits utiles dans le registre C, SP doit mémoriser le résultat de N sommes de UTIL (C) bits. On a donc :

$$\mathbf{UTIL\ (SP) \leq UTIL\ (C) + Log_2(N)} \qquad \textbf{(III.31)}$$

D'après (III.30), l'intégration du registre C sous la forme d'un registre 8 bits (UTIL(C) $\leq$ 8) permet de construire un réseau de 128 neurones aux performances maximales. Le registre SP doit alors contenir au moins 15 bits. En effet, d'après (III.31) nous avons :

$$\begin{aligned} \text{UTIL (SP)} &\leq \text{UTIL (C)} + Log_2(N) \\ &\leq 8 + Log_2(128) \\ &\leq 8 + 7 = 15 \end{aligned}$$

SP est alors intégré sous forme d'un registre 16 bits.

II.2. Mode Série/Parallèle

Les 16 bits du registre SP peuvent être transmis de cellule en cellule soit en parallèle soit en série. Le mode parallèle permet d'effectuer l'évaluation d'une somme partielle en un cycle d'horloge mais nécessite un opérateur arithmétique travaillant sur 16 bits. De plus, le mode parallèle multiplie par 16 le nombre de pattes du circuit intégré.

Par contre, les transmissions en mode série, plus lentes, réduisent les connexions entre cellules et les plots entre circuits.

Il s'agit donc de trouver un compromis entre la complexité et la rapidité du circuit.

Comme le circuit offre un temps de réponse et un débit de traitement largement suffisants par rapport aux ordinateurs susceptibles de l'exploiter, le mode parallèle est donc à éviter.

Avec un mode purement série (ou "*bit-serial*") comme celui adopté par [BUT88], chaque cellule calcule et mémorise un seul bit de la somme partielle. Le calcul d'une somme partielle est répartie sur plusieurs cellules. Ce mode n'autorise pas le traitement en pipeline de 2N vecteurs et les performances du circuit sont moindres.

Un mode mixte, série/parallèle a donc été choisi : les cellules calculent en parallèle les sommes partielles qu'elles stockent temporairement, mais les transmissions des sommes partielles entre cellules se font sur un mode série. Cette technique permet d'optimiser les performances du circuit tout en respectant les contraintes liées à l'intégration.

II.3. Partie calcul de la cellule

Au vu de la faible complexité des calculs effectués par la cellule, une approche classique partie opérative/partie contrôle (PC/PO) peut paraître disproportionnée. Cependant une partie contrôle réduite doit prendre en charge la sérialisation des calculs et des transmissions de SP : le séquenceur. Un compromis entre la simplicité du séquenceur et de la PO doit être trouvé.

Les éléments de la partie opérative sont :

- des registres de calcul (SP et C),
- un opérateur arithmétique,
- des registres de communication (D, M, NV).

La structure des registres de calcul va avoir une influence directe sur la complexité de la partie opérative et de son contrôle. Différentes solutions ont été étudiées pour les registres SP et C en termes de complexité et de souplesse. En effet, du choix de leur structure va directement dépendre le contrôle de la sérialisation. Par souci de régularité, les mêmes structures sont adoptées pour les registres SP et C.

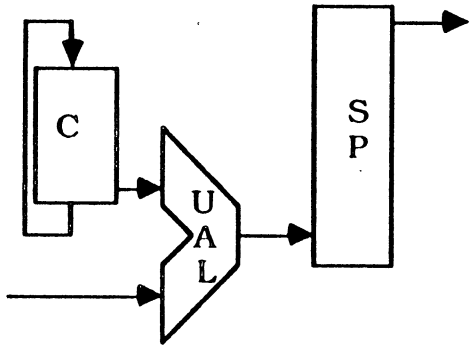


Figure III.18 Architecture 1

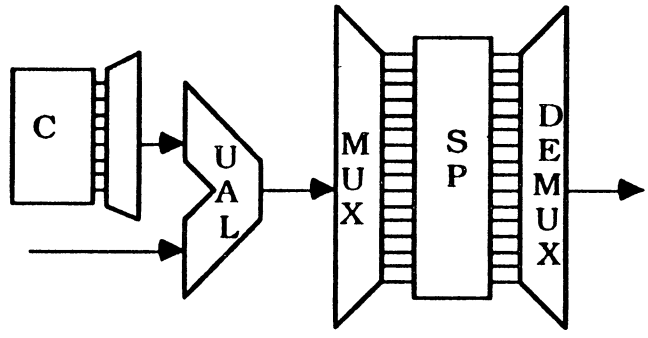


Figure III.19 Architecture 2

- Première solution : les registres SP et C sont des registres à décalage de 16 bits. L'UAL stocke toujours son résultat dans le bit 0 de SP et le bit 15 est envoyé à la cellule voisine pour traitement. L'avantage de cette structure est l'absence de séquenceur. En effet, une simple horloge est nécessaire : c'est l'ordre de décalage des registres.

  Cependant, cette simplicité de contrôle entraîne une augmentation du nombre de transistors de la partie calcul : les registres doivent être du type maître/esclave. D'autre part, si SP est intégré sous la forme d'un registre à décalage P bits (P vaut 16 d'après (III.31)), P cycles de calcul sont obligatoires quelque soit le nombre de bits utiles de SP. Une architecture plus souple devrait permettre d'optimiser le nombre de cycles de calcul en fonction du nombre de bits utiles de SP. Cette solution n'apportant pas cette souplesse, elle ne permet pas d'optimiser le traitement en fonction de l'application.

- Deuxième solution : les bits des registres C et SP sont indépendamment adressables comme le symbolisent les MUX et DEMUX de la figure III.19. Les registres C et SP ne sont plus des registres à décalage mais des registres simples. Le nombre de transistors par élément à mémoriser est donc divisé par 2. Par contre, le séquenceur augmente en complexité car il doit gérer cet adressage.

La dernière solution est la plus avantageuse lorsqu'on remarque que le séquenceur peut être partagé entre plusieurs cellules. En effet, en mode série/parallèle, les cellules travaillent en parallèle sur le même bit.
Le séquenceur va donc coder le rang du bit qui est traité : 16 états sont nécessaires car on a besoin d'un état par bit du registre SP. Deux solutions de codage apparaissent :

- un codage direct de ces états ne requiert que 4 bits mais nécessite un décodage au niveau de chaque registre ou de chaque cellule,
- un codage 1 parmi 16 permet un contrôle direct de chaque registre.

Ce dernier est alors préférable car l'augmentation de complexité n'est sensible qu'au niveau du séquenceur, commun à plusieurs cellules.

Le codage est réalisé par un registre à décalage de 16 bits, contrôlé par un signal de décalage et un signal RAZ. Le signal RAZ permet l'initialisation du registre dans son état initial : un "1" dans le premier bit du registre et des "0" dans les 15 autres. Par le biais de ce signal, on peut définir le début d'un calcul. Le signal de décalage permet de passer à l'état suivant. Le calcul peut se limiter au nombre de bits significatifs de SP en agissant sur le signal RAZ.
La solution adoptée a donc comme avantage supplémentaire d'apporter la souplesse qui manquait à la première solution : optimiser le temps de calcul en fonction du nombre de neurones du réseau.

Elle nécessite pour sa mise en œuvre, deux horloges :

- HC : horloge haute fréquence (High frequency Clock) définit le passage à l'état suivant. Elle contrôle la transmission des bits de SP et le signal de décalage du séquenceur.
- LC : horloge à basse fréquence (Low frequency Clock) définit le début d'un calcul dans une cellule. Elle sert à réinitialiser l'opérateur arithmétique et à positionner le séquenceur dans son état initial. Elle participe donc à l'élaboration du signal RAZ du registre à décalage (RAZRAD).

Le rapport entre les fréquences de ces horloges va définir le nombre de bits calculés pour chaque opération, c'est à dire le nombre de pas de calcul nécessaires pour évaluer tous les bits utiles de SP. On a alors :

$$\frac{\textbf{freq (HC)}}{\textbf{freq (LC)}} = \textbf{UTIL (SP)} + 1 \qquad \textbf{(III.32)}$$

Pour UTIL (SP) = 4, on obtient les chronogrammes suivants :

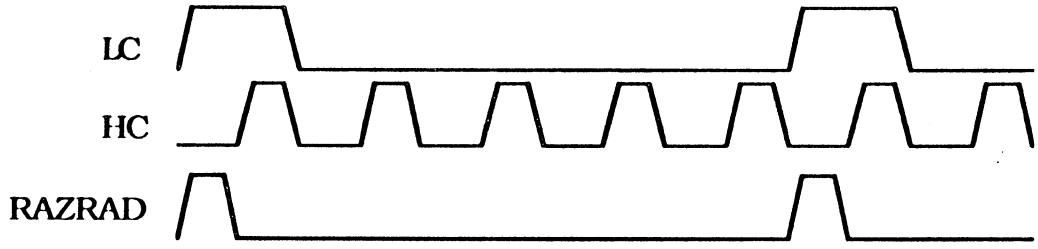


Figure III.20 Signaux du séquenceur pour 4 bits utiles dans SP

La sérialisation réduit la complexité de l'UAL qui devient un simple additionneur/soustracteur 1 bit travaillant en complément à 2. D'après (III.29), l'opérateur est stocké dans le registre D. L'UAL reçoit deux opérandes : le registre interne C et la valeur de $SP_{i,j-1}$ transmise par la cellule Ouest.

Une barrière temporelle est nécessaire pour le bon fonctionnement de l'UAL. Elle peut se trouver soit avant l'opérateur de calcul, soit après. Les deux cas

sont parfaitement symétriques et la première solution a été adoptée. La valeur est stockée dans le registre 1 bit SPin.

Le séquenceur sélectionne le bit C_i nécessaire au traitement et l'UAL effectue le calcul SPin + C_i (respectivement SPin - C_i) si le registre D vaut 0 (respectivement 1). Le résultat de l'opération est stocké dans le bit SP_i sélectionné par le séquenceur.

L'additionneur/soustracteur 1 bit travaillant en complément à 2 est réalisé de façon classique comme le montre la figure III.21.

Le code opération D sélectionne le bit C_i ou son complément. En complément à 2, la retenue doit être initialisée à 1 pour une soustraction et à 0 pour une addition. Aussi, la retenue est initialisée avec le code opération D.

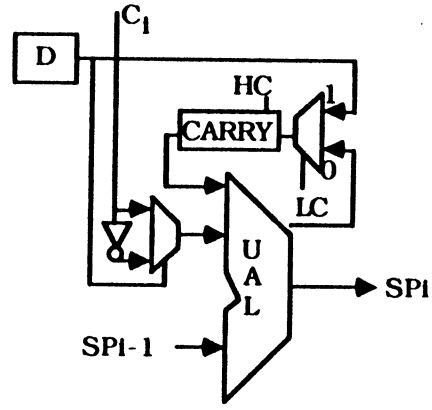


Figure III.21 Unité Arithmétique et Logique

D contient le code opération de l'UAL. Si D vaut 0 (respectivement 1) on fait une addition (respectivement une soustraction). Au début d'un calcul (LC haut), la retenue est mise à zéro (respectivement à un). La valeur initiale de la retenue est chargée par un multiplexeur qui, au début du calcul, affecte la valeur du registre D à la retenue. Ensuite, les bits non complémentés (respectivement complémentés) de C sont ajoutés. Le choix entre C_i et son complément est effectué par un multiplexeur contrôlé par D. L'opérateur est donc un additionneur 1 bit.

Le registre SP va transmettre bit à bit sa valeur vers la cellule voisine à l'Est pour qu'elle la traite. Mais, afin de bien concevoir la cellule qui servira de base à un réseau de plusieurs cellules, nous devons anticiper les problèmes liés à l'interconnexion de plusieurs circuits.

Cette considération nous avait déjà amenés à préférer un mode de communication sériel entre les cellules pour limiter le nombre de plots. La communication entre deux cellules peut donc se faire au travers des plots d'Entrée/Sortie des circuits. Le registre SP doit alors être capable de charger sa sortie et le système de communication entre circuits. Afin de réduire les délais de transmission qui pénaliseraient les performances globales du circuit, la lecture de la valeur à transmettre est anticipée dans chaque cellule. Pendant que le $i^{ème}$ bit est traité, le $i+1^{ème}$ est stocké dans le registre SPout. Ainsi, le registre 16 bits SP est un registre simple. Les chargements de SPin et de SPout sont déphasés : SPin est chargé sur le niveau bas de HC tandis que SPout est chargé sur le niveau haut.

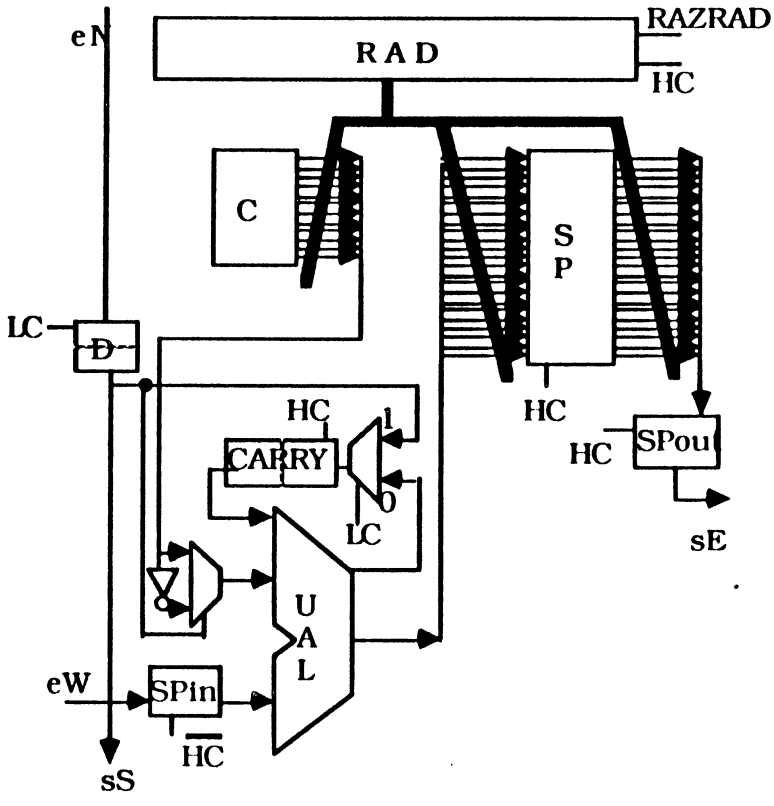
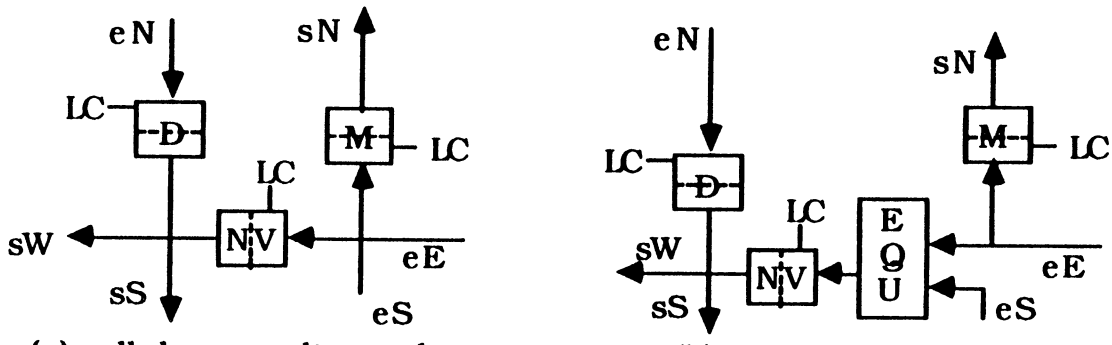


Figure III.22 Partie calcul

II.4. Partie communication de la cellule

La partie communication assure la transmission des valeurs entre cellules voisines. Dans la description de l'architecture fonctionnelle (cf figure III.12), deux types de communication se distinguent.



(a) cellules non diagonales (b) cellules diagonales

Figure III.23 Communications

La figure III.23a présente l'implantation des communications dans les cellules non diagonales. Les valeurs D, M et NV sont stockées dans des registres Maître/Esclave. Le transfert a lieu au début du calcul et les valeurs de D, M et NV sont donc échantillonnées sur l'horloge LC. Dans les cellules diagonales, l'ancienne valeur de X arrive par le coté Sud (eS) et la nouvelle valeur par le coté Est (eE). La convergence locale est alors élaborée en comparant eE et eS par un comparateur câblé EQU. Le résultat est ensuite propagé vers l'Ouest via le registre NV.

Le rebond des composantes sur la diagonale, c'est à dire l'affectation nécessaire au changement de pas de récurrence, est assuré par le chargement du registre M avec la valeur arrivant de l'Est. La partie communication des cellules diagonales est représentée en figure III.23b.

Afin que la diversité des modes de communication entre les cellules diagonales et les autres ne nuise pas à la régularité du réseau, le typage des cellules diagonales est programmable.

La partie communication doit alors intégrer une programmation du type de la cellule : diagonale ou non. Cette différenciation est assurée par l'insertion de deux multiplexeurs (cf figure III.24) contrôlés par un signal global DIAG.

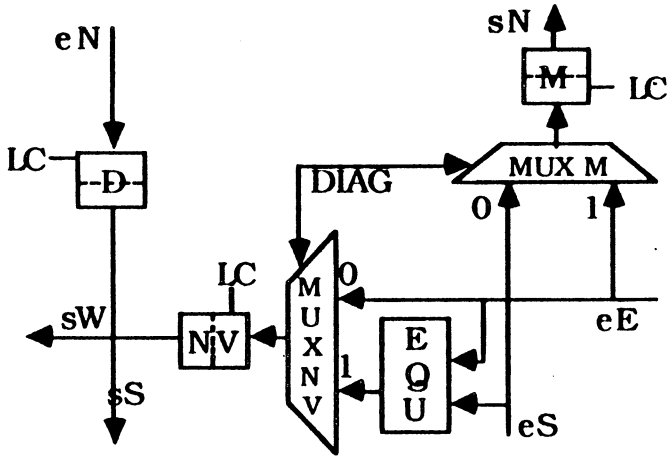


Figure III.24 Partie communication

Comme le montre la figure III.25, il faut différencier les cellules diagonales des sous-réseaux non diagonaux (CDND) des cellules diagonales des sous-réseaux diagonaux (CDD). Seules ces dernières doivent se comporter comme des cellules diagonales. Le typage des sous-réseaux dans la figure III.13 s'effectue par le signal DIAG. Une cellule diagonale d'un sous-réseau peut se comporter :

- soit comme une cellule diagonale (CDD) si le signal DIAG est à 1,
- soit comme une cellule normale (CDND) si le signal DIAG est à 0.

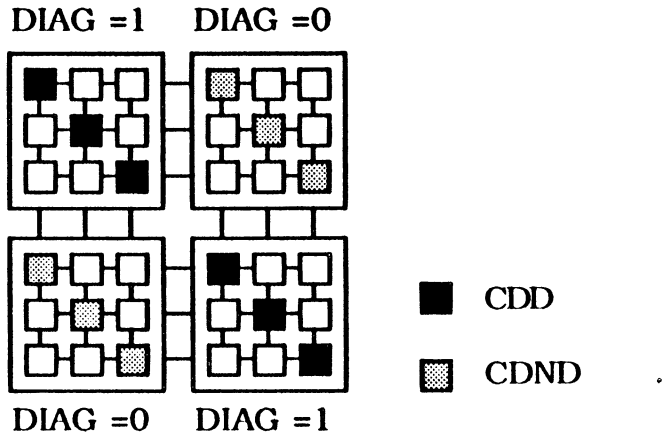


Figure III.25 Typage des sous-réseaux

II.5. Architecture complète de la cellule

Toutes les fonctionnalités de la cellule sont présentées dans l'architecture suivante :

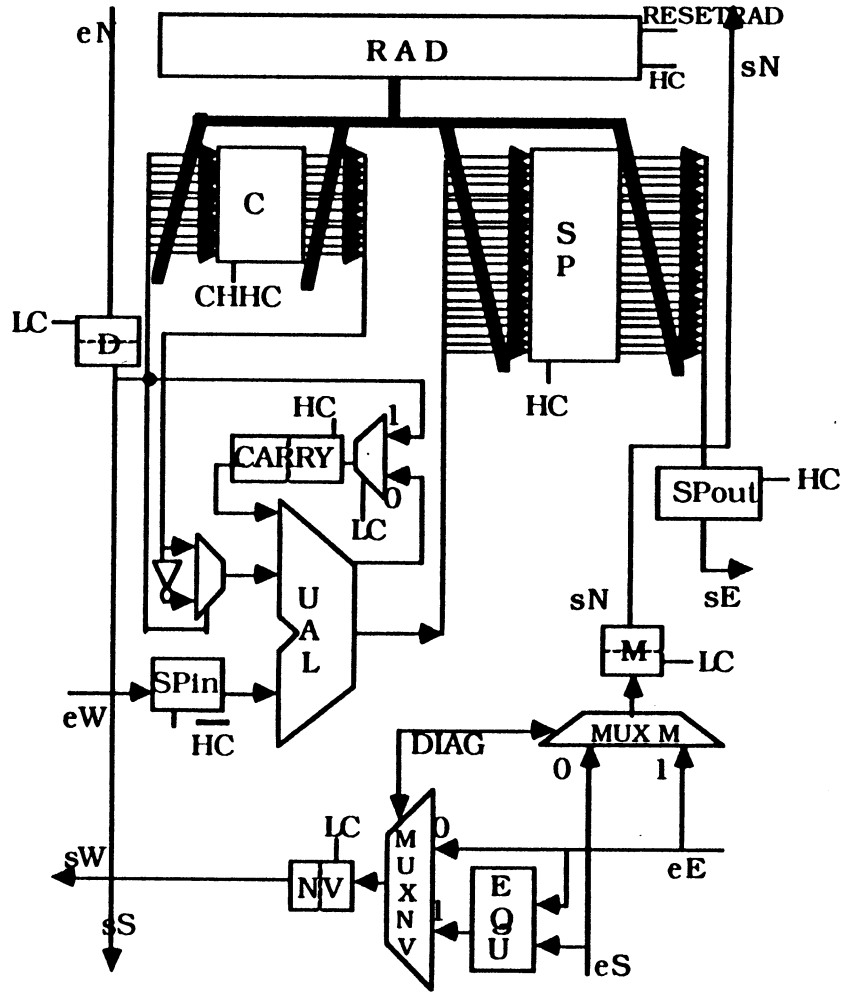


Figure III.26 Architecture de la cellule

III. ARCHITECTURE LOGIQUE DU RESEAU

III.1. Dispositif d'Entrées/Sorties

III.1.1. Principe de la "fermeture Eclair"

La notion de "fermeture Eclair" a déjà été introduite dans la figure III.10. Ce dispositif, situé sur le bord Est du réseau permet d'effectuer les

Entrées/Sorties et le rebouclage. Les sommes pondérées correspondant aux composantes d'un même vecteur apparaissent décalées dans le temps sur le bord Est. Pour un vecteur donné, nous observons :

- à t_0, l'apparition de la première somme partielle sur le bord Est,
- à t_0+1, l'apparition de la deuxième somme partielle sur le bord Est,

...

- à $t_0+(N-1)$, l'apparition de la dernière somme partielle sur le bord Est.

La "fermeture Eclair" a été implantée afin de tenir compte de ce décalage et d'effectuer soit une Entrée/Sortie, soit un rebouclage. Le signal de programmation INTRO est injecté dans le registre à décalage (noté "r" dans la figure III.27) qui s'étend sur le bord du réseau. Chaque élément de ce registre contrôle un multiplexeur (noté "m" dans la figure III.27) qui sélectionne soit la composante d'un nouveau vecteur (opération d'entrée) soit le résultat du seuillage de la somme partielle (opération de rebouclage).

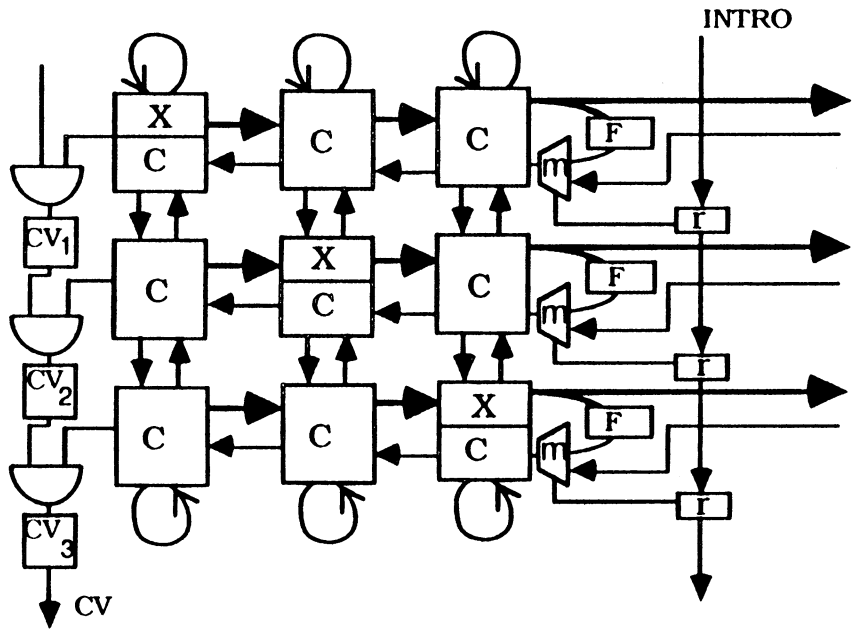


Figure III.27 Dispositif d'Entrée/Sortie

Lors d'une opération d'Entrée/Sortie, la fonction de seuil n'est pas appliquée afin de transmettre la somme pondérée dans sa totalité.

III.1.2. Chargement d'un vecteur

Le fonctionnement de la "fermeture Eclair" est détaillé pour l'opération de chargement des vecteurs.

L'hypothèse sur l'origine des temps posée pour l'équation (III.13) sous-entend que les vecteurs sont introduits dans les registres NV des cellules du bord Est : $NV_{I,N}$. Les 2N vecteurs sont présentés comme le montre la figure suivante.

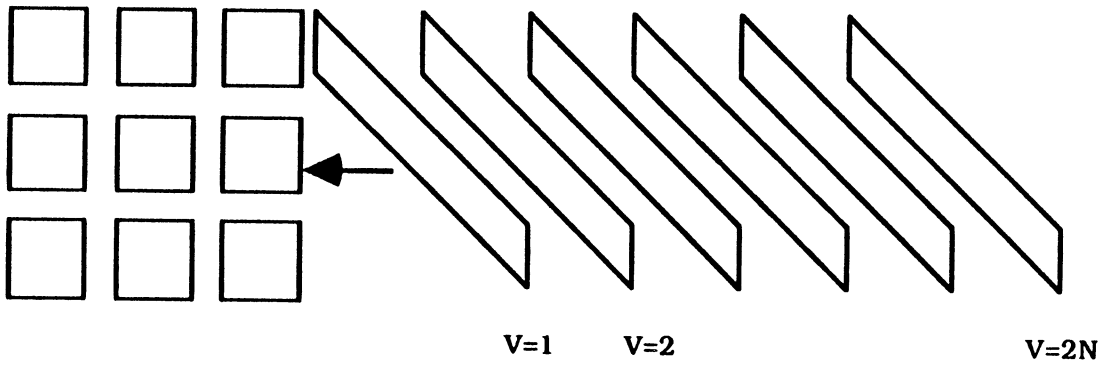


Figure III.28 Présentation de 2N vecteurs au réseau

Afin de réaliser cette opération de chargement, il faut programmer le dispositif d'Entrée/Sortie. Pour cela, INTRO est forcé à "1" pendant les 2N premiers pas seulement (ici 2N = 6), puis remis à "0". La figure III.29 montre l'évolution de la "fermeture Eclair" pendant l'opération de chargement.

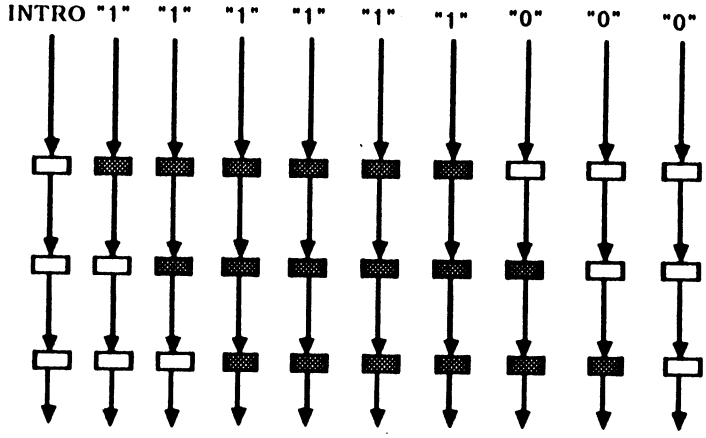


Figure III.29 "Fermeture Eclair" pendant le chargement d'un vecteur

- t0 à t6 : INTRO doit rester à "1" pendant les 2N premiers pas de l'introduction pour empêcher le rebouclage et autoriser l'entrée des nouvelles composantes.
- t7 à t9 : après le pas 2N, il faut progressivement fermer "la fermeture Eclair" : INTRO est mis à zéro. Au pas 2N+i, pour i=1 à N, "la fermeture Eclair" permet le rebouclage dans la partie supérieure du réseau (pour les cellules $C_{x,N}$ pour x=1 à i) et l'entrée dans la partie inférieure (pour les cellules $C_{x,N}$ pour x=i+1 à N).
- après t9: la "fermeture Eclair" se ferme après l'entrée de la dernière composante du dernier vecteur, c'est à dire au pas 3N.

III.2. Fonction de seuil

La fonction de seuil doit être appliquée aux sommes partielles qui atteignent le bord Est. Avec le modèle de Mac Culloch et Pitt, les neurones prennent comme valeurs -1 et +1 et la fonction de seuil est la fonction de signe.

Avec le codage choisi (0 pour coder (+1) et 1 pour coder (-1)), il suffit de réinjecter le bit de signe de SP (bit le plus significatif). La transmission de SP se fait du bit de poids faible au bit de poids fort : le bit de signe est donc le dernier transmis. Il suffit alors de reboucler tous les bits de SP dans le registre 1 bit NV et seule la dernière valeur (le bit de signe de SP) subsistera dans NV.

La fonction de seuil n'est pas appliquée aux sommes pondérées qui sortent. C'est le système hôte qui se charge d'appliquer la fonction de seuil aux sommes pondérées sortantes. Ce dispositif présente deux avantages :

- le test est facilité, car la connaissance de la somme pondérée exacte permet de vérifier le bon fonctionnement des opérateurs de calcul,
- la sortie de la somme pondérée assure la cascadabilité des circuits.

III.3. Chargement des coefficients

Pour compléter la définition architecturale, le chargement de la matrice des coefficients doit être prévu. Chaque cellule doit contenir un coefficient synaptique dans le registre 8 bits C.

Ces coefficients emprunteront l'interface d'Entrée/Sortie précédemment décrite. Présentés aux cellules du bord Est comme le montre la figure suivante, les bits des coefficients $C_{i,j}$ vont s'acheminer vers leur cellule respective en transitant par les registres NV, M puis D (Cf figure III.30).

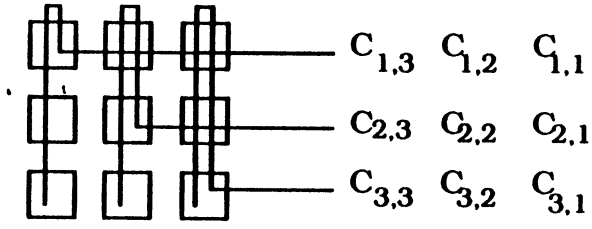


Figure III.30 Chargement des coefficients

Grâce à la sérialisation du calcul, le chargement des coefficients peut se faire bit à bit. Les cellules recevant toutes le $i^{ème}$ bit de leur coefficient respectif via le registre D, on doit pouvoir charger D dans le $i^{ème}$ bit du registre C. De façon à ne pas modifier le séquenceur pour cette phase d'initialisation, les cellules effectuent des calculs non significatifs et le séquenceur contrôle successivement la modification et la lecture de chacun des bits des registres C et SP.

En mode de calcul normal, le chargement de C est inhibé par le signal CHHC qui n'est pas valide.

Pendant la phase d'initialisation, le signal de chargement CHHC doit alors être synchronisé avec le séquenceur et doit être valide au $i^{ème}$ pas de calcul pour

charger le 1ème bit de C comme le montre le chronogramme présenté sur la figure III.31.

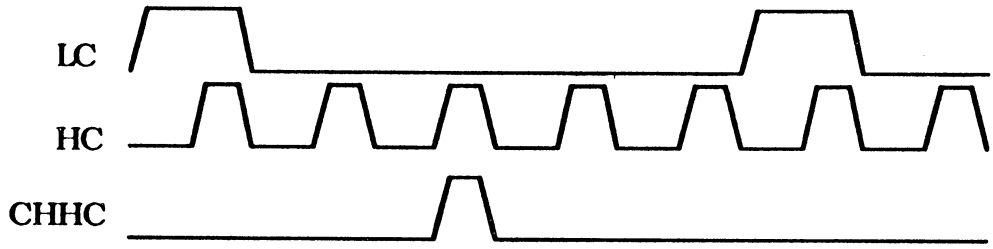


Figure III.31 Chargement du 2ème bit de C

IV . CONCLUSION

L'architecture systolique présentée dans ce chapitre permet d'effectuer l'algorithme récurrent de la phase de reconnaissance. La récurrence est directement implantée sur le réseau par un rebouclage adéquat du flux de données. La circulation des données, telle qu'elle a été décrite, permet aussi d'implanter la détection de convergence à moindre frais (très peu de matériel supplémentaire). Les performances du réseau sont optimales car plusieurs vecteurs peuvent être traités en parallèle dans le réseau. En mode stationnaire, le taux d'activité des processeurs peut atteindre 100%. La faible complexité de ces processeurs nous permet d'envisager une intégration à grande échelle avec un maximum de fiabilité (répétitivité du dessin) et de compacité (nombre de transistors effectifs assez faible).

Ce travail de conception de circuits intégrés est décrit dans les deux prochains chapitres qui présenteront :

- la conception d'une cellule ; l'important est d'obtenir un dessin très compact, cette cellule étant l'élément de base du second circuit,
- la conception d'un sous-réseau ; le but est d'intégrer le plus grand sous-réseau possible sur un circuit, tout en respectant les contraintes fonctionnelles et technologiques.

IV. Projet Aplysie 1 : Intégration d'une cellule

I. DEMARCHE DE CONCEPTION

Une fois les spécifications architecturales définies, nous avons débuté la conception d'une cellule. La démarche de conception que nous avons adoptée est la suivante.

Le plan de masse a été défini en fonction des contraintes fonctionnelles et logiques.

Tous les constituants de base (registre, additionneur, multiplexeur...) ont été définis en termes de transistors. De ces schémas ont été extraites des descriptions permettant la simulation électrique.

Une fois ces éléments de base validés, nous avons poursuivi la description de l'architecture. A chaque étape, nous procédions à une extraction afin de pouvoir effectuer une simulation.

Lorsque le schéma électrique complet de la cellule a été réalisé, nous avons procédé au dessin des masques des éléments de base. De ces dessins ont été extraites des descriptions permettant de comparer les schémas définis préalablement et le dessin. Ces descriptions extraites permettent d'effectuer des simulations plus fines prenant en compte des paramètres électriques tels que les capacités des lignes.

Enfin la cellule a été assemblée, entièrement simulée et vérifiée.

Ce travail de conception a été entièrement réalisé avec la chaîne de C.A.O. Silvar Lisco et le logiciel de simulation électrique ELDO.

II. PLAN DE MASSE

Cette étape essentielle de l'intégration a été guidée par l'aspect cellulaire de l'architecture de ce réseau SIMD. Les cellules fonctionnent de façon synchrone en mode série/parallèle, elles peuvent partager le même séquenceur. Pour pouvoir utiliser au mieux cette particularité, les signaux issus du séquenceur

doivent pouvoir traverser les cellules : nous parlerons de "transparence". Elles constituent donc l'ossature de la cellule sous laquelle les éléments des registres SP et C sont intégrés. La mise en vis-à-vis des trois registres RAD, SP et C définit le plan de masse général présenté par la figure IV.1.

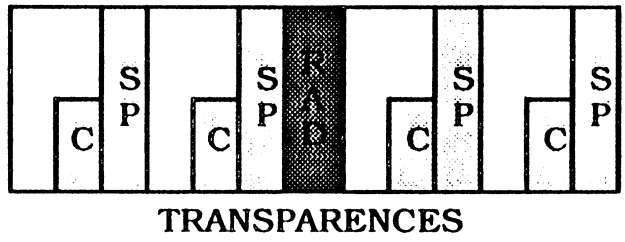


Figure IV.1 Plan de masse général

Les transparences ainsi que les alimentations sont intégrées en niveau métal 2, et les horloges en niveau polysilicium.

Nous allons étudier la réalisation de tous les éléments qui constitue la cellule.

III. LE SEQUENCEUR

Le séquenceur est constitué d'un registre à décalage de 16 bits. Le composant de base est une classique bascule Maître/Esclave.

III.1. Bascule Maître/Esclave

Le schéma classique de la bascule Maître/Esclave, contrôlée par les commandes C1 et C2, est le suivant :

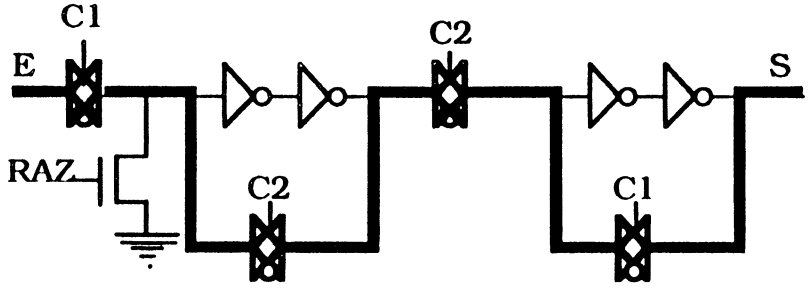


Figure IV.2 Bascule Maître/Esclave

Le fait que ce schéma requiert des commandes non recouvrantes est un inconvénient majeur. En effet, si C1 et C2 se recouvrent, un transfert intempestif risque d'avoir lieu par le chemin en gras de la figure IV.2. Une perte de la mémorisation peut alors survenir. De plus de "vraies" portes de transfert nécessitent la génération du signal de commande et de son complément.

Afin de simplifier le contrôle des registres, des portes de transfert mono-transistor ont été utilisées : le transistor sert d'interrupteur. De tels dispositifs doivent être utilisés avec précaution car ils engendrent des pertes de seuil et une dégénérescence des signaux. En effet, un transistor N (respectivement P) va transmettre correctement un "0" logique (respectivement "1") mais va engendrer une perte de seuil lors du passage d'un "1" logique (respectivement "0").

Pour éviter la distribution délicate d'horloges non recouvrantes, une seule horloge est utilisée. Le type du transistor utilisé comme interrupteur est choisi en fonction de la commande qui le contrôle : une porte de transfert passante sur C1 sera implantée avec un transistor de type N (respectivement de type P) contrôlé par C1 (respectivement sur C2). Un schéma analogue est utilisé pour une porte de transfert passante sur C2 comme le montre la figure suivante :

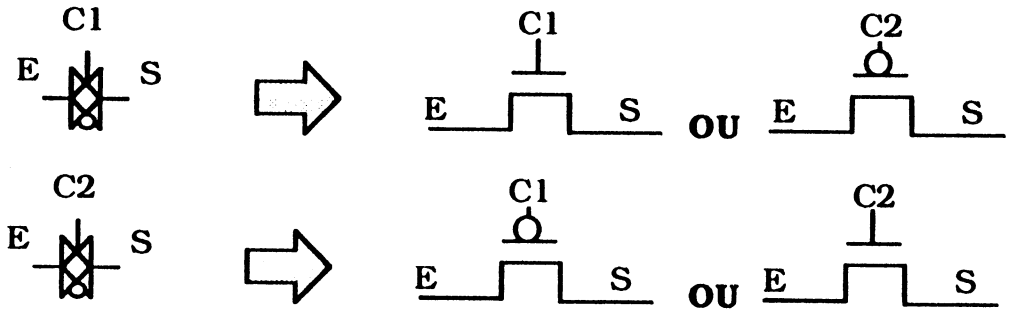


Figure IV.3 Implantation des portes de transfert

Le schéma de la bascule Maître/Esclave présenté figure IV.2 peut être simplifié si l'on utilise de telles portes de transfert. On obtient alors l'un des schémas suivants, selon que C1 ou C2 est générée.

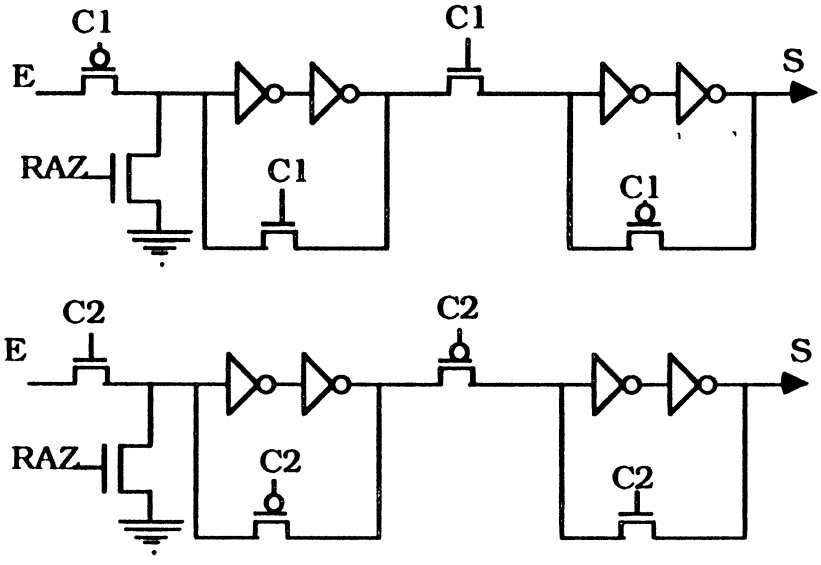


Figure IV.4 Bascules Maître/Esclave

Ce choix a été validé par des simulations électriques. Leurs résultats sont présentés en annexe.

III.2. Registre à décalage

Lors des spécifications architecturales, nous avons mis en évidence la nécessité de deux horloges :

- horloge lente LC,
- horloge rapide HC.

Chaque horloge est biphasée et on obtient le chronogramme suivant :

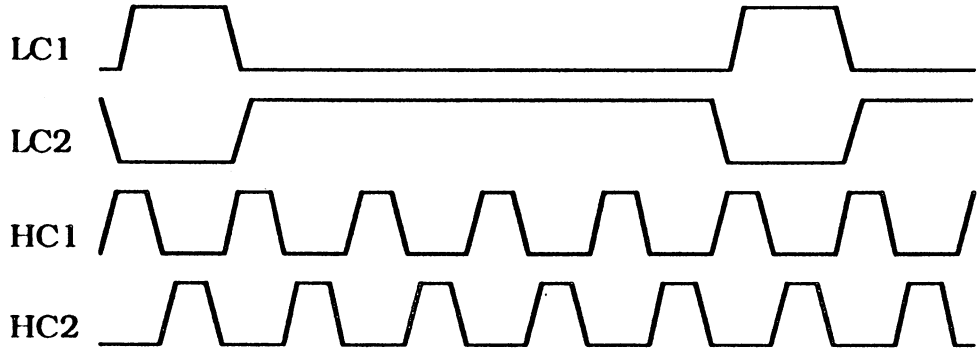


Figure IV.5 Horloges lentes (LC) et rapides (HC)

Mais le type de bascule défini dans le paragraphe précédent (figure IV.4), ne nécessite qu'une seule phase de chaque horloge. Nous avons décidé arbitrairement de ne générer que LC1 et HC2.

Chaque élément du registre à décalage est contrôlé par l'horloge rapide (HC2). Il devient alors :

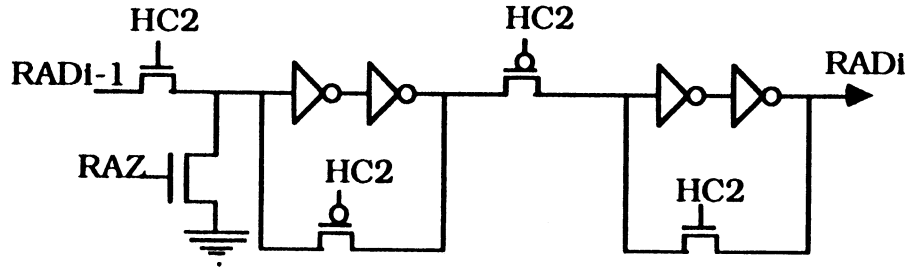


Figure IV.6 Elément du registre à décalage

Le registre à décalage doit revenir à son état initial (REI) au début de chaque calcul synaptique. La solution évidente consiste à avoir des bascules avec remise à 0 et d'autres avec remise à 1.

Mais cette solution nuit à la régularité du dessin et nous avons préféré n'utiliser qu'un seul type de bascule (RAZ) et procéder en deux étapes pour effectuer la REI :

- remise à zéro des 16 éléments du registre,
- chargement de l'état initial.

Le niveau haut de l'horloge lente LC1 est le signal du début de calcul. Le signal de remise à zéro du registre à décalage (RAZRAD) est généré selon le chronogramme présenté figure IV.7, par la logique qui suit :

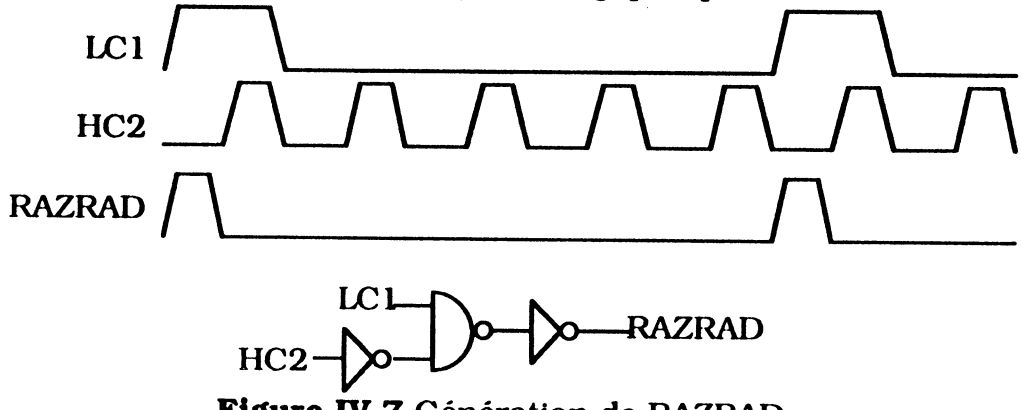


Figure IV.7 Génération de RAZRAD

Ce signal va remettre à zéro les 16 éléments du registre pendant la 1ère phase. Pour charger l'état initial, il suffit de mettre à 1 l'entrée du premier élément du registre à décalage (RAD0) pendant le début du calcul, puis de la mettre à 0. Afin d'éviter la génération d'un signal spécial, l'entrée de RAD0 est reliée au signal d'horloge LC1 car il est à 1 au début du calcul et à 0 après.
Les signaux issus du registre à décalage (RADi) contrôlent les accès des registres C et SP des cellules.

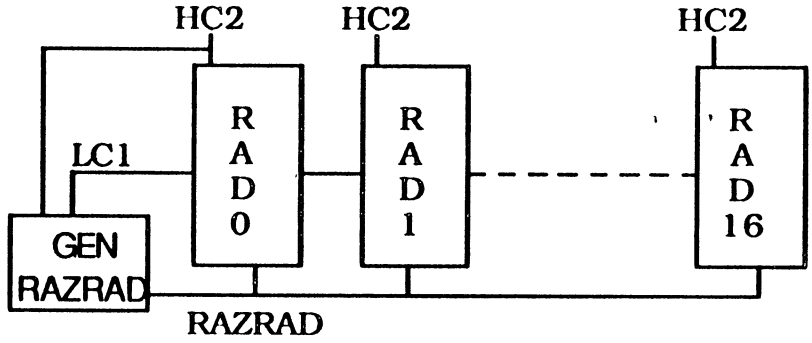


Figure IV.8 Séquenceur

IV. LA PARTIE OPERATIVE

Suivant les définitions architecturales présentées dans le chapitre précédent, la partie opérative est constituée des registres C et SP et de l'UAL.

IV.1. Le registre C

Ce registre simple contient le coefficient synaptique de la cellule sur 8 bits.
Le chargement de ce registre se fait bit à bit à partir du registre D. C'est la commande CHH2 qui contrôle ce chargement durant l'initialisation de la matrice.
Ce registre charge le "bus" de sortie Cout qui est l'un des opérandes de l'U.A.L.
Les signaux RADi, issus du séquenceur, sélectionnent l'élément du registre C nécessaire au calcul.

Le schéma de ce registre simple est donc le suivant :

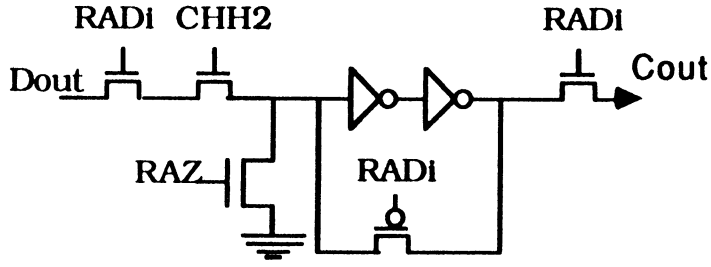


Figure IV.9 Elément du registre C

Ce registre 8 bits participe aux opérations de l'U.A.L. qui se font sur 16 bits. Il faut donc étendre le format 8 bits sur 16 bits. Pour cela, le bit de signe C7 est étendu comme le montre la figure IV.10.

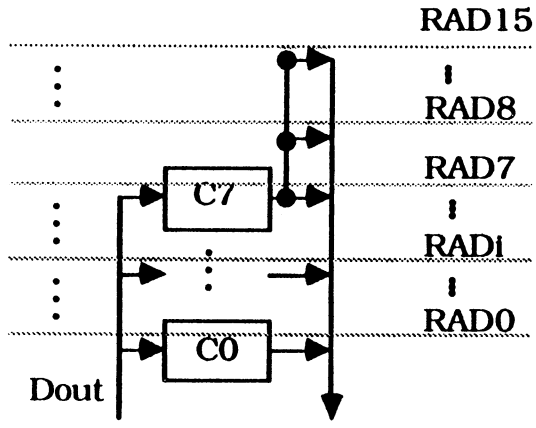


Figure IV.10 Registre C

IV.2. Le registre SP

Le registre SP est un registre simple 16 bits qui contient la somme partielle de la cellule.

Il reçoit en entrée la sortie de l'U.A.L. Son chargement se fait sur HC2. Les signaux RADi, issus du séquenceur, sélectionnent les éléments du registre SP qui sont écrits et lus. La lecture d'un bit de SP est anticipée : lorsque le $i^{ème}$

bit est écrit, le i+1ème bit est lu. Le bit lu est mémorisé dans le registre simple SPout sur HC2, puis transmis vers la cellule voisine.

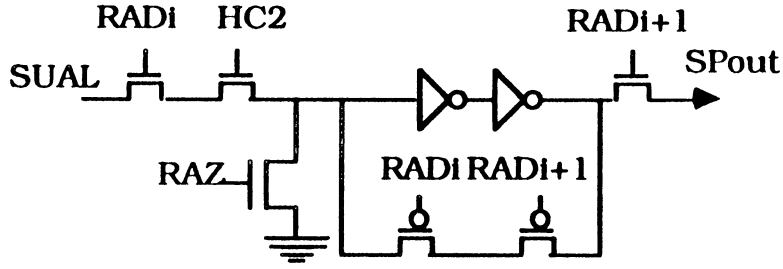


Figure IV.11 Elément du registre SP

IV.3. L'U.A.L.

L'U.A.L. est un additionneur/soustracteur 1 bit en complément à deux.
L'élément de base de l'additionneur est le "ou exclusif". Nous avons choisi le schéma minimal qui n'utilise que 6 transistors :

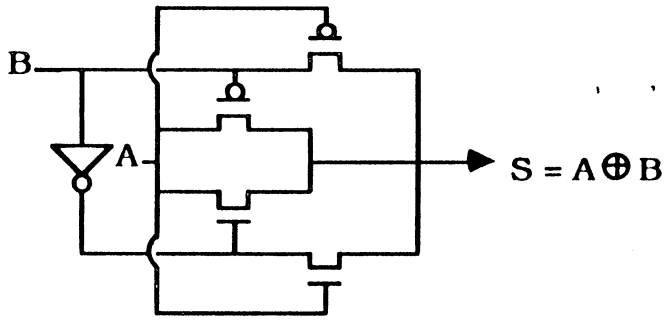


Figure IV.12 Ou exclusif

Suivant le schéma classique, l'additionneur est constitué de deux demi-additionneurs :

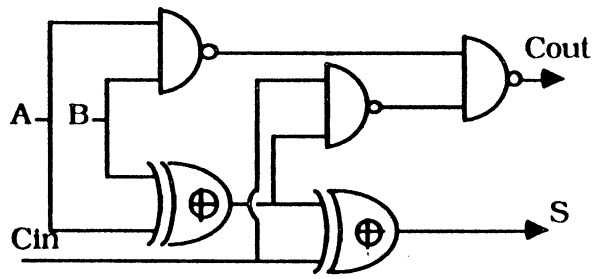


Figure IV.13 Additionneur

Ce schéma a un inconvénient majeur : les signaux d'entrée risquent d'être dégradés. Nous avons donc intercalé des inverseurs qui régénèrent les signaux. Ce schéma a évidemment été validé par les simulations électriques.

La retenue engendrée est mémorisée dans un registre Maître/Esclave (CARRY).

L'additionneur/soustracteur est réalisé en complément à deux. Les opérandes de l'additionneur sont :

- le registre simple SPin,
- le "bus" Cout complémenté ou non suivant l'opération (D=0 : addition, D=1 : soustraction). Ce choix peut se faire par une porte "ou exclusif" comme le montre la figure suivante.

Figure IV.14 Complément à deux

L'U.A.L. évalue son résultat pendant la première phase de l'horloge (HC2 au niveau bas). Il est mémorisé dans le registre somme partielle pendant la deuxième phase (HC2 au niveau haut).

V. LA PARTIE COMMUNICATION

La partie communication est principalement constituée des registres NV, M et D. Ces registres Maître/Esclave mémorisent les valeurs communiquées par les cellules voisines sur le niveau haut de LC afin que les valeurs soient disponibles dès le début d'une nouvelle phase de calcul.

Le schéma de ces registres est identique à celui du registre à décalage.

La partie communication doit aussi assurer la détection de convergence. Le dispositif, nommé EQU dans la définition architecturale, permet de comparer les entrées Sud et Est. Afin de minimiser le nombre de transistors à dessiner, nous avons décidé de détecter l'inégalité plutôt que l'égalité : la fonction "ou exclusif" déjà utilisée dans l'additionneur a été réutilisée.

VI. LE CIRCUIT COMPLET

Dans le plan de masse général présenté par la figure IV.1, la mise en vis-à-vis des trois registres RAD, SP et C crée un espace libre au-dessus de C, puisqu'il n'est constitué que de 8 bits et non de 16. Cet espace est utilisé pour intégrer l'additionneur/soustracteur à proximité directe de ses opérandes (SPin et C) et de son résultat (SP). Les registres CARRY et D sont intégrés au plus proche de l'additionneur car ils lui sont nécessaires. La partie communication, qui est relativement indépendante du reste de la cellule, est intégrée dans l'espace restant. On obtient alors le plan de masse présenté par la figure IV.15.

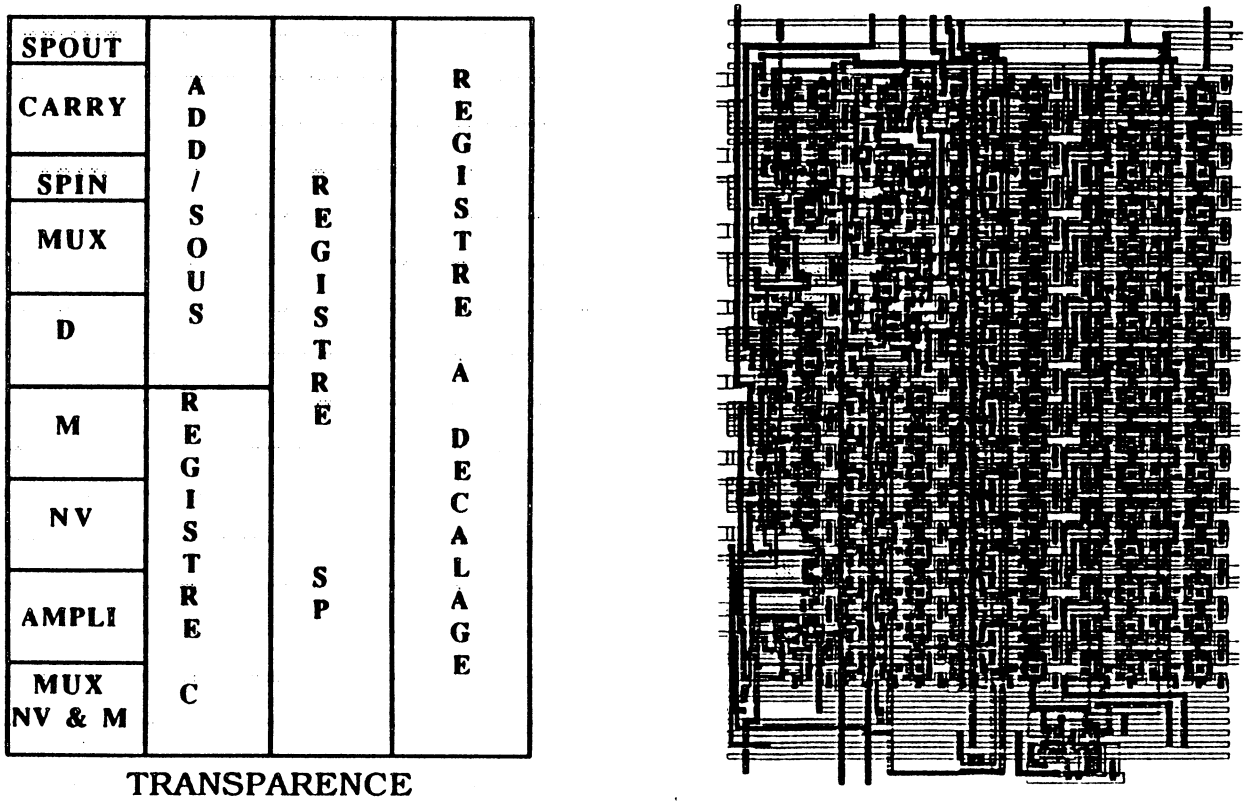


Figure IV.15 plan de masse et dessin d'une cellule

Le circuit complet a été intégré selon le plan de masse précédemment défini. Les entrées et les sorties de la cellule sont reliées au boîtier par des plots d'Entrée/Sortie.

Le dessin complet de la cellule avec son séquenceur mais sans les plots occupe une surface de 350µm x 510µm = 0.18mm^2. Il est composé de 600 transistors. La densité est donc de 3333 transistors au mm^2. Ce dessin est donc "dense" pour cette technologie 2µm 2 niveaux de métal.

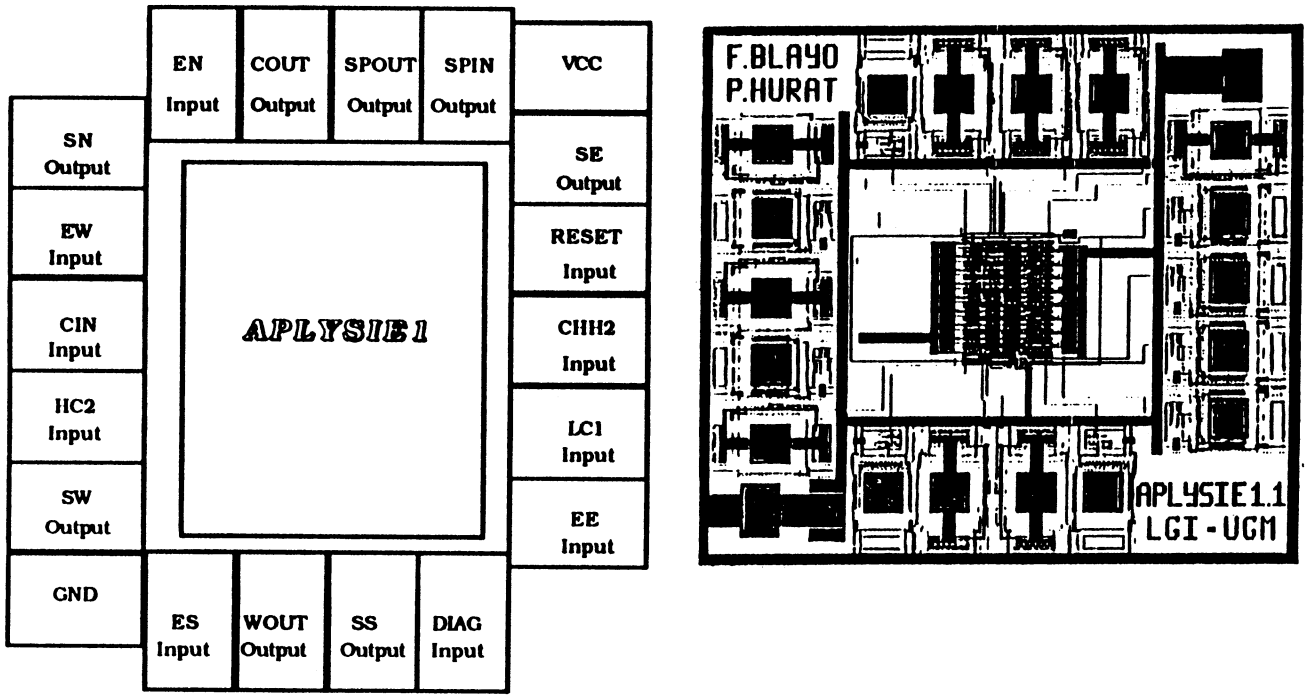


Figure IV.16 Circuit complet et dessin

Le dessin de la figure IV.16 donne une impression de faible densité (surface 4,8mm^2). Ceci provient du grand nombre de plots.

En effet, ce circuit a pour but de valider les choix architecturaux et confirmer les résultats des simulations. Aussi, différents points internes du circuit ont été reliés aux plots d'Entrée/Sortie afin de faciliter le test.

VII. TEST

Le circuit a été fabriqué par ES2 grâce au service du CMP français [DEL87]. Il a été ensuite monté en boîtier 40 pattes.
Ceci nous a permis de procéder au test. Dans un premier temps, nous avons utilisé le matériel du Centre Inter-universitaire de Micro-Electronique (CIME) qui nous a permis d'effectuer un test fonctionnel.
Cette procédure se décomposa en deux parties :

- test de la partie communication,
- test de la partie calcul.

Une fois ce test effectué avec succès, il fallait vérifier les performances du circuit. Ce test électrique nécessitait des équipements plus performants que le LETI (Laboratoire d'Electronique et Technologie d'Instrumentation, laboratoire du CEA) mit à notre disposition. Les résultats obtenus ont confirmé ceux indiqués par les simulations. Le circuit fonctionne avec une horloge de base HC2 de 25 MHz.

Ce premier circuit nous a permis de valider l'architecture et son implantation sur le silicium. La conception du deuxième circuit qui consiste en un assemblage de ces cellules de base, a pu alors être effectué avec le maximum de fiabilité.

V. Intégration d'un sous-réseau : Projet APLYSIE16

Il s'agit d'intégrer un ensemble de cellules de base (Aplysie 1) sur un circuit. Pour conserver la modularité de l'architecture, un sous-réseau carré est intégré. Ce sous-réseau peut, à son tour, être l'élément de base d'un plus grand réseau, constitué d'un ensemble de circuits interconnectés.
Dans un premier temps, nous étudierons l'intégration d'un circuit qui comprend un sous-réseau de 16x16 cellules. Puis, nous détaillerons comment, en interconnectant de tels circuits, on peut former de plus grands réseaux de neurones.

I . PROBLEMES LIES A L'INTEGRATION

Les principaux problèmes rencontrés lors de l'intégration d'une architecture systolique sont de deux ordres :

- problèmes liés aux Entrées/Sorties : le nombre d'E/S croît comme la racine carré du nombre de cellules (Nb) d'un réseau carré. Les limites technologiques des boîtiers de circuits intégrés sont rapidement atteintes.
- problèmes liés à la distribution des signaux de contrôle : il s'agit de commander un ensemble de cellules travaillant en parallèle. Comme ces cellules communiquent par voisinage, il est absolument impératif d'éviter tout décalage important de signaux entre deux cellules voisines.

Ces deux préoccupations ont motivé nombre des choix effectués.

II. ARCHITECTURE DU CIRCUIT APLYSIE 16

II.1. Fonctionnalité du circuit

Le circuit à intégrer est le composant de base du réseau final. Ce réseau doit pouvoir détecter la convergence et comprendre un dispositif d'Entrée/Sortie

comme nous l'avons décrit dans les chapitres précédents. Afin de conserver la modularité de l'architecture, chaque circuit intègre ces dispositifs. Il a été jugé préférable de perdre un peu de place sur chaque circuit et de ne fabriquer qu'un seul type de circuit. La surface de chaque cellule est augmentée de 2.5%. De plus, au vu du plan de masse d'Aplysie 1, cette place pourrait difficilement être récupérée.

Un réseau final à base de circuits Aplysie16 comporte donc des dispositifs inutiles (cf figure V.1) qui ne doivent cependant pas perturber son bon fonctionnement.

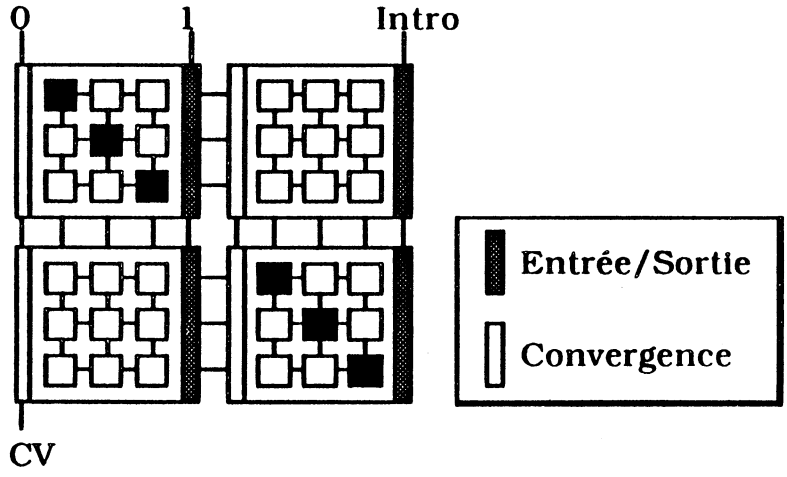


Figure V.1 Interconnexion des circuits

Le dispositif d'Entrée/Sortie, aussi appelé "fermeture Eclair®", permet soit le rebouclage et l'application de la fonction de seuil (si Intro vaut 0) soit la transmission transparente horizontale (si Intro vaut 1). Les dispositifs intermédiaires ne doivent en aucun cas effectuer le rebouclage. Il faut donc forcer à 1 le signal Intro des "fermetures Eclair" intermédiaires pour ne pas perturber la circulation. Le dispositif d'Entrée/Sortie des circuits Est doit être programmé selon la gestion d'Entrée/Sortie choisie : locale, globale ou probabiliste.

Le dispositif de convergence des circuits intermédiaires évalue des valeurs non significatives. Seuls les dispositifs des circuits Ouest calculent effectivement la convergence.

II.2. Intégration des dispositifs latéraux

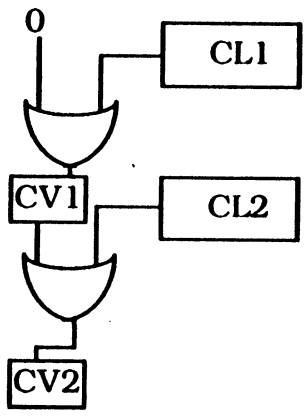
Les dispositifs latéraux sont la détection de la convergence et la "fermeture Eclair".

Le dispositif de convergence doit effectuer le ET des convergences locales qui arrivent sur le bord Ouest. La convergence locale est détectée dans les cellules diagonales par la table de vérité suivante :

eE	eS	CL
0	0	0
0	1	1
1	0	1
1	1	0

Figure V.2 Fonction de convergence locale

La valeur qui arrive sur le bord Ouest est donc $\overline{CL}$. Le dispositif de convergence calcule donc $\overline{CV} := \overline{NOR(CL_i)}$



CL1	CL2	CV1	CV2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	1

Figure V.3 Dispositif de convergence

La "fermeture Eclair" doit être transparente, si Intro vaut 1, ou effectuer le rebouclage, si Intro vaut 0.

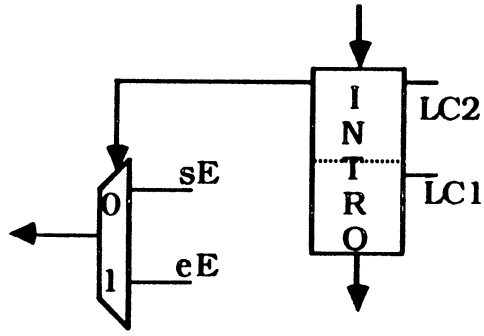


Figure V.4 "fermeture Eclair"

II.3. Intégration du séquenceur

Comme nous l'avons déjà dit dans les spécifications architecturales, plusieurs cellules peuvent partager le même séquenceur. A la limite, un seul séquenceur suffit par circuit. Cependant, pour limiter les problèmes de propagation des 16 signaux issus du séquenceur, des blocs de 8 cellules ont été constitués. Chaque bloc possède son propre séquenceur situé en son centre, comme le montre la figure V.5.

Le nombre de cellules par bloc a été choisi en fonction de la sortance du registre à décalage. Ce choix a bien sûr été validé par des simulations électriques.

Comme dans Aplysie 1, les signaux RADi traversent en niveau métal2 les cellules. Les commandes nécessaires au contrôle des cellules (CTRL) sont intégrées dans l'espace inter-bloc comme le montre la figure V.5.

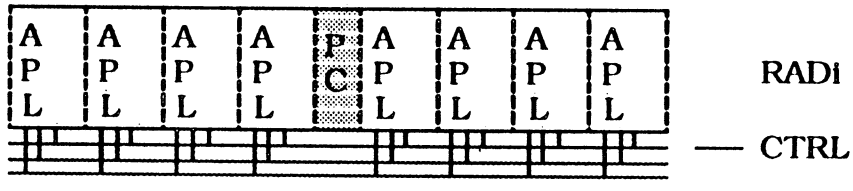


Figure V.5 Structure d'un bloc

II.4. Taille du sous-réseau

La taille du sous-réseau à intégrer dépend essentiellement de deux facteurs technologiques :

- la taille maximale de la puce : le contrat de fabrication entre le service du CMP et ES2 spécifie que les puces ne doivent pas dépasser la dimension de 7x7=49 mm^2,
- le nombre maximal de pattes du boîtier.

D'un point de vue fonctionnel, le réseau à construire étant carré (128x128), la taille d'un sous-réseau doit être un diviseur de 128x128. La cellule de base étant un rectangle de 510µm de hauteur, un circuit fonctionnellement carré (NxN cellules) est physiquement rectangulaire. Comme le circuit devait tenir dans un carré de 7 mm de côté, nous avons choisi, dans un premier temps, d'intégrer un réseau rectangulaire 8x16 (8x0,51<7 et 16x0,51>7).

Cependant, la structure rectangulaire de ce sous-réseau était pénalisante. Elle nécessitait deux signaux de programmation de la diagonale. Mais surtout, ce réseau n'utilisait pas au mieux la place qui nous était impartie.

Le service du CMP nous a obtenu une dérogation d'ES2 pour fabriquer un circuit de 49mm^2 physiquement rectangulaire qui nous permet d'intégrer un sous-réseau fonctionnellement carré de 16x16 cellules.

II.5. Problème des plots

II.5.1. Solution triviale

Les Entrées/Sorties fonctionnelles nécessaires pour un réseau 16x16 se décomposent en 4 catégories :

- entrées des signaux CTRL et d'alimentation : 7,
- entrées des cellules situées sur les côtés du circuit : 4x16 = 64,

- sorties des cellules situées sur les côtés du circuit : 4x16 = 64,
- entrées/sorties de la "fermeture Eclair®" et du dispositif de convergence : 4.

On obtiendrait donc un total de 139 plots.

Ceci pose deux problèmes :

- les boîtiers traditionnels ne permettent pas un aussi grand nombre de pattes. Il faut alors utiliser des boîtiers dit "*Chip carrier*",
- la place perdue par les plots est importante.

II.5.2. Multiplexage des plots

Afin de réduire le nombre de plots nécessaires, envisageons le multiplexage des Entrées/Sorties.

On peut remarquer que les communications ne se font pas toutes sur la même horloge. Seule la transmission horizontale d'Ouest en Est des sommes partielles s'effectue sur l'horloge rapide HC2. Toutes les autres communications se font sur l'horloge lente LC1. La différence de fréquence permet de mettre en place le multiplexage des Entrées/Sorties correspondant aux communications verticales et horizontales d'Est en Ouest.

Ces communications, se font sur l'horloge lente, c'est à dire que la communication s'effectue sur le niveau bas de LC1 et les valeurs doivent être à l'entrée de la cellule à la fin du niveau bas de LC1 pour que la valeur puisse être chargée sur le front montant de LC1.

Le multiplexage des plots de Sortie est simple. Il suffit d'envoyer alternativement, pendant le niveau bas de LC1, les valeurs à communiquer sur le plot comme le montre la figure V.6b.

De même, les plots d'Entrée reçoivent alternativement les valeurs comme le montre la figure V.6a. Pour que les cellules puissent charger leur valeur sur le

front montant de LC1, ces valeurs sont mémorisées temporairement dans des buffers.

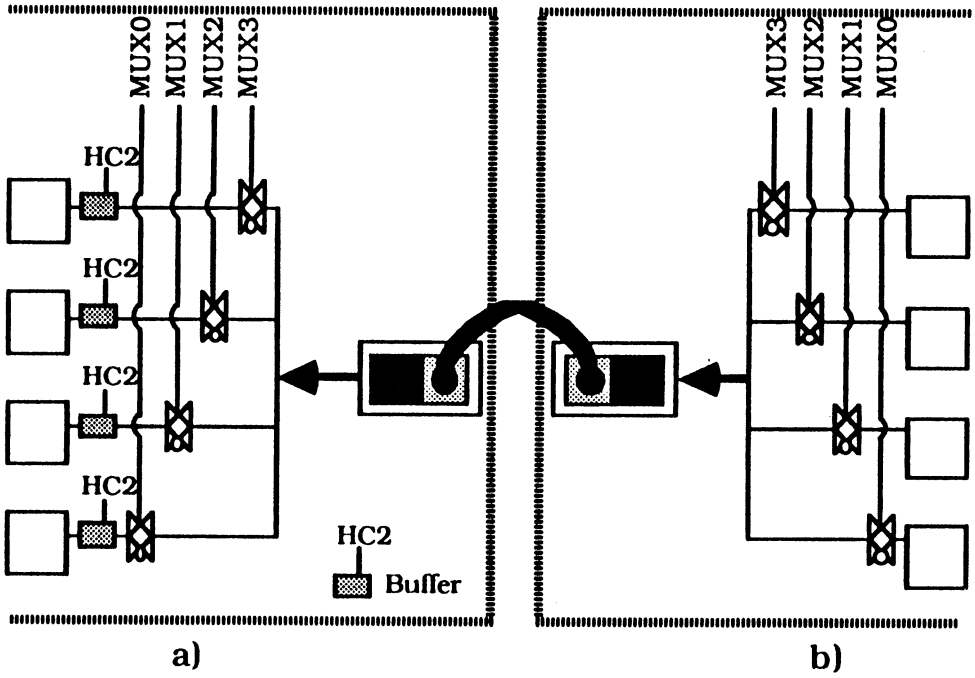


Figure V.6 Multiplexage des communications

Les signaux nécessaires au multiplexage (MUX0,...,MUX3,...,MUX7) doivent suivre le chronogramme suivant :

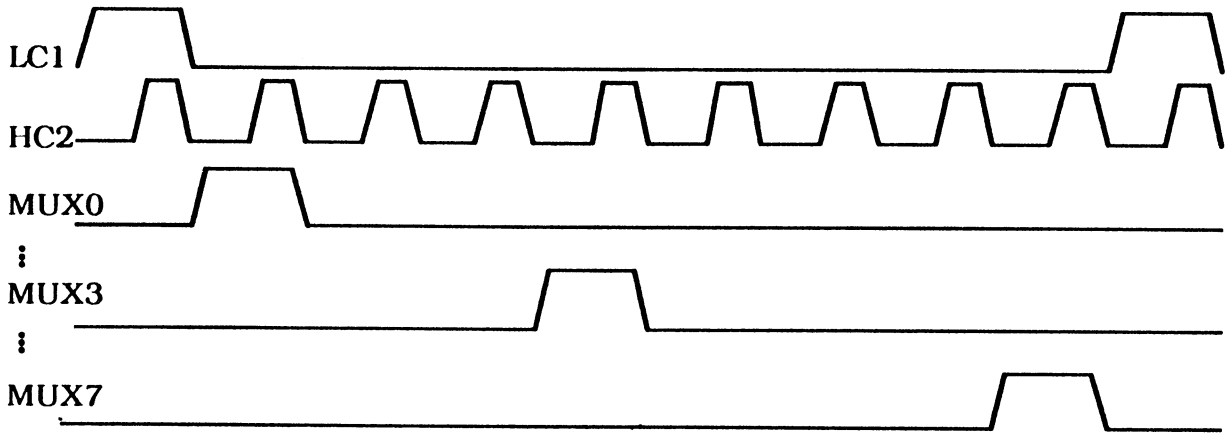


Figure V.7 Chronogramme des commandes de multiplexage

On peut remarquer que les commandes MUX_i sont identiques aux commandes issues du séquenceur RADi. Le multiplexage des communications horizontales est directement contrôlé par les signaux du séquenceur.

Pour le multiplexage vertical, les commandes du séquenceur sont physiquement inaccessibles et des registres à décalage supplémentaires ont été ajoutés (2 au Nord et 2 au Sud du chip). Ces RAD ne comprennent que 4 éléments car les communications verticales sont regroupés 4 par 4. Ce détail du circuit peut être vu en annexe.

Remarque : dans les circuits du bord Est du réseau, le multiplexage des entrées Est est possible car la fonction de seuil est appliquée à l'intérieur du circuit.
Supposons que la fonction de seuil soit externe. Les cellules du bord Est délivrent les 16 bits des sommes partielles pendant toute la durée du niveau bas de LC1. Pour connaître le signe de cette somme partielle, il faut attendre le bit le plus significatif et c'est ce dernier bit qu'il faut réinjecter par les plots d'Entrée Est. Cette réinjection a lieu simultanément pour toutes les cellules du bord Est à la fin du niveau bas de LC1, et les valeurs doivent être réinjectées avant le niveau haut de LC1 : on n'a pas le temps de multiplexer les entrées.
Avec la fonction de seuil intégrée au circuit, les plots d'Entrée Est ne servent pas au rebouclage. Le multiplexage peut avoir lieu.

II.5.3. Solution adoptée

Les communications verticales sont regroupées 4 par 4 et les communications horizontales d'Est en Ouest 8 par 8. Le décompte des plots donne alors :

- plots d'entrée et de sortie d'Ouest en Est : 16 x 2 = 32,
- plots d'entrée et de sortie d'Est en Ouest : 2 x 2 = 4,
- plots d'entrée et de sortie du Nord au Sud : 4 x 2 = 8,

- plots d'entrée et de sortie du Sud au Nord : 4 x 2 = 8,
- plots d'entrées des signaux CTRL et alimentations : 7,
- plots de la "fermeture Eclair®" : 2,
- plots du dispositif de convergence : 2.

On obtient donc un total de 63 plots.

Le multiplexage permet de réduire le nombre de pattes du circuit afin que celui-ci puisse tenir dans un boîtier standard 64 broches.

La "couronne" de plots est intégrée sous forme de "U" autour des cellules afin de ne pas dépasser la taille critique de 49 mm^2. Le circuit final obtenu occupe une surface de 37mm^2 sans la couronne de plots et de 48,6 mm^2 avec les plots.

II.6. Distribution des commandes et de l'alimentation

Le projet Aplysie 1, nous a permis de valider la cellule de base du circuit Aplysie16. La principale difficulté d'intégration réside alors dans la distribution des commandes et l'alimentation du circuit.

II.6.1. Principe

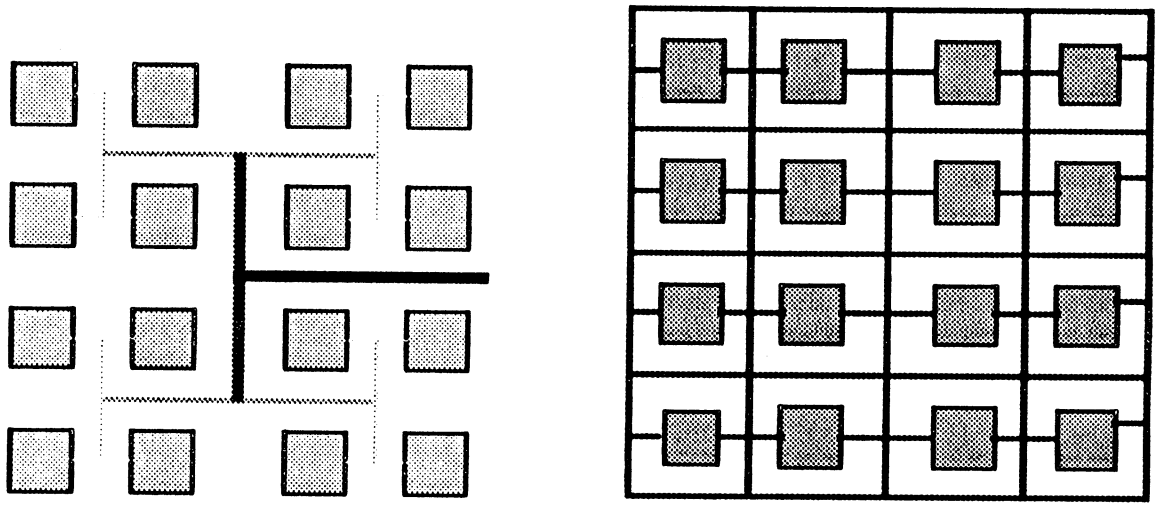
Les signaux CTRL sont au nombre de 4 :

- le signal de remise-à-zéro : RESET,
- l'horloge lente : LC1,
- l'horloge rapide : HC2,
- le signal de chargement de C : CHH2.

Le signal DIAG est un signal de programmation. Cependant, il est statique et sa distribution ne pose pas de problème majeur.

Les lignes d'alimentation et de masse sont intégrées suivant une structure de grille qui permet une bonne distribution.

L'implantation sur le silicium des autres signaux est assez critique. Il faut tenir compte des décalages induits par les temps de propagation. La technique la plus simple est d'intégrer ces signaux suivant une structure arborescente en H [FRI85] comme le montre la figure V.8a. Avec cette structure, les branches de l'arbre de distribution d'horloge sont égales. De plus l'architecture étant cellulaire, les feuilles de l'arbre sont identiques. Les signaux arrivant à ces feuilles sont donc parfaitement en phase.



a) b)

Figure V.8 Distribution des signaux de commandes et de l'alimentation

II.6.2. Réalisation

La structure en arbre présentée pour la distribution des signaux évite leur déphasage mais elle augmente la capacité des lignes de distribution. Aussi, nous avons adopté une structure intermédiaire. Une feuille de l'arbre n'est plus une cellule mais un bloc de 8 cellules comme le montre la figure V.9a. Les 4 branches maîtresses de l'arbre sont dotées d'amplificateurs adaptés à la capacité des lignes.

De même les alimentations ont été intégrées sur une structure en grille autour des blocs.

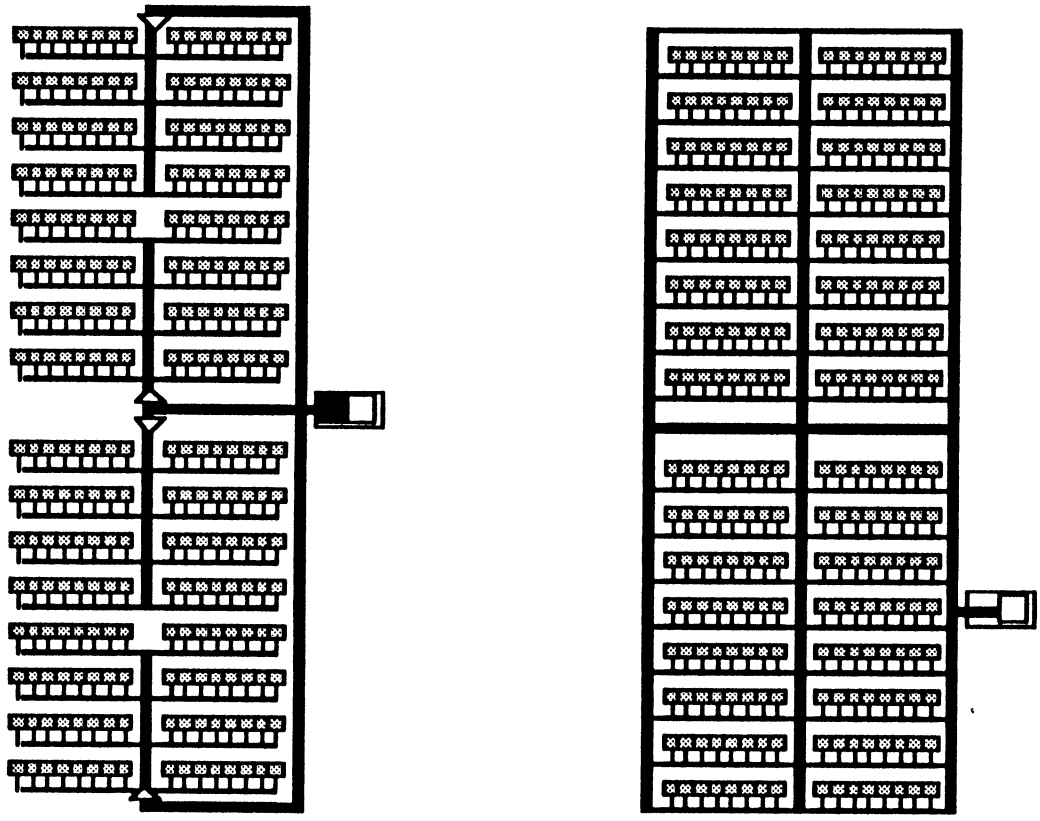


Figure V.9 Distribution des commandes et de l'alimentation sur le circuit

Dans la distribution des contrôles, chaque branche maîtresse attaque 4 blocs. Les extractions faites à partir du dessin des masques ont permis de calculer les capacités des lignes de distribution pour les 4 signaux de CTRL.

Les valeurs obtenues sont :

- pour RESET : 16 Pf
- pour HC2 : 8 Pf,
- pour LC1 : 7 Pf,
- pour CHH2 : 4 Pf.

Le calcul des amplificateurs adéquats doit être fait avec soin. Nous avons suivi une méthode systématique qui permet de définir le nombre d'étages

d'amplificateurs et leurs dimensions en fonction de la capacité à charger, des temps de montée et de descente voulus et des caractéristiques du bloc d'où est issu le signal.

Dans cette méthode, on suppose que le temps de montée Tm et le temps de descente Td sont égaux. On admet aussi que le rapport des dimensions des transistors P et des transistors N vaut à peu près 2.

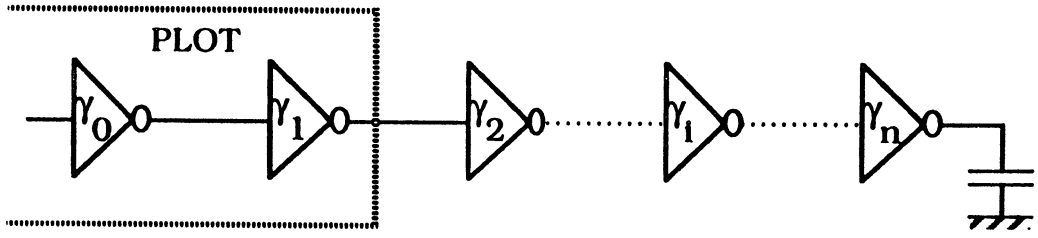


Figure V.10 Chaîne d'amplification

Dans notre application, les étages 0 et 1 sont constitués par le plot d'entrée. Soient VDD la tension d'alimentation, n le nombre d'étages de la chaîne d'amplification, $\gamma_i = W_i/L_i$ le dimensionnement du $i^{ème}$ étage, C la capacité à charger par le $N^{ème}$ étage, $K = \mu C_{ox}/2$ une constante technologique, on a :

$$n = Ln\,(\gamma_n / \gamma_0)$$

$$\gamma_n = 4C/(K.VDD.Tm) \qquad \textbf{(V.1)}$$

$$\gamma_i = \gamma_0\,(\gamma_n / \gamma_0)^{1/n}$$

Application Numérique :

VDD= 5 V; K = 3.10^{-5}; Tm = 5.10^{-9} s; γ_0 = 10;

Pour RESET : C= 16.10^{-12} F

$\gamma_n = (4\,.\,16.10^{-12}) / (3.10^{-5}\,.\,5\,.\,5.10^{-9}) \approx 90$

L = 2µm ⇒ W_N = 180µm ⇒ W_P = 360µm.

n = Ln (360/10) ≈ 3

$\gamma_2 = 10\,(\,90 / 10)^{2/3} \approx 40$

L = 2µm ⇒ W_N = 80µm ⇒ W_P = 160µm.

Pour HC2 : $C = 8.10^{-12}$ F

$\gamma_n = (4 . 8.10^{-12}) / (3.10^{-5} . 5 . 5.10^{-9}) \approx 45$

$L = 2\mu m \Rightarrow W_N = 90\mu m \Rightarrow W_P = 180\mu m.$

$n = Ln (180/10) \approx 3$

$\gamma_2 = 10 (45 / 10)^{2/3} \approx 30$

$L = 2\mu m \Rightarrow W_N = 60\mu m \Rightarrow W_P = 120\mu m.$

Pour LC1 : $C = 7.10^{-12}$ F

$\gamma_n = (4 . 7.10^{-12}) / (3.10^{-5} . 5 . 5.10^{-9}) \approx 40$

$L = 2\mu m \Rightarrow W_N = 80\mu m \Rightarrow W_P = 160\mu m.$

$n = Ln (160/10) \approx 3$

$\gamma_2 = 10 (40 / 10)^{2/3} \approx 30$

$L = 2\mu m \Rightarrow W_N = 60\mu m \Rightarrow W_P = 120\mu m.$

Pour CHH2 : $C = 4.10^{-12}$ F

$\gamma_n = (4 . 8.10^{-12}) / (3.10^{-5} . 5 . 5.10^{-9}) \approx 20$

$L = 2\mu m \Rightarrow W_N = 40\mu m \Rightarrow W_P = 80\mu m.$

$n = Ln (80/10) \approx 2$

$\gamma_2 = 10 (20 / 10)^{2/3} \approx 15$

$L = 2\mu m \Rightarrow W_N = 30\mu m \Rightarrow W_P = 60\mu m.$

Ces dimensions de transistors sont des valeurs minimales. Le dimensionnement des transistors implantés est légèrement différent mais toujours supérieur. De plus, ces chaînes d'amplification ont fait l'objet de simulations électriques dont les résultats sont présentés en annexe.

Les signaux distribués sont retardés par rapport au signal d'entrée mais leur décalage respectif est négligeable.

III. REALISATION D'UN RESEAU COMPLET

Le circuit Aplysie16 a été conçu pour construire un réseau de 128x128 cellules. Un ensemble de 8x8 circuits Aplysie16 permet de réaliser ce réseau.

III.1. Réalisation du réseau 128x128

Afin de réaliser un réseau fonctionnellement correct, il faut programmer les différents circuits en fonction de leur localisation dans le réseau. La figure suivante distingue 10 localisations numérotées de 0 à 9 sur la figure V.11 pour lesquelles les circuits doivent être différenciés.

NORD

OUEST								EST
	6	3	3	3	3	3	3	7
	2	1	0	0	0	0	0	4
	2	0	1	0	0	0	0	4
	2	0	0	1	0	0	0	4
	2	0	0	0	1	0	0	4
	2	0	0	0	0	1	0	4
	2	0	0	0	0	0	1	4
	9	5	5	5	5	5	5	8

SUD

Figure V.11 Localisations des circuits dans le réseau final

Tous les circuits sont identiques, mais une programmation adéquate permet de spécifier le comportement des circuits en fonction de leur localisation (cf annexe).

III.2. Evaluation des performances

On suppose que le circuit peut fonctionner avec une horloge de base de 20MHz. Nous allons évaluer les performances théoriques du réseau en régime stationnaire.

Si les coefficients synaptiques sont sur 4 bits, un circuit Aplysie16 seul a les caractéristiques théoriques suivantes :

- période de l'horloge de base HC2 = $1/20.10^6 = 5.10^{-8}$ s,
- l'évaluation d'une somme partielle de 8 bits nécessite 9 périodes de l'horloge de base, qui définissent alors la période de l'horloge lente LC1,
- temps d'évaluation d'un vecteur de 16 composantes = $4,5.10^{-7}$ s,
- nombre de vecteurs par seconde = $2,2.10^6$ vecteurs/s,
- performances = 35.10^6 mises-à-jour de neurones par seconde,
  = $0,6.10^9$ opérations synaptiques par seconde soit 0,6 GSOPS,
- le temps de chargement de la matrice 16x16 = 64µs

Avec une horloge à 20 MHz, le circuit travaille à 0,6 GSOPS pour mettre à jour 35 millions de neurones par seconde. A cette fréquence, plus de 2 millions de produits matrice-vecteur sont effectués chaque seconde.

Pour un réseau de 128x128 cellules soit 8x8 circuits Aplysie16, 8 bits sont nécessaires pour le registre C et 16 bits pour SP. On obtient alors:

- période de l'horloge de base HC2 = $1/20.10^6 = 5.10^{-8}$ s,
- l'évaluation d'une somme partielle de 16 composantes nécessite 17 périodes de l'horloge de base, soit une période de l'horloge lente LC1,
- temps d'évaluation d'un vecteur de 128 composantes = $8,5.10^{-7}$ s,
- nombre de vecteurs par seconde = $1,18.10^6$ vecteurs/s,
- performances = 150.10^6 mises-à-jour de neurones par seconde,
  = $19,2.10^9$ opérations synaptiques par seconde (GSOPS).
- temps de chargement de la matrice 128x128 = 1ms.

Les calculs complets des performances sont exposés en détail en annexe.

Avec une horloge à 20 MHz, le réseau travaille à 20 GSOPS pour mettre à jour 150 millions de neurones par seconde. A cette fréquence, plus de 1 million de produits matrice-vecteur sont effectués chaque seconde.

III . 3 . Comparaison des performances

Afin d'effectuer une comparaison avec des processeurs classiques, nous avons procédé à des simulations sur différentes machines. Ces simulations portent sur le produit matrice-vecteur et la détection de convergence. Deux dimensions de matrice ont été utilisées : matrice 16x16 et 128x128. Les résultats de ces simulations sont présentées dans la table de la figure V.12.

	68000	68030	T800	Aplysie16	Aplysie 128
16x16	78 vec/s	416 vec/s	1302 vec/s	$2,2.10^6$ vec/s	----
128x128	1,4 vec/s	7 vec/s	19 vec/s	11.10^3 vec/s	$1,18.10^6$ vec/s

Figure V.12 Comparaison de performances

Les simulations sur 68000 et 68030 ont été effectuées à partir d'un programme Pascal compilé puis exécuté sur un micro-ordinateur Macintosh Plus et Macintosh II X d'Apple. De même, les simulations effectuées sur Transputer ont été écrites en langage OCCAM puis exécutées sur un seul Transputer. Vu le grain de parallélisme du problème qui est très fin, l'utilisation d'un Transputer par neurone n'augmenterait pas de façon significative les performances : perte de temps pour effectuer les communications. Le meilleur compromis serait de regrouper les calculs relatifs à plusieurs neurones sur un même Transputer. Les évaluations sur les circuits Aplysies sont effectuées à partir des résultats de la première phase de test.

Les simulations permettent d'évaluer les performances de l'architecture d'Aplysie. Même si la programmation des simulations en langage évolué (Pascal et Occam) réduisent les performances des 68000 et du Transputer, cette comparaison permet de prendre toute la mesure du gain de performance qu'apporte la conception de circuits dédiés.

VI. Perspectives

Nous présentons dans ce chapitre différentes extensions possibles du circuit, aussi bien au niveau de la fonctionnalité que de l'échelle d'intégration.
Le circuit précédemment décrit permet d'effectuer la phase de reconnaissance sur les réseaux monocouche. Il est évidemment possible d'envisager l'implantation de la phase d'apprentissage sur l'architecture définie. D'autre part, les spécifications ont conduit à la définition d'une architecture pour un réseau de 128 neurones. Mais, cette limitation à 128 neurones n'est pas incontournable. Des réseaux de plus grande taille peuvent être construits à partir de l'architecture définie. Cela nous conduit aussi à étudier son intégration sur tranche entière.

I. IMPLANTATION DE L'APPRENTISSAGE

Comme nous l'avons présenté au chapitre "Modèles et applications", les algorithmes d'apprentissage diffèrent selon l'application envisagée. Néanmoins ils s'appliquent principalement à deux classes :

- les mémoires associatives,
- la résolution de problèmes d'optimisation combinatoire.

Les lois d'apprentissage pour ce dernier cas sont spécifiques à chaque application et leur intégration n'est pas envisageable. Par contre, les trois algorithmes dédiés aux applications de mémoires associatives sont suffisamment généraux pour être étudiés plus en détail.

I.1. Loi de Hebb

Nous rappelons que la modification de la matrice synaptique s'effectue selon la formule (VI.1).

$$\Delta C = (X X^t) / N \quad \text{avec } C = [0] \text{ avant l'apprentissage et } C_{i,i} = 0 \text{ après} \quad \text{(VI.1)}$$

Cette mise à jour peut être facilement effectuée sur un réseau systolique carré comme le montre la figure VI.1.

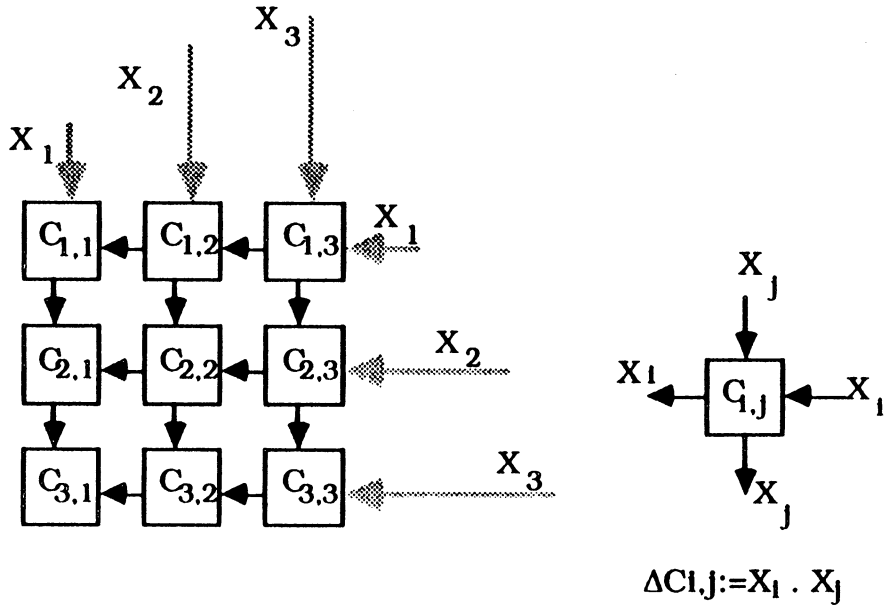


Figure VI.1 Apprentissage de Hebb

Comme les composantes des vecteurs prototypes X sont bivaluées (-1, +1), le produit X_iX_j est équivalent à un changement de signe. La seule difficulté provient de la division par N. Deux cas se présentent :

- N est une puissance de 2 : la division par N peut se faire par décalage,
- N n'est pas une puissance de 2 : cette division peut être omise si la précision des coefficients le permet. En effet, la division n'a aucune influence sur la phase de reconnaissance. Comme la formule (VI.2) l'énonce, la fonction de seuil F qui prend le signe de la somme pondérée, permet de supprimer le facteur 1/N.

$$F\left(\Sigma\ 1/N(\ C_{i,j}\ .\ X_j\ )\right) = F\left(1/N\ \Sigma\ (\ C_{i,j}\ .\ X_j\ )\right) = F\left(\Sigma\ (\ C_{i,j}\ .\ X_j\ )\right) \quad \textbf{(VI.2)}$$

L'opération effectuée est donc d'une complexité comparable à celle réalisée pendant la phase de reconnaissance.

I.2. Loi de la projection

La loi de la projection est plus complexe à réaliser.

Soit $y = (X^tX - X^tCX)$ alors $\Delta C = 1/y\,(CX - X)(CX - X)^t$ **(VI.3)**

La mesure "y" qui permet de quantifier le supplément d'information apporté par la mémorisation d'un prototype, est délicate à évaluer physiquement sur un réseau systolique car c'est une valeur globale. Elle ne peut pas être calculée localement par les cellules, et une fois calculée sa valeur doit être diffusée à toutes les cellules.

Aussi, une approximation de cette loi a été développée pour résoudre ce problème : la loi de la delta-projection.

I.3. Loi de la delta-projection

La répétition de la formule (VI.4) conduit à la création d'une matrice similaire à la matrice de la projection.

$\Delta C = 1/N\,(X - CX)X^t$ **(VI.4)**

Cette loi est intéressante sur des réseaux systoliques car son implantation ne nécessite que des valeurs locales.

Comme pour la loi de Hebb le facteur 1/N peut être omis. Cependant, les calculs sont plus complexes que ceux nécessaires à la phase de reconnaissance. L'architecture de la cellule devrait donc être modifiée pour effectuer cet apprentissage. Mais, une circulation systolique à réinjection, proche de celle que nous avons développée, serait bien adaptée à ce problème.

II. RESEAUX DE GRANDE TAILLE

La cellule a été spécifiée pour pouvoir faire partie d'un réseau de 128 neurones. La précision de C (8 bits) est suffisamment grande pour envisager des réseaux de taille supérieure à 128 neurones.

Les 16 bits du registre SP permettent déjà de réaliser un réseau de 256 neurones sans perte de précision. Ce réseau peut être réalisé par l'interconnexion de plusieurs cartes de 128 neurones. Cependant le nombre de cartes nécessaires croit avec le carré du nombre de neurones car il faut raisonner en terme de synapses. Sachant qu'un circuit Aplysie16 intègre 256 synapses, pour 128 neurones il faut 16384 synapses soit 64 circuits Aplysie16, pour 256 neurones on atteint 65536 synapses soit 256 circuits Aplysie16, ou 4 cartes Aplysie128 et pour 1 KNeurone (1024) on atteint le million de synapses soit 4096 circuits Aplysie16.

Au vu de cette progression, une autre stratégie peut être employée : la pagination. Le réseau peut être décomposé en plusieurs ensembles de synapses d'une taille correspondant aux circuits implantés (128x128 synapses par exemple). L'architecture précédemment présentée est alors considérée comme un accélérateur de calcul synaptique. Pour chaque ensemble de 128x128 synapses, la matrice synaptique qui lui correspond est chargée dans le circuit et le produit matrice-vecteur est réalisé pour un ensemble de prototypes.

Cette approche nécessite un environnement qui permet de mémoriser les coefficients, les composantes des vecteurs et les résultat intermédiaires.

Par exemple, le réseau de 1KN peut être simulé sur une carte de 128 neurones en 64 étapes. Le temps de chargement ajouté au temps nécessaire à la reconnaissance d'un vecteur de dimension 1024 sera de l'ordre du dixième de seconde (77 ms contre ≈ 3 ms pour un réseau 1KN directement implanté). Ce temps moyen peut être diminué si l'on a les moyens de mémoriser, à l'extérieur du circuit, les résultats intermédiaires de la reconnaissance de plusieurs vecteurs. Le traitement en mode pipeline peut alors se faire comme précédemment.

La pagination présente des performances modestes par rapport à une implantation directe (25 fois moins performante qu'une implantation directe pour 1KN). Elle nécessite un environnement plus sophistiqué. Cependant, elle est beaucoup plus souple qu'une implantation directe car la limitation en taille des réseaux traités est moins sévère.

III. INTEGRATION SUR TRANCHE

Lorsqu'on étudie l'intégration sur carte des circuits Aplysie16, le nombre de circuits croît avec le carré du nombre de neurones. Aussi, l'approche WSI est attrayante car elle réduit les problèmes dus au grand nombre de composants discrets.

Le circuit Aplysie16 intégré en VLSI a permis de valider l'architecture. Il peut servir de base à la conception d'un réseau de neurones en technologie WSI [BLA89]. Aussi, pour simplifier l'approche, nous considèrerons, dans un premier temps, la tranche comme une juxtaposition de circuits Aplysie16.

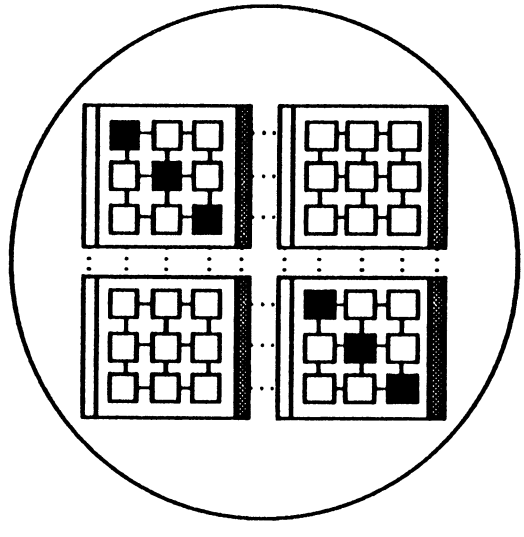


Figure VI.2 Intégration sur tranche entière

Cependant l'intégration sur tranche entière ne va pas sans problème. Dans le prochain chapitre nous étudierons plus en détail les difficultés liées au WSI. D'ores et déjà, nous pouvons effectuer une étude de faisabilité.

III.1. Problèmes liés aux WSI

L'intégration sur tranche se fait sur de telles surfaces que la présence de défauts physiques est inévitable. Cependant, nous devons différencier leurs manifestations.

Les défauts physiques sont la résultante de mauvaises manipulations ou de pollutions de la tranche pendant la phase de fabrication.

Les défauts qui n'ont pas été détectés ou pas réparés provoquent des erreurs logiques appelées pannes. Les pannes interviennent pendant la durée de vie du circuit.

La présence de défauts de fabrication est inhérente à l'intégration sur tranche. Nous supposons que les moyens mis en œuvre pour l'intégration sur tranche permettent d'éviter et de réparer les défauts de fabrication ; ceux-ci seront d'ailleurs étudiés plus en détail dans le prochain chapitre. Aussi, dans la fin de ce chapitre, nous parlerons des pannes et des mauvais fonctionnement.

III.2. Notion de tolérance aux pannes

Comme nous l'avons présenté dans le chapitre "Modèles et application", la mémorisation d'information dans une mémoire associative neuronique se fait par la définition des poids synaptiques. Ces poids synaptiques sont répartis dans les cellules synaptiques. La réponse du système est élaborée par l'ensemble de ces coefficients. On peut ainsi établir une analogie entre la mémorisation d'un prototype dans une mémoire associative neuronique et la mémorisation d'une image par un hologramme.

En effet, une image holographique est construite à partir de l'information mémorisée sur tout le film ; si un morceau de l'hologramme est altéré, l'image

globale est légèrement dégradée mais aucune partie de l'image n'est totalement perdue.

De même, le cerveau biologique perd quelques centaines de milliers de cellules nerveuses chaque jour. Pourtant, il sait toujours effectuer les mêmes tâches, bien que ses performances globales diminuent progressivement.

Nous retrouvons les mêmes caractéristiques dans les modèles de réseaux de neurones grâce à la représentation distribuée de l'information dans les cellules neuroniques ou synaptiques. Si quelques neurones présentent un mauvais fonctionnement, les performances du réseau se dégradent progressivement [FAH87]. De plus la fonction de mémoire associative implantée peut réduire les conséquences des dysfonctionnements locaux.

Cette tolérance vis-à-vis des pannes est un atout majeur pour l'intégration sur tranche entière.

III.3. Adaptation de l'architecture

Lors de l'intégration d'Aplysie16, nous avons spécifié l'architecture souhaitable pour un sous-réseau de 16x16 cellules. A chaque sous-réseau 16x16 sont ajoutés un dispositif de détection de convergence (DDC) et un dispositif d'Entrée/Sortie (DES) afin que le réseau final puisse effectuer la détection de convergence et les opérations d'Entrée/Sortie à moindre coût. (cf figure VI.2)

Cependant, la tolérance intrinsèque vis-à-vis des pannes du réseau de neurones ne doit pas être altérée par l'adjonction de ces dispositifs. Il faut envisager leur pannes et évaluer les conséquences sur le fonctionnement global du réseau.

Nous n'envisageons pas la présence de pannes fatales comme les courts-circuits. Nous limitons notre étude au cas de pannes logiques simples comme les collages.

III.4. DDC en panne

Si la gestion d'Entrée/Sortie nécessite la détection de la convergence (gestion locale ou globale mais non-probabiliste), les circuits situés sur le bord Ouest du réseau où est évaluée la convergence doivent fonctionner correctement.

III.5. DES en panne

Une panne intervenant dans le dispositif d'Entrée/Sortie peut affecter le bon fonctionnement de la "fermeture Eclair". Celle-ci permet le rebouclage et l'application de la fonction de seuil qui ne doivent être effectués que sur le bord Est du réseau.

Aussi, si nous nous situons dans le cas où chaque circuit intègre un DES, les DES des circuits intermédiaires doivent être transparents.

Un mauvais fonctionnement de la "fermeture Eclair" peut être désastreux. Si la panne se situe dans un circuit du bord Est, on risque de se trouver dans l'impossibilité d'effectuer une Entrée/Sortie. Dans l'autre cas, l'application de la fonction de seuil et du rebouclage au milieu du réseau, le rend inutilisable.

Cependant, l'intégration du rebouclage à l'intérieur du circuit n'est utile que dans l'optique d'une intégration VLSI. Dans ce cas, si le rebouclage était effectué à l'extérieur, les communications d'Est en Ouest ne pourraient plus être multiplexées.

Mais pour une intégration WSI, les contraintes sur les plots ne sont plus aussi fortes. La présence du DES dans chaque chip 16x16, n'est donc pas nécessaire et ce dispositif ne sera pas intégré.

III.6. Cellule en panne

Les pannes dans les cellules peuvent avoir deux conséquences car les cellules synaptiques de notre architecture ont deux fonctionnalités :

- évaluation de la convergence,
- calcul synaptique (produit matrice-vecteur).

La détection de la convergence s'effectue à partir des convergence locales évaluées par les cellules diagonales. Une panne située dans les cellules peut provoquer une convergence locale erronée et par conséquent nuire à la détection de convergence.

Dans la majorité des cas de pannes logiques simples, le calcul synaptique n'est pas altéré de façon grave par la présence de pannes dans les cellules. Cependant, il ne faut pas que la densité des pannes soit trop importante.

III.7. Conclusion

Le dispositif intégré sur le bord Ouest du réseau doit évaluer la convergence globale. Ce dispositif est très simple et on peut aisément s'assurer de son bon fonctionnement. Mais la convergence globale est élaborée à partir des convergences locales évaluées par les cellules ; un mauvais fonctionnement des cellules peut alors affecter gravement la détection de convergence.
Aussi, dans l'optique WSI, il est préférable d'adopter une gestion probabiliste et de supprimer la détection de convergence.
Le dispositif d'Entrée/Sortie est critique car c'est le moyen de communication avec l'extérieur. Si la technologie le permet, un dispositif spécial pourra être intégré sur le bord Est de la tranche. Sinon, on le simplifiera au maximum afin de réduire la sensibilité aux pannes du réseau.
Le calcul synaptique bénéficie de la tolérance vis-à-vis des pannes des architectures neuroniques. Aussi, cet atout majeur permet d'envisager l'intégration du réseau systolique carré en technologie WSI.
Cependant, afin de limiter le nombre de pannes, des dispositifs sont intégrés sur la tranche afin de permettre l'élimination après fabrication des éléments présentant des défauts. Ces techniques sont l'objet du prochain chapitre.

VII. Intégration sur tranche

L'intégration sur tranche entière se fait sur une telle surface que de nombreux problèmes de conception se posent. Entre autres, la présence de défauts de fabrication est inévitable. Nous étudierons les stratégies mises en œuvre pour limiter l'impact des défauts et pour restaurer le bon fonctionnement du circuit. Nous présenterons tout d'abord les techniques qui permettent d'optimiser la reconfiguration. Puis, nous détaillerons l'étude faite sur un élément de commutation qui est le dispositif matériel de base de la reconfiguration de réseaux systoliques.

I. PROBLEMES LIES A L'INTEGRATION SUR TRANCHE

Intégrer un réseau systolique sur tranche pose différents problèmes. Nous présenterons tout d'abord les problèmes d'ordre technologique. Puis nous étudierons l'influence de ces contraintes sur le choix d'architectures adaptées à l'intégration sur tranche. Enfin nous décrirons les difficultés liées à la conception de circuits sur tranche.

Nous présenterons plus en détail les problèmes posés par la présence inévitable de défauts de fabrication qui conduit à étudier la tolérance aux pannes de l'architecture, à concevoir de façon à minimiser ces défauts et à prévoir des mécanismes permettant de ne pas utiliser les éléments défectueux.

I.1. Rappels et définitions

Dans ce chapitre, nous nous intéresserons particulièrement à l'intégration des réseaux systoliques rectangulaires car un réseau carré, du type de notre architecture neuronique, n'est qu'un cas particulier des réseaux rectangulaires.

Les éléments de ces réseaux systoliques sont localement interconnectés avec leurs voisins immédiats N, S, E et W comme le montre la figure VII.1.

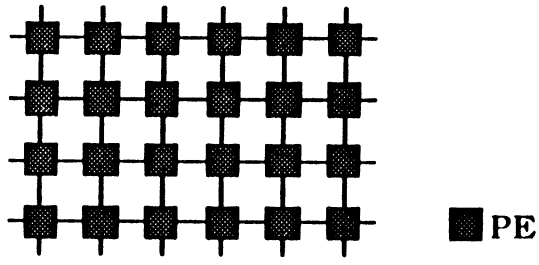


Figure VII.1 Réseau systolique rectangulaire ou 2D

Les réseaux systoliques rectangulaires sont aussi appelés réseaux 2D car ce sont des tableaux de processeurs organisés selon les deux dimensions (par opposition aux réseaux linéaires 1D).

Les éléments de ces tableaux sont couramment appelés cellules ou PE (pour Processeurs Elémentaires ou "*Processing Elements*"). Cependant lorsque la cellule est de grande taille, on préfère le terme plus représentatif de "*chip*".

L'intégration de ces réseaux de cellules sur tranche doit tenir compte des défauts et des pannes inévitables qui perturbent leur bon fonctionnement.

Les définitions présentées dans le chapitre précédent permettent de différencier défaut et panne. En fin de fabrication, la présence de défauts physiques est inévitable. Les procédures mises en œuvre à la fin de la fabrication doivent éliminer les conséquences néfastes des défauts sur le fonctionnement du circuit. Au cours de la vie du circuit, les défauts qui n'ont pas été détectés ou le vieillissement du circuit provoquent des pannes.

Les défauts de fin de fabrication sont détectables par une stratégie de test adéquate. A partir des résultats de ce test, un réseau systolique doit être créé en utilisant les bonnes ressources de la tranche. Les éléments présentant un mauvais fonctionnement doivent être remplacés par des éléments de réserve : c'est l'étape de reconfiguration. Le terme "reconfiguration" recouvre aussi bien la configuration immédiatement après la fabrication que la modification du

réseau d'interconnexion pendant la vie du circuit pour "réparer" les pannes. La reconfiguration peut être soit contrôlée de l'extérieur, soit directement implantée sur le silicium (auto-reconfiguration).
La performances d'une stratégie de reconfiguration peut s'évaluer selon plusieurs critères :

- la qualité de la "moisson", c'est à dire le taux d'utilisation des bonnes ressources de la tranche,
- la longueur maximale des interconnexions,
- la complexité des dispositifs matériels de reconfiguration.

La reconfiguration peut s'effectuer si l'interconnexion des PE est modifiable. Dans ce but, les cellules sont intégrées dans des réseaux de communication qui comprennent des éléments de commutation permettant cette modification. Aussi, on dit que le réseau est reconfigurable.

Les éléments de réserve aussi appelés éléments redondants, le réseau de communication reconfigurable et l'auto-reconfiguration s'il y en a, sont les dispositifs supplémentaires intégrés sur la tranche qui permettent d'effectuer la reconfiguration. Ils doivent être choisis avec précaution. Ils doivent permettre la reconfiguration dans de bonnes conditions tout en minimisant la surcharge en surface.

I.2. Contraintes technologiques et leur influence sur le choix d'architecture

Les contraintes technologiques sont liées à la fabrication de circuits sur des tranches et à l'encapsulage de tels circuits.

I.2.1. Encapsulage et architecture

La technologie des boîtiers a beaucoup évolué pour l'encapsulage des circuits VHSIC (Very High Speed Integrated Circuits = circuits intégrés très rapides).

Les contraintes électriques, thermiques et mécaniques imposées par ces nouveaux circuits ont nécessité l'intégration de dispositifs spéciaux pour en préserver les performances. Ces contraintes sont encore plus fortes pour l'encapsulage de tranches en boîtier [McK86], [PIT87], [VAL86]. Ces nouveaux boîtiers doivent permettre :

- la fixation de la tranche dans le boîtier,
- la dissipation de la chaleur produite par de tels circuits,
- la protection du circuit vis-à-vis de l'environnement extérieur,
- la connexion du circuit.

Les trois premières contraintes sont d'un ordre purement technologique et n'influent pas sur le choix de l'architecture à intégrer.

Par contre, le nombre limité de plots d'Entrée/Sortie de ces boîtiers peut être une contrainte forte pour la technologie WSI. L'augmentation de surface qu'apporte l'intégration sur tranche, permet une augmentation de la complexité des circuits et, donc des besoins de communication. La limitation du nombre de plots peut influencer le choix des architectures ou leur intégration.

I.2.2. Fabrication et architecture

La fabrication de circuits sur de grandes surfaces entraîne :

- des disparités des caractéristiques électriques,
- un manque de précision de la lithogravure,
- des défauts survenant lors de la fabrication.

Les processus technologiques utilisés pour les circuits VLSI atteignent leurs limites avec l'intégration sur tranche. Par exemple, les masques qui servent à la lithographie n'ont pas une précision suffisante pour graver directement la tranche. Il faut alors fractionner le circuit en réticules et procéder en plusieurs étapes. Le coût lié à la fabrication des masques limite généralement le nombre de réticules. Aussi, la plupart des architectures étudiées pour

l'intégration sur tranche sont des structures répétitives comme les mémoires ou les tableaux de processeurs.
Néanmoins, l'avènement de la lithographie par faisceau d'électrons permet d'envisager l'intégration de structures plus irrégulières.

I.2.3. Défauts, pannes et architecture

Le choix de l'architecture à intégrer doit tenir compte des pannes susceptibles de survenir pendant la vie du circuit. On a tout intérêt à choisir une architecture sur laquelle la présence de pannes a peu d'influence : architecture tolérante vis-à-vis des pannes. Comme nous l'avons déjà vu dans le chapitre précédent, les architectures neuroniques possèdent cet atout majeur.

Mais les défauts de fabrication ont beaucoup d'influence sur la technique de conception et leur présence nécessitent des dispositifs spéciaux que nous allons étudier.

I.3. Problèmes de conception

I.3.1. Types de défauts

Les défauts de fabrication peuvent être divisés en trois catégories [GEN87] :

- les défauts de paramètres : ils résultent d'un mauvais processus technologique. Ils sont détectés par le test des paramètres électriques. En principe, ils rendent la tranche entière inutilisable [WAL87],
- les défauts aléatoires : ils sont de petite taille, provoqués par des particules microscopiques. Ils peuvent provoquer, par exemple, des coupures ou des court-circuits [STA86].

- les amas de défauts : ils sont dus à des regroupements de particules sur la tranche. Leur présence nuit au bon fonctionnement d'un ou plusieurs circuits suivant leur localisation et leur taille [STA86].

Les défauts de paramètres sont facilement détectables et de toute façon conduisent au rejet de la tranche.

La taille des circuits intégrés en technologie WSI nous assure qu'ils seront touchés par les deux autres catégories de défauts. La conception de systèmes WSI doit donc en tenir compte. Il faut avoir les moyens de pouvoir remplacer les mauvais éléments par des bons.

Ces défauts doivent être détectés par la phase de test qui a lieu après la fabrication. Cependant, tous les défauts présents sur la tranche peuvent ne pas être détectés. Les défauts cachés situés dans les parties opérationnelles de la tranche se manifestent sous forme de pannes pendant la vie du circuit. Ces pannes peuvent être détectées lors des tests de maintenance. On procède alors soit au rejet du circuit, soit à la restauration de son fonctionnement si elle est possible.

I.3.2. Défauts et conception

La conception doit minimiser la sensibilité aux défauts des parties critiques. Des règles de conception spécifiques permettent d'assurer un bon rendement de ces parties. Mais le rendement n'est pas maximal et ces parties restent vulnérables aux défauts.

Comme ces mesures ne suffisent pas à éliminer la présence de défauts, la conception doit prévoir des éléments redondants. Cette redondance permet de remplacer les éléments défectueux et ainsi assurer un fonctionnement correct du circuit final.

Pour rendre cette redondance exploitable, les communications reliant les PE doivent être "flexibles" afin de pouvoir connecter les bonnes cellules de la

tranche. Les canaux de communication doivent donc intégrer des dispositifs de commutation pour être reconfigurables.

I.3.3. Distribution des horloges et alimentations

Une des principales difficultés de conception réside dans la distribution des signaux de contrôle et d'alimentation.
Parmi tous les signaux de contrôle, les horloges sont les plus critiques. Leur distribution conditionne directement les performances du circuit final. Distribuer un signal d'horloge correct sur toute la tranche est délicat car les horloges ont une grande sortance, travaillent à "haute" fréquence et doivent être synchronisées.
Il faut aussi que le circuit soit correctement alimenté. Le système d'alimentation doit satisfaire les conditions suivantes [FRI85] :

- atténuer les variations de courant,
- résister aux bruits,
- ne pas perturber la distribution des autres signaux.

L'alimentation d'un circuit WSI pose d'autres problèmes et doit en plus :

- pouvoir supporter la puissance nécessaire,
- pouvoir être réparée afin qu'un court-circuit n'affecte pas tout le circuit d'alimentation,
- atténuer les disparités de charge qui surviennent après la reconfiguration,
- satisfaire les contraintes imposées par la technologie (réticule).

Toutes ces contraintes doivent être satisfaites par les systèmes de distribution des signaux et alimentations.

I.4. Reconfiguration de réseaux systoliques rectangulaires

A partir du réseau de cellules implanté sur la tranche, la reconfiguration d'un réseau systolique rectangulaire doit créer le réseau voulu en éliminant les

cellules défectueuses. Il s'agit donc d'obtenir un réseau systolique rectangulaire de M cellules valides (réseau cible) à partir d'une tranche comportant N cellules (réseau physique). N est évidemment supérieur à M et leur différence représente les éléments redondants qui sont en réserve.
La matrice cible est immergée sur les bonnes cellules du réseau physique (phase de placement) puis l'interconnexion des processeurs choisis est effectuée (phase de routage).
Le test après la fabrication permet d'établir une cartographie des cellules non défectueuses afin de pouvoir procéder au placement. Les algorithmes de placement doivent prendre en compte les contraintes liées à la double dimension du réseau systolique (connexions horizontales et verticales).
La phase de routage nécessite des canaux de communication configurables. Les éléments détectés non valides ainsi que les éléments rejetés par la phase de placement sont déconnectés. Les éléments choisis par le placement pour composer la matrice cible sont inter-connectés.

Les algorithmes de reconfiguration (placement + routage) sont nombreux. Toutefois, il faut savoir les adapter en fonction de l'architecture à reconfigurer. Dans le cas des réseaux rectangulaires de processeurs systoliques qui nous intéresse, les contraintes topologiques liées aux deux dimensions du réseau sont fortes : l'algorithme de reconfiguration doit les satisfaire. La taille du processeur élémentaire est aussi une des caractéristiques essentielles d'un réseau. Si l'unité de reconfiguration est de petite taille, on parle de reconfiguration à grain fin. Dans le cas contraire, il s'agit de reconfiguration à gros grain.
De même, la structure des réseaux de communication et des éléments de commutation qui y sont intégrés est influencée par la taille du grain.
Les techniques mises en place pour la reconfiguration dépendent donc essentiellement de la grosseur de ce grain de reconfiguration.

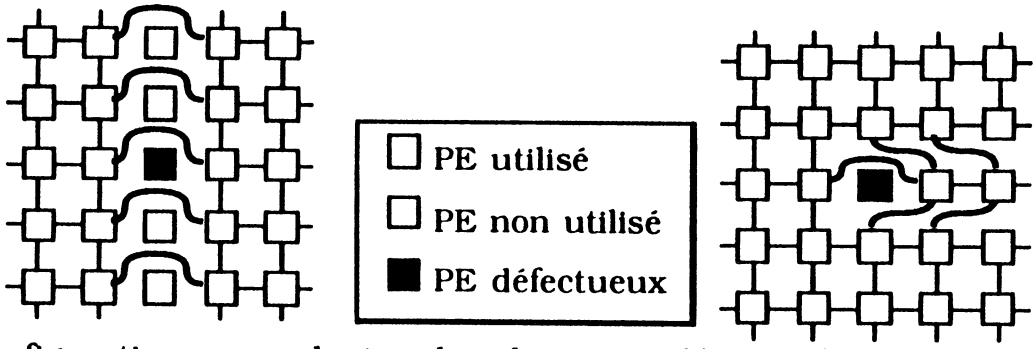
Un grain fin sera appelé cellule ou PE (Processeur Elémentaire, "*Processing Element*"). Pour un gros grain, on utilisera essentiellement le terme de chip.

I.4.1. Reconfiguration à grain fin

Avec un grain fin, les PE sont de faible taille : il est donc possible d'en intégrer un grand nombre sur la tranche. Il s'agit alors de constituer des réseaux de grande taille (grand nombre de PE). Vu la faible complexité de ces cellules, les dispositifs de commutation doivent être simples car ils sont nombreux sur la tranche et la surcharge en surface de la redondance et de la reconfiguration ne doit pas être trop importante.

Au vu du faible coût relatif de chaque PE, l'algorithme de reconfiguration rejette certaines cellules qui fonctionnent correctement, mais qui pénalisent le routage ou allongent de façon excessive les connexions. On utilise alors des algorithmes qui n'utilisent pas toutes les bonnes cellules de la tranche.

Parmi ceux-ci on peut noter l'algorithme simple de l'exclusion de colonne ou de ligne [HED82]. Une colonne ou une ligne entière est éliminée et contournée si au moins une de ses cellules est endommagée (cf figure VII.2a). Les dispositifs de reconfiguration sont extrêmement simples. Cependant cette méthode est très coûteuse car elle nécessite beaucoup d'éléments de réserve.



a) reconfiguration par exclusion de colonne. b) reconfiguration par ligne.

Figure VII.2 Techniques de reconfiguration

La reconfiguration par ligne (cf figure VII.2b) de Moore et Mahat [MOO85] est plus économique. Le nombre de cellules utilisables par ligne est limité par le minimum des nombres de cellules bonnes dans chaque ligne du réseau. Les dispositifs de reconfiguration sont assez simples mais néanmoins plus complexes que dans le précédent. Aussi est-il plus sensible aux défauts. L'efficacité est très limitée en cas de faible rendement.

Enfin, on peut citer la "*patching method*" développée par Leighton [LEI86]. Elle est basée sur un découpage dichotomique du réseau physique en régions. Le nombre de cellules dans chaque région est déterminé en fonction du rendement de la tranche. Il est fixé pour que chaque région contienne un nombre minimal de bonnes cellules (m) pour construire un sous-réseau. Si le nombre de bonnes cellules d'une région n'est pas suffisant (inférieur à m), la région complète est inutilisée. Dans le cas contraire, l'algorithme choisit les m cellules qui minimisent les connexions.

I.4.2. Reconfiguration à gros grain

Les contraintes imposées par un gros grain de reconfiguration sont totalement différentes. Les cellules sont d'une plus grande complexité et leur nombre sur la tranche est réduit : on parle alors de "*chip*". Les réseaux sont donc relativement de petite taille (faible nombre de chips). Les communications entre chips sont généralement plus importantes et les chemins de données à reconfigurer sont plus larges.

Le rendement est généralement faible car la probabilité de panne est proportionnelle à la surface occupée. Il s'agit donc d'utiliser au maximum les processeurs valides de la tranche pour constituer le réseau opérationnel. Les algorithmes de reconfiguration doivent donc être très efficaces afin d'optimiser la moisson car les pertes coûtent cher.

Le système de reconfiguration automatique CRAWL [KOU88] est dédié à la reconfiguration de réseaux systoliques. Pour les réseaux rectangulaires, trois approches sont possibles comme le montre la figure VII.3 :

- immersion par ligne,
- immersion par colonne,
- immersion par secteur.

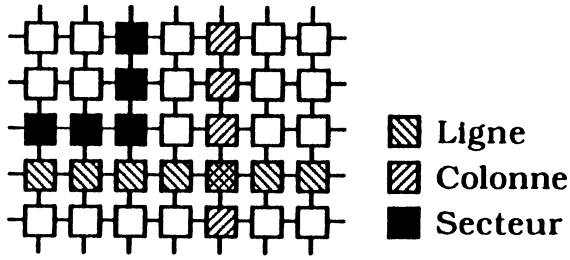


Figure VII.3 Immersion par ligne, colonne ou secteur

Les trois algorithmes fonctionnent de façon similaire. Le point de départ est déterminé par le placement du processeur de la première ligne et de la première colonne du réseau cible. Ensuite, les structures (ligne, colonne ou secteur) sont récursivement immergées dans le réseau physique. A chaque étape, le routage des connexions entre processeurs est effectué en concurrence avec le placement. Si le routage échoue, le placement est remis en cause. L'interconnexion physique des processeurs est donc prise en compte.

L'algorithme *"divide and conquer"* de Leighton [LEI86] permet d'utiliser théoriquement toutes les bonnes cellules de la tranche. Cet algorithme se décompose en deux étapes. En premier lieu, la tranche est récursivement partagée en deux et les bonnes cellules de chaque partie sont dénombrées. L'algorithme définit les sous-réseaux qui peuvent être construits dans chaque partie. Ce premier pas se poursuit tant que l'optimisation locale ne peut être réalisée facilement. D'après Leighton, le nombre de cellules par partie doit

être de l'ordre de Log(N) où N est le nombre de cellules de la tranche. On passe au deuxième pas de l'algorithme lorsque le nombre de cellules par partie est suffisamment faible pour que la construction du sous-réseau soit simple. On peut alors soit procéder de façon systématique en consultant des tables de configuration pré-établies, soit mettre en œuvre un autre algorithme adapté à un grain de reconfiguration fin.

A l'opposé de ces algorithmes conventionnels qui effectuent une reconfiguration dirigée depuis l'extérieur, Evans adopta une approche originale fondée sur l'auto-reconfiguration [EVA85].
A chaque cellule est associée une logique de contrôle qui permet l'auto-configuration de la matrice de cellules. Les cellules déclarées valides par le test de fin de fabrication (l'idéal étant un self-test intégré sur le silicium), envoient des requêtes de connexions à leurs voisines. A partir de ces requêtes, la logique de contrôle de chaque cellule réalise effectivement les connexions nécessaires.
L'idée est très séduisante car la reconfiguration s'effectue sans aucune intervention extérieure. Il est préférable d'utiliser cette technique avec un gros grain de reconfiguration car l'augmentation de surface due à la logique est trop importante avec un grain de reconfiguration fin. Mais l'augmentation de la taille du processeur entraîne une baisse du rendement et l'auto-reconfiguration est peu efficace en cas de rendement faible. De plus, la présence de pannes dans la logique de contrôle risque de conduire à une reconfiguration comportant des éléments défectueux ou à l'échec de l'auto-reconfiguration.

I.4.3. Reconfiguration hiérarchique

Les approches par partitionnement ("*patching method*" et "*divide and conquer*") laissent entrevoir une solution intermédiaire qui peut être mise en

place dans le cas particulier de grands tableaux de petits processeurs. C'est d'ailleurs le cas dans lequel se situe le réseau neuronique que nous avons défini précédemment. Pour de telles architectures une reconfiguration à grain fin est naturelle. Mais les dispositifs de reconfiguration seraient nombreux et la perte de place risquerait d'être importante. D'où l'idée de grossir le grain afin de pouvoir mettre en place une reconfiguration à plusieurs niveaux : reconfiguration hiérarchique.

Au plus haut niveau de cette hiérarchie, le grain de reconfiguration est constitué de sous-réseaux de cellules élémentaires (appelés par la suite "chips"). Il s'agit d'une reconfiguration à gros grain. A l'intérieur de ces chips, une reconfiguration à grain fin peut être mise en place pour améliorer le rendement des chips.

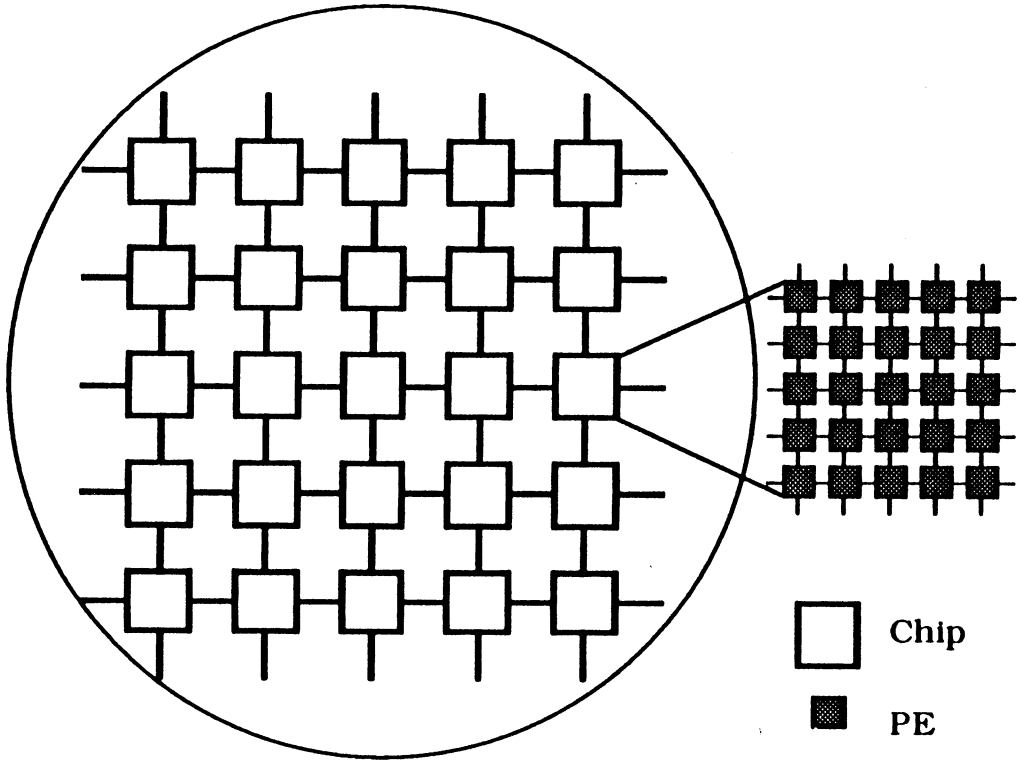


Figure VII.4 Reconfiguration hiérarchique

Cette méthode a été utilisée pour la reconfiguration de la machine WASP [HED82]. La tranche est divisée en régions de 12 cellules, dans lesquelles on

construit un sous-réseau carré de 4 cellules. La reconfiguration par exclusion de ligne ou de colonne d'Hedlund qui est considérée comme coûteuse lorsqu'elle est employée à l'échelle de la tranche, s'avère très efficace au niveau de la région. Mais la redondance reste néanmoins très importante.
La reconfiguration hiérarchique a aussi été utilisée dans le projet Esprit WSI [IVE87], [HUR87]. Les PE du réseau systolique carré sont regroupés pour former des chips. Chaque chip intègre une colonne de PE supplémentaire qui constitue la redondance. Ainsi le taux de redondance reste dans des proportions raisonnables. Une reconfiguration par ligne, proche de celle développée par Moore et Mahat [MOO85], est utilisée au niveau du chip. Au niveau de la tranche, les chips sont intégrés dans un réseau de communication reconfigurable.

I.4.4. Conclusion

La localisation des défauts détectés par le test de fin de fabrication permet de déterminer les bonnes ressources de la tranche. A partir de ces données, la reconfiguration permet la constitution d'un réseau systolique opérationnel.
Les algorithmes de reconfiguration sont extrêmement nombreux. Les recherches sont très actives dans ce domaine, par exemple [SAM86], [LOM87], [DIS88], [NEG85].

Les algorithmes de reconfiguration travaillent souvent sur un plan théorique en supposant que l'interconnexion des processeurs ne pose pas de problèmes majeurs. Ils tiennent compte de la longueur des connexions qui est la fonction de coût à minimiser, mais l'implantation physique de ces connexions n'est pas forcément prise en compte. Pour être plus près de la réalité, les algorithmes doivent aussi satisfaire les contraintes liées au routage des connexions comme le fait le système CRAWL qui effectue concurremment placement et routage [KOU88].

Les algorithmes de reconfiguration ne peuvent pas être analysés uniquement sur un plan théorique. Ils doivent également être performants sur la tranche. Il faut aussi que les dispositifs matériels de reconfiguration permettent d'implanter physiquement les résultats de ces algorithmes. Il s'agit donc de trouver une bonne adéquation entre l'algorithme et le système à reconfigurer.

II. LES RESEAUX DE COMMUNICATION

Pour pouvoir effectuer la reconfiguration, les cellules du réseau systolique doivent être intégrées dans un réseau de communication reconfigurable. Ce réseau comprend des éléments de commutation qui permettent de modifier les interconnexions entre cellules.

Nous étudierons plusieurs réseaux de communication et leurs performances relatives. Nous avons abordé cette étude de performances suivant trois points :

- la qualité de la moisson réalisable grâce aux réseaux,
- le nombre maximal d'éléments de commutation sur les connexions après reconfiguration,
- la complexité des éléments de commutation des réseaux.

Nous présenterons d'abord les différents réseaux et leurs performances quant à la qualité de la moisson. Puis, nous comparerons les délais induits par les éléments de commutation des interconnexions reconfigurables. Nous finirons cette étude par l'analyse détaillée des éléments de commutation.

II.1. Différents réseaux

De nombreux réseaux de communication ont été étudiés pour effectuer la reconfiguration. Nous en présentons ici 4 exemples.

II.1.1. Réseau de Moore et Mahat

Le réseau de Moore et Mahat [MOO85] permet une reconfiguration en décalant les communications verticales comme le montre la figure VII.5. Elle a été

développée pour utiliser la reconfiguration par ligne développée par les mêmes auteurs.

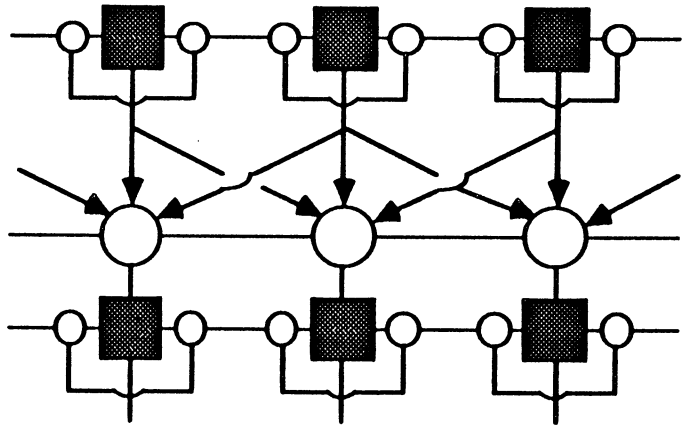


Figure VII.5 Réseau de Moore et Mahat

Le réseau de Moore et Mahat apporte une faible surcharge en surface car il est simple. Mais ses capacités de reconfiguration sont limitées. Lorsque plusieurs PE sont défectueux (marqués d'un double X figure VII.6) sur la même ligne, le décalage des connexions verticales devient important. Dans ce cas, l'algorithme de reconfiguration est obligé de laisser de côté des PE valides (marqués d'un double U pour "*unused*" figure VII.6) car ils sont hors de portée comme le montre la figure VII.6. Ce réseau de reconfiguration induit donc une moisson de mauvaise qualité.

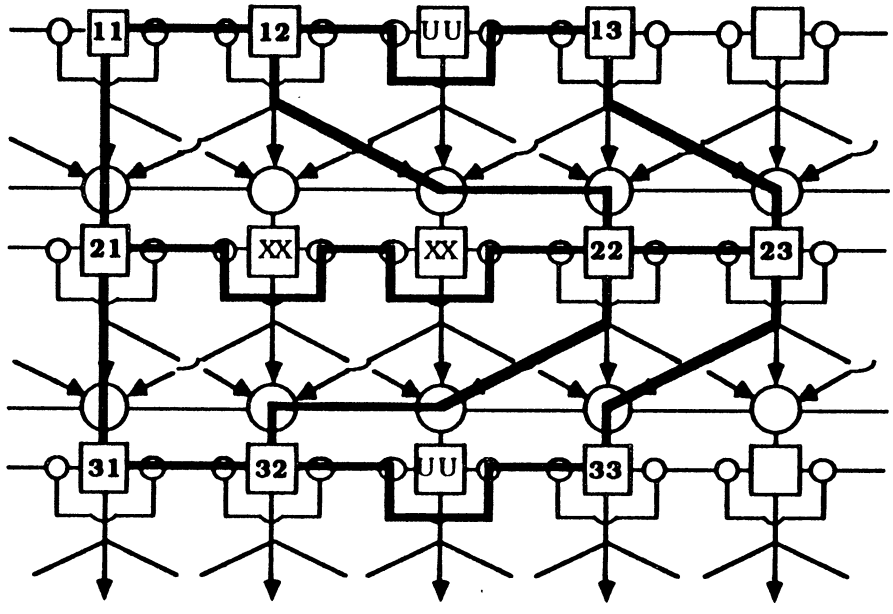


Figure VII.6 Exemple de reconfiguration avec le réseau de Moore et Mahat

Le réseau de Moore et Mahat est rapidement mis en échec lorsque plusieurs éléments sont défectueux sur la même ligne. Un décalage des communications verticales ne suffit pas. Ce réseau respecte donc mal les contraintes liées aux deux dimensions des réseaux rectangulaires.

II.1.2. Réseaux d'Hedlund

Hedlund a proposé deux réseaux de reconfiguration assez proches. Ce sont des réseaux orthogonaux qui respectent mieux les contraintes liées aux deux dimensions des réseaux rectangulaires.
Nous allons d'abord étudier le réseau simple rail montré par la figure VII.7.

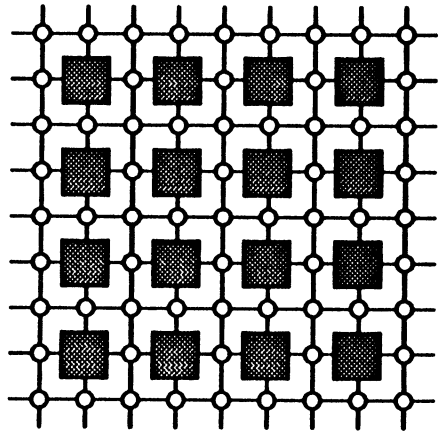


Figure VII.7 réseau simple rail

Sur le même exemple que précédemment, la figure VII.8 montre les performançes de reconfiguration qu'apporte le réseau simple rail.

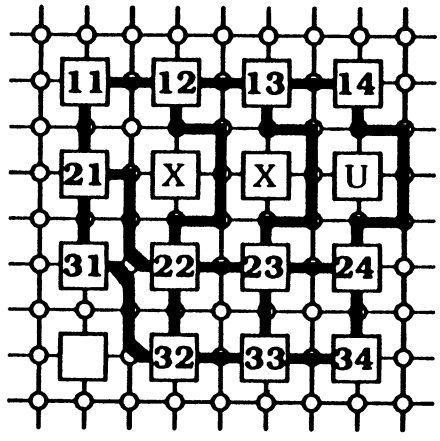


Figure VII.8 Exemple de reconfiguration pour le réseau simple rail

Ce réseau n'apporte toujours pas de solution pour plusieurs PE défectueux sur la même ligne. La moisson n'est encore pas optimale.

Pour améliorer la moisson, Hedlund a été conduit à augmenter la densité du réseau : le réseau double rail (cf figure VII.9).

Hedlund a développé initialement ce réseau pour une reconfiguration à grain fin [HED82]. Mais il peut aussi être utilisé pour une reconfiguration à gros grain. Ce réseau permet d'obtenir une meilleure reconfiguration car il est plus souple. Sa plus grande densité assure une meilleure résistance aux pannes [FRA87].

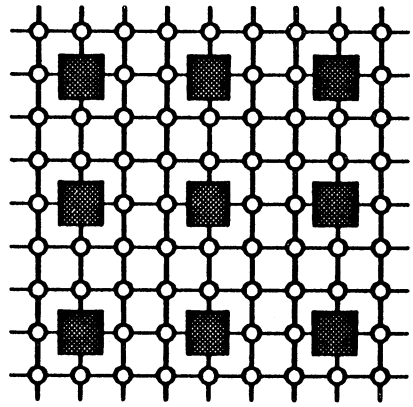


Figure VII.9 Réseau double rail de Hedlund

Grâce à sa grande densité d'éléments de commutation, ce réseau permet une reconfiguration efficace et une bonne moisson comme le montre la figure VII.10.

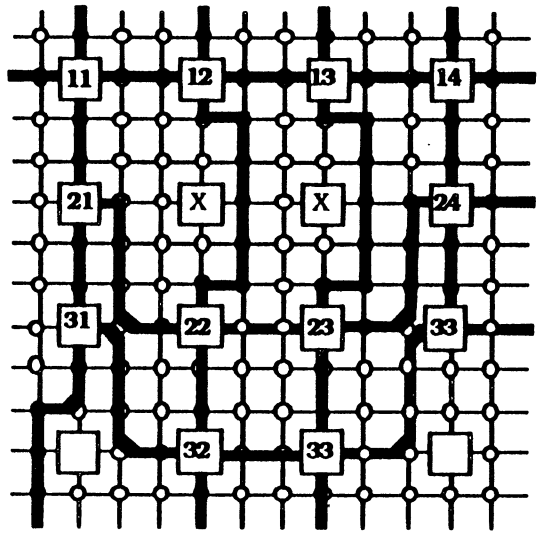


Figure VII.10 Exemple de reconfiguration avec le réseau double rail

II.1.3. Réseau de Franzon

Le réseau proposé par [FRA86] est assez proche du précédent, comme le montre la figure VII.11. Chaque chip est entouré d'un anneau de commutateurs qui permet de contourner ou d'accéder au chip.

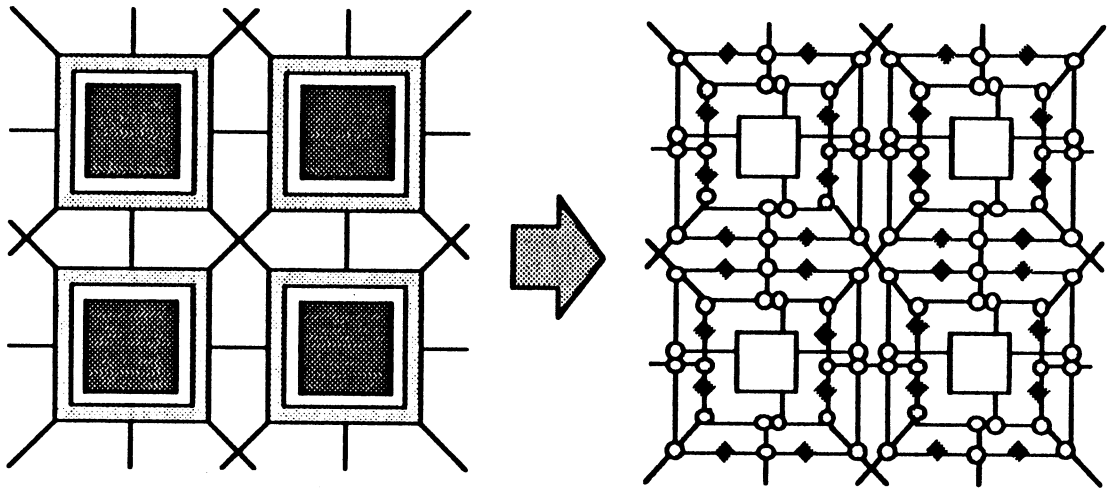


Figure VII.11 Réseau de Franzon

Les éléments de commutation (cercles de la figure VII.11) sont d'un type particulier comme le montre la figure VII.12. Ils autorisent soit le croisement, soit un court-circuit entre les deux bus de données. Lorsqu'un élément de commutation autorise le croisement (C=0), aucun dispositif n'affecte la transmission des signaux sur les bus. Si la commande C est à "1", les deux bus sont en court-circuit et le signal traverse une porte de transfert.

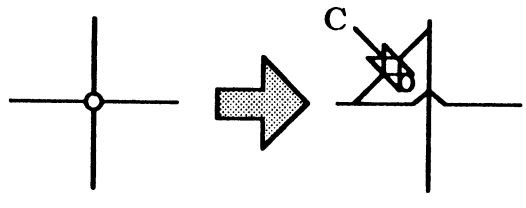


Figure VII.12 Elément de commutation de Franzon

D'autre part, les portes de transfert (losanges gris de la figure VII.11) permettent de définir des segments de bus et ainsi définir les contournements.

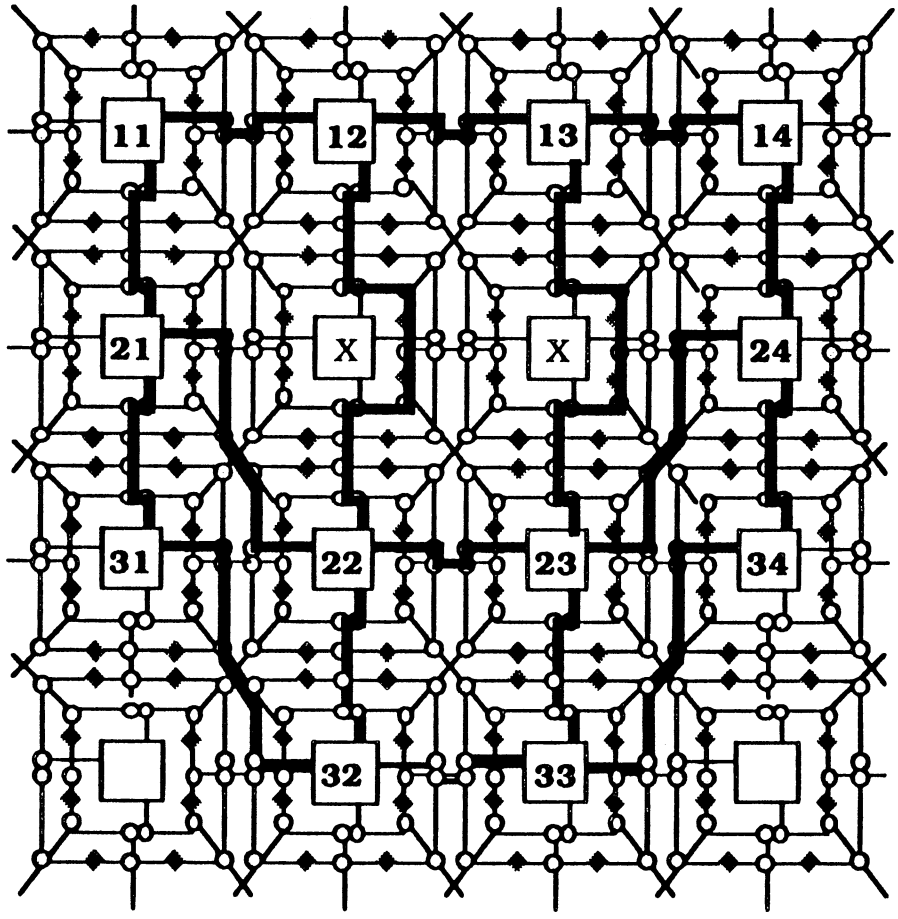


Figure VII.13 Exemple de reconfiguration avec le réseau de Franzon

Avec un tel réseau, la moisson est bonne et aucun élément valide n'est laissé de côté.

II.1.4. Etude comparative

Les performances de reconfiguration ne se limitent pas à la qualité de la moisson. Il faut aussi minimiser la longueur des interconnexions. Dans les interconnexions, les dispositifs de commutation introduisent des délais proportionnels au nombre d'éléments de commutation traversés.

Aussi, nous allons procéder à l'étude comparative entre les deux derniers réseaux qui présentaient déjà de bonnes performances quant à la moisson.

Si nous étudions les exemples de reconfiguration présentés par les figure VII.10 et figure VII.13, nous remarquons que le réseau double rail présente de bonnes performances quant à la longueur des chemins. La plus longue connexion avec le réseau de Franzon comprend 8 éléments de commutation alors qu'avec le réseau double rail, cette même connexion est limitée à 7 éléments de commutation.

Mais, pour ne pas limiter notre étude à un cas particulier, nous avons défini quatre chemins qui sont les contournements les plus courants dans la reconfiguration de réseaux rectangulaires.

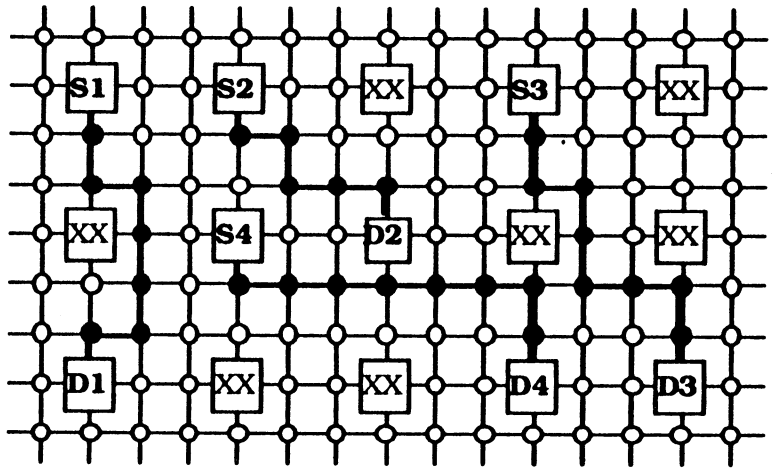


Figure VII.14 Exemples de contournement d'un PE sur le réseau double-rail

Sur le réseau double rail, les chemins 3 (S3 à D3) et 4 (S4 à D4) représentent les plus longues connexions car ils comprennent 8 éléments de commutation, ce qui correspond à 8 portes de transfert.

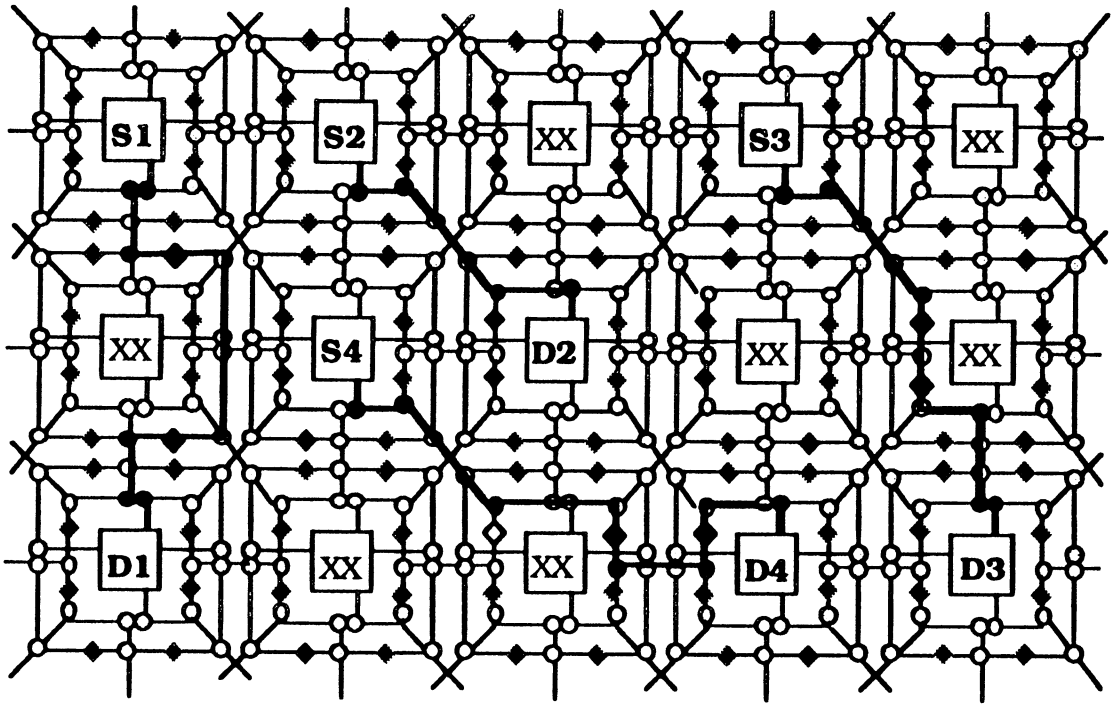


Figure VII.15 Exemples de contournement d'un PE sur le réseau de Franzon

Sur le réseau de Franzon, on remarque que les chemins 1, 3 et 4 sont de longueur maximale. Ils comprennent chacun 8 éléments de commutation (les éléments de commutation qui induisent des délais sont en noir sur la figure VII.15).

Les réseaux comparés ont donc des performances très proches. Cependant, le réseau de Hedlund présente les deux avantages suivants :

- les configurations des éléments de commutation peuvent être contrôlées indépendamment les unes des autres. Ainsi le réseau de communication est plus tolérant aux pannes.
- la structure très régulière du réseau facilite l'intégration.

On peut en conclure que le réseau double rail proposé par Hedlund a de bonne performances quant à la qualité de la moisson. Une bonne reconfiguration peut être effectuée sans allongement rédhibitoire des interconnexions. La tolérance aux pannes et la régularité du réseau sont des atouts importants pour

l'intégration. Aussi nous choisirons un réseau double rail, comme réseau de communication pour notre architecture systolique qui est très proche de celle présentée dans [IVE87].

Afin de compléter notre étude comparative, nous devons poursuivre avec l'analyse détaillée des différents éléments de commutation.

II.2. Eléments de commutation

Les réseaux pour être flexibles doivent intégrer des éléments de commutation. Ces éléments peuvent être de différente nature.

Les fusibles et anti-fusibles laser qui sont au stade de développement industriel [NIC86], programment de façon permanente les éléments de commutation. Ils permettent aussi de déconnecter les processeurs présentant un mauvais fonctionnement et d'effectuer des réparations sur les lignes de distribution.

Les portes à grille flottante sont plus souples car leur programmation qui se fait par faisceau d'électrons est théoriquement réversible. Cette technique est encore à l'état de prototype et sa fiabilité doit encore être améliorée.

Tout comme les dispositifs laser, les portes à grille flottante nécessitent un environnement de programmation lourd.

Enfin, les portes de transfert contrôlées par des portes logiques sont la technologie la plus traditionnelle. Ces éléments de commutation sont programmés depuis le bord de la tranche par l'envoi de commandes appropriés. Cette programmation peut se faire dans l'environnement d'utilisation normale. La configuration après fabrication et la reconfiguration après détection d'une panne durant le fonctionnement du circuit, sont donc possibles.

Les éléments de commutation étudiés sont constitués d'un ensemble de portes de transfert qui permettent la commutation des chemins de données. Leur programmation pose un problème d'accessibilité.

Katevenis a résolu ce problème en distinguant chaque élément de commutation par une adresse qui est fixée à la fabrication [KAT86]. Les données de programmation contiennent l'adresse du destinataire.

La programmation des éléments de commutation nécessaires au réseau de Franzon (cf figure VII.11) est faite par un registre à décalage qui parcourt tous les éléments de commutation du réseau. Ce registre permet d'amener les données de programmation aux différents éléments du réseau de communication. En cas de panne dans le registre à décalage, tous les éléments de commutation situés en aval de la panne sont inaccessibles. Les données de programmation codent l'état de tous les dispositifs de commutation d'un même chip. Ce codage se fait sur 13 bits et nécessite un décodage câblé assez complexe. Si cette partie contrôle est défectueuse, tous les éléments de commutation sont inexploitables et le chip est inaccessible. Ce décodage augmente donc la partie critique des éléments de commutation et réduit la tolérance aux pannes du réseau de communication.

Un autre mode de programmation a été spécifié par [CHE85]. Il utilise la notion de chemin. Un chemin est une succession d'éléments de commutation. La création d'un chemin s'effectue progressivement en configurant les différents éléments de ce chemin, les uns après les autres.

Cette technique de programmation a été initialement conçue pour la communication inter-processeurs. Nous la détaillons dans le prochain paragraphe car nous avons adapté l'architecture de cet élément de commutation au problème particulier qui nous intéresse : la reconfiguration de réseaux systoliques rectangulaires.

III. ARCHITECTURE D'UN ELEMENT DE COMMUTATION

Nous définissons ici un élément de commutation, appelé Switch dont le principe de base a été spécifié par [CHE85].
Une fois ce principe rappelé, nous détaillons la programmation du Switch. Son adaptation au problème de la reconfiguration a nettement modifié son architecture et son intégration.

III.1. Principe de base

Le principe de base de cet élément de commutation a été développé pour la communication inter-processeurs [CHE85]. Pour communiquer avec un autre processeur ou pour accéder à un périphérique, un processeur devait pouvoir établir lui-même la connexion. Une programmation par chemin avait été développée pour répondre à ces spécifications.
Supposons les processeurs reliés par un maillage d'éléments de commutation. Lorsqu'un processeur P1 désire établir une connexion avec le processeur P2, un chemin entre P1 et P2 doit être établi comme le montre la figure VII.16. Le chemin se construit en 4 pas. A chaque pas un nouvel élément de commutation est ajouté au chemin.

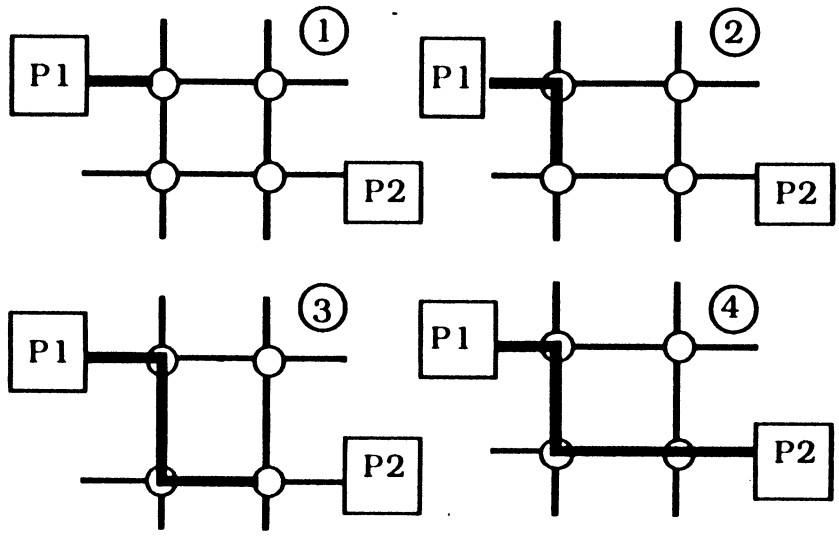


Figure VII.16 programmation par chemin

Le chemin de Switchs est construit pas à pas, en ajoutant successivement les Switchs du chemin : les données de programmation sont communiquées au Switch à ajouter via le début du chemin déjà construit.

Les éléments de commutation reçoivent donc leurs données de programmation via le bus de données à commuter. Chaque extrémité (N, S, E ou W) de l'élément de commutation est dotée d'une logique de contrôle (voir figure VII.17) qui permet de configurer le Switch et ainsi d'établir la connexion désirée.

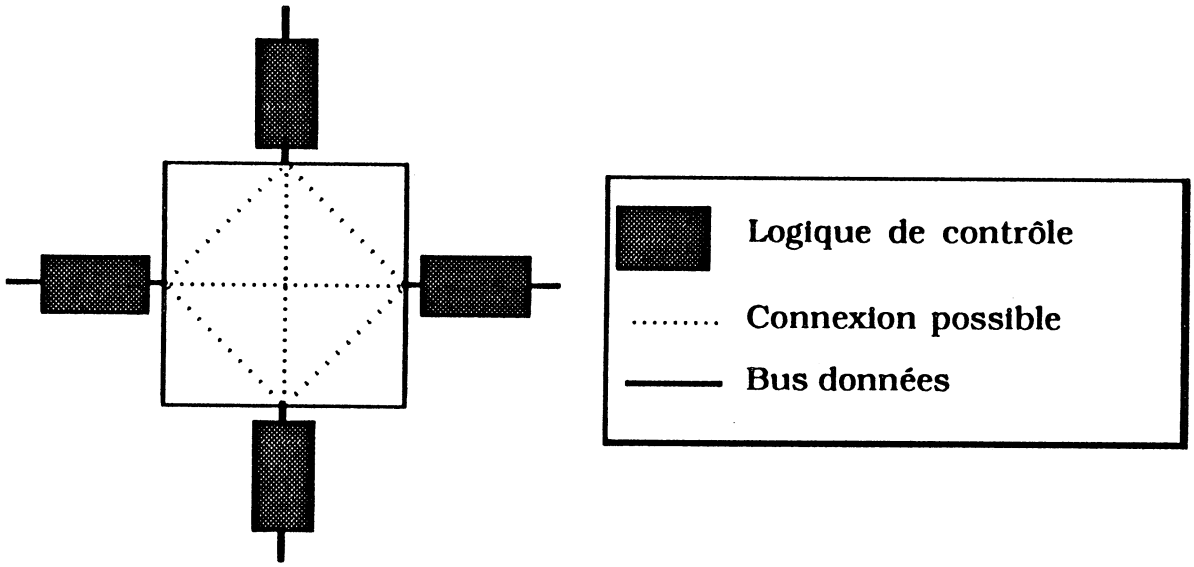


figure VII.17 Elément de commutation programmable

Quatre configurations sont possibles pour chaque extrémité comme le montre la figure VII.18.

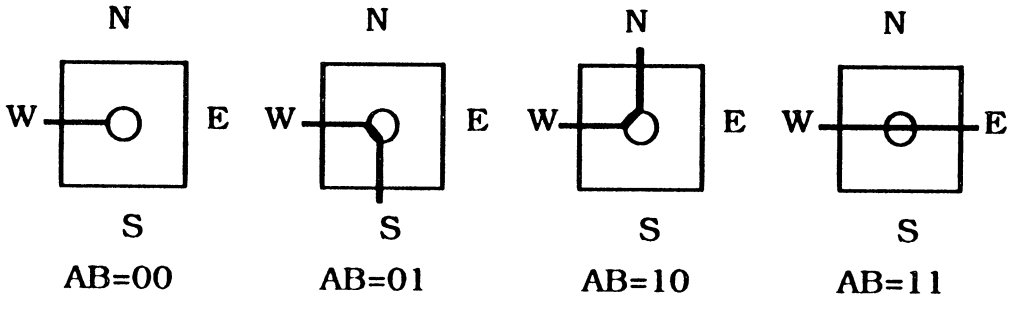


Figure VII.18 Configurations possibles

Pour établir ces configurations, la logique de contrôle reçoit un signal PROG qui lui signale que les données qui arrivent sur le bus de données sont des données de programmation et de synchronisation qui lui sont destinées. Sur le front descendant du signal PROG, la partie contrôle établit la configuration grâce aux données qu'elle a reçues.

Le signal PROG n'est pas un signal global. Il se propage de Switch en Switch et subit les commutations comme les données.

III.2. Modifications apportées pour la reconfiguration

Dans le cas de la reconfiguration de réseaux systoliques 2D qui nous intéresse, l'immersion de la matrice cible sur les bonnes ressources de la tranche se fait par construction de chemins horizontaux et verticaux (cf figure VII.10). Ces chemins relient les bords opposés de la tranche et traversent les chips.
L'élément de commutation basé sur la notion de chemin est donc tout à fait adapté à notre problème. Les données de programmation issues du bord de la tranche doivent pouvoir traverser les chips de part en part.
Pour pouvoir inclure cet élément de commutation dans un réseau de communication reconfigurable, quelques adaptations sont nécessaires. Dans le cas de la reconfiguration de réseaux systoliques rectangulaires, les communications entre PE sont simples. Il n'est donc pas nécessaire d'avoir quatre parties contrôle différentes pour pouvoir établir les configurations. Nous avons donc regroupé la logique de contrôle dans une partie contrôle unique. Ceci présente l'avantage de minimiser le contrôle et d'éviter les conflits de programmation.

De plus, les données de programmation et de synchronisation étaient transmises par le bus de données dans la première version. La récupération de

ces données nécessitait la présence de dispositifs (portes de transfert) supplémentaires sur certains fils du bus de données. Pour éviter les délais engendrés par ces dispositifs, les données de programmation et de synchronisation sont maintenant envoyées par un fil supplémentaire dédié à la programmation nommé DATA.

Enfin, dans le cadre de la reconfiguration, les dix configurations qu'offrait la première version, ne sont pas nécessaires. L'élément de commutation doit permettre de commuter les bus d'interconnexion dans les quatre directions comme le montre la figure VII.19 :

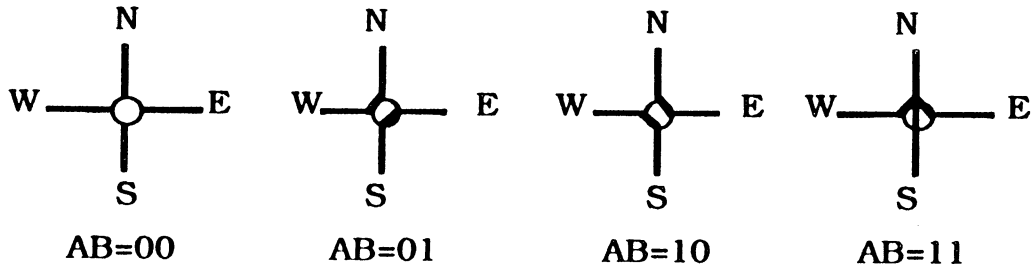


Figure VII.19 Fonctionnalités du Switch

Ces diverses commutations peuvent être réalisées logiquement par un ensemble de 6 portes de transfert présenté figure VII.20.

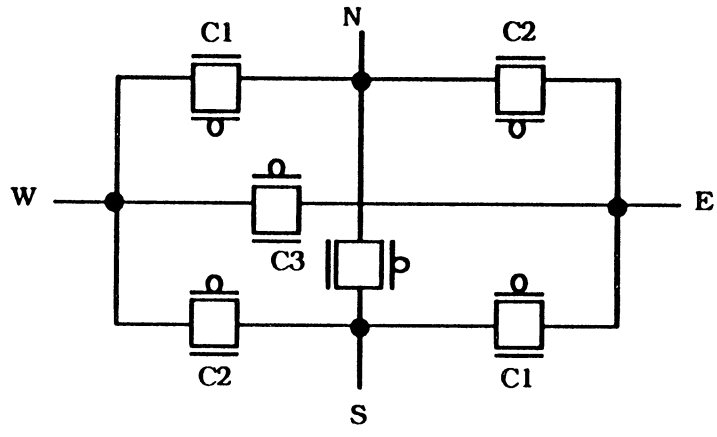


Figure VII.20 Partie Commutation

Pour commuter un bus de N fils, il faut intégrer N parties Commutation programmées de façon identique.

Les commandes C_1, C_2 et C_3 qui contrôlent les portes de transfert sont définies à partir du code de configuration AB comme le montre la figure VII.21.

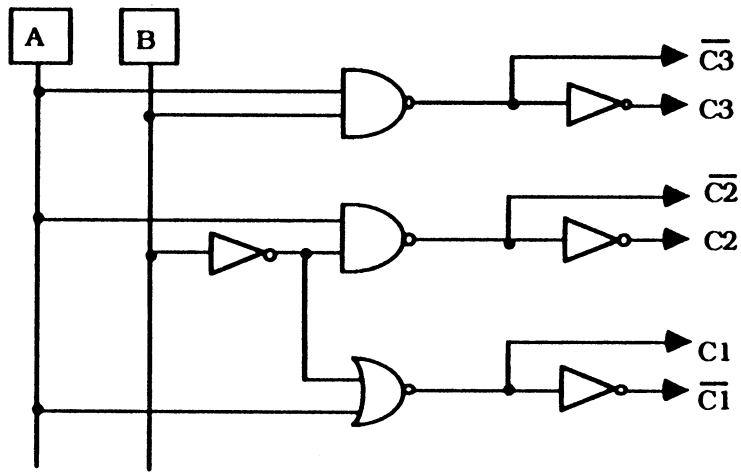


Figure VII.21 Décodage

A et B sont des registres binaires. Programmer le Switch se résume à mémoriser la configuration dans ces registres.

III.3. Programmation du Switch

La programmation n'utilise pas de fils spécialisés reliant directement les Switchs au bord de la tranche : ces connexions seraient vraiment trop pénalisantes. La programmation fait appel à la notion de chemin : elle se fait en série de Switch en Switch.

Outre les fils réservés aux données qui transitent entre deux Switchs, deux fils supplémentaires sont nécessaires pour la programmation :

- PROG,
- DATA.

Ces signaux ne sont pas des signaux globaux. Ce sont des fils supplémentaires ajoutés au bus de données qui jouent un rôle particulier dans la programmation. Mais ils sont commutés comme les autres fils du bus de données. Ils se propagent de Switch en Switch.

Au début, les Switchs sont initialisés avec la configuration AB=00 pour isoler les fils N, S, E et W. Cette initialisation est effectuée par le signal Reset. Le signal Reset est un signal global d'initialisation du réseau de communication. Nous décrivons la méthode de programmation de 2 Switchs SW_1 et SW_2 montrés sur la figure VII.22. Nous rappelons que les chips sont transparents pour les données de programmation. Aussi, les chips n'apparaissent pas dans les dessins expliquant la méthode de programmation.

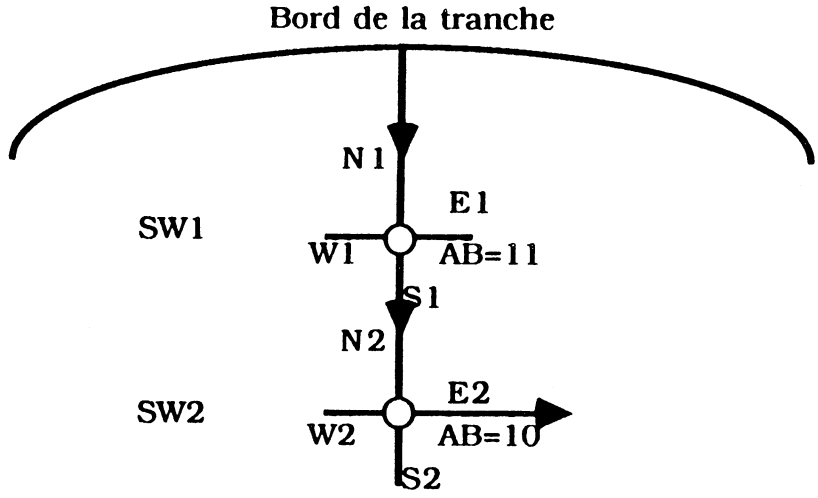
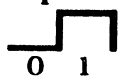


Figure VII.22 Programmation de SW_1 et SW_2

La programmation de ces 2 Switchs est effectuée en deux pas. Comme la programmation se fait en série, pour programmer SW_2, il faut d'abord avoir configuré SW_1. Une fois SW_1 programmé avec AB=11, les données de la programmation peuvent atteindre SW_2.

Pour mettre en place cette programmation, chaque Switch doit contenir un registre à décalage de 2 bits dans lequel transitent les données de configuration. Sur le front descendant du signal PROG, les valeurs stockées dans le registre à décalage sont mémorisées dans les registres A et B.

Notation : la programmation est faite en série. Pour configurer un Switch avec AB=10, le signal DATA doit prendre successivement les valeurs 0 puis 1, que nous noterons pour plus de clarté [10].

[10] = 

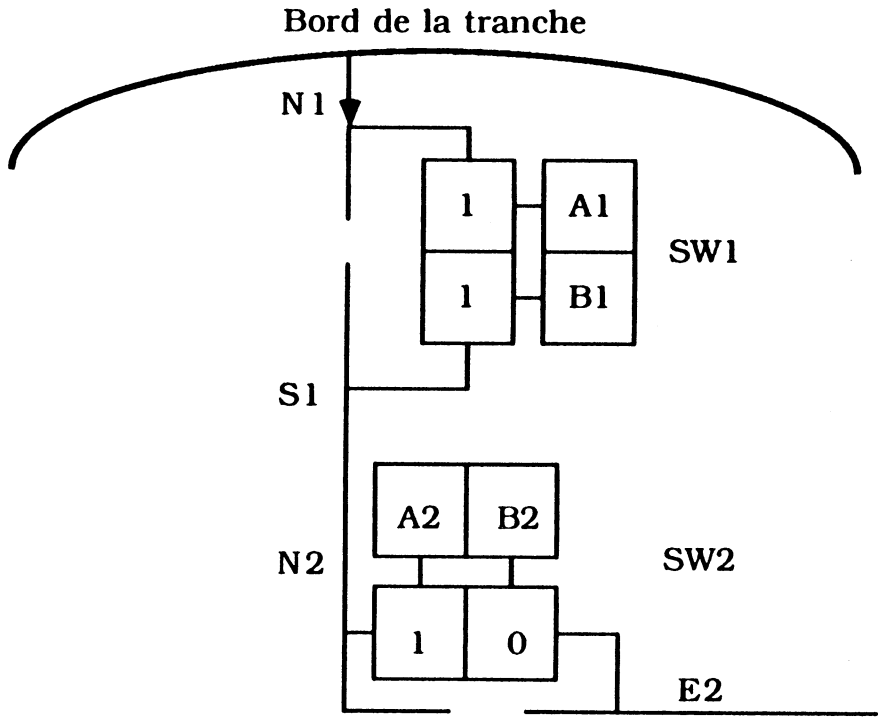


Figure VII.23 Résultat de la programmation

Détaillons la configuration des Switchs SW_1 et SW_2 :

- Etape 1 : il faut tout d'abord configurer SW_1 avec 11 :

* le signal $PROGN_1$ est mis au niveau haut,
* la séquence [11] est envoyée à SW_1 par le signal $DATAN_1$. Cette séquence signifie que le SW_1 doit connecter N_1 à S_1 et E_1 à W_1.

* sur le front descendant de $PROGN_1$, la programmation se fait réellement : A_1B_1 mémorise les valeurs stockées dans le registre à décalage A_1B_1=11.

- Etape 2 : une fois SW_1 configuré, la programmation de SW_2 peut avoir lieu car la programmation de SW_2 se fait à travers SW_1. Pour programmer SW_2, il faut envoyer [10] afin de connecter N_2 à E_2 puis [11] pour confirmer la programmation de SW_1.

* le signal $PROGN_1$ est mis au niveau haut,
* comme SW1 est déjà programmé pour connecter N_1 à S_1, le signal $PROGN_1$ se propage jusqu'à atteindre SW2. On a donc équivalence entre $PROGN_1$ et $PROGN_2$,
* la séquence [1110] est envoyée à SW_1 par le signal $DATAN_1$. Les données [10] transitent par le registre à décalage de SW_1 pour atteindre SW_2,
* sur le front descendant de $PROGN_1$ et $PROGN_2$, la programmation se fait réellement : A_1B_1 et A_2B_2 mémorisent les valeurs stockées dans leurs registres à décalage respectifs : A_1B_1=11 et A_2B_2=10.

Les chronogrammes nécessaires à la programmation sont présentés par la figure suivante.

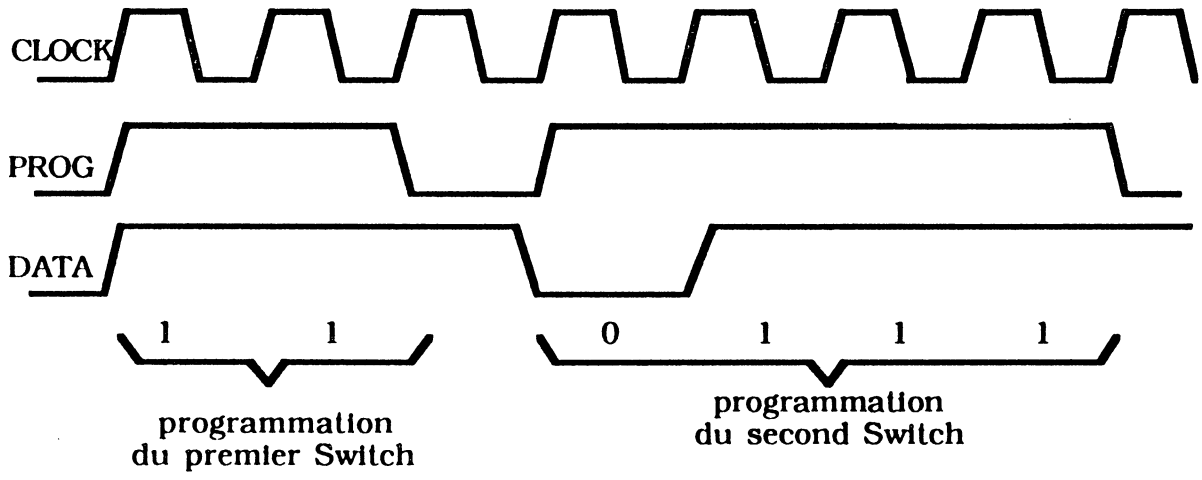


Figure VII.24 Chronogramme de programmation

Pour créer un chemin de N Switchs, une programmation en N étapes est nécessaire.

Pour créer un nouveau chemin, tous les Switchs n'appartenant qu'à ce chemin doivent être configurés, c'est à dire :

Soit S_i l'ensemble des Switchs appartenant au chemin i, pour créer le nouveau chemin j, les Switchs à programmer sont :

$$S_j - (S_j \cap (\bigcup_{i=1 \text{ à } j-1} S_i))$$

Cependant la configuration ne se réduit pas à la création de chemins : il faut aussi pouvoir les détruire.

Pour détruire un chemin, les Switchs n'appartenant qu'à ce chemin doivent être remis à zéro. Pour cela, il faut configurer avec 00 tous les Switchs appartenant à :

$$S_j - (S_j \cap (\bigcup_{\substack{i=1 \text{ à } n \\ i <> j}} S_i))$$

Nous allons illustrer ces principes de programmation sur l'exemple présenté dans la figure VII.25.

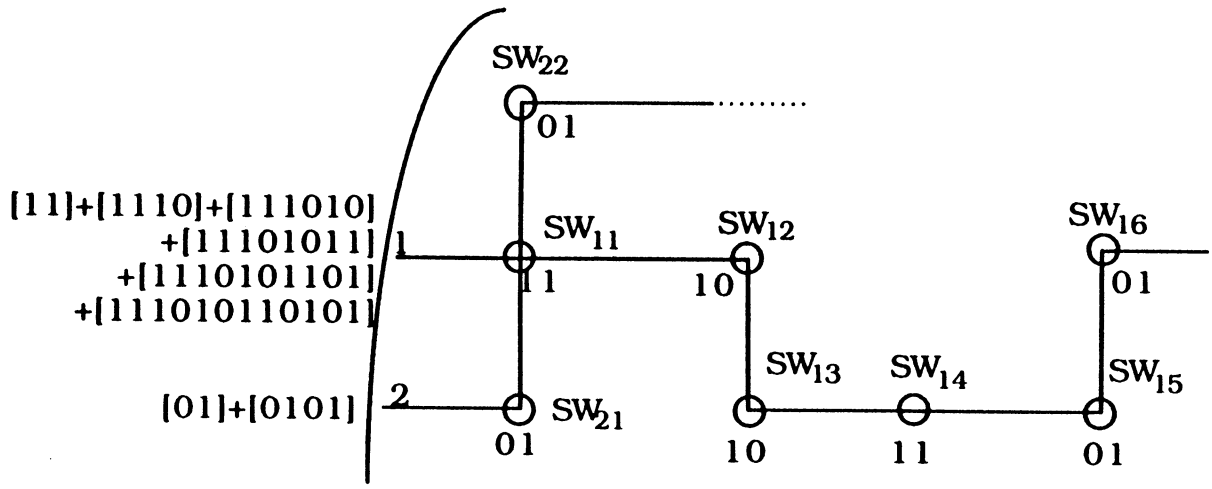


Figure VII.25 Exemples de création de chemins

Pour programmer le chemin 1 de la figure VII.25 il faut envoyer la séquence suivante :

[11]+[1110]+[111010]+[11101011]+[1110101101]+[111010110101]

Le chemin 2 est ajouté en configurant SW_{21} et SW_{22}. Il est inutile de programmer SW_1 puisque la création du chemin 1 a déjà configuré ce Switch. La programmation du chemin 2 débute par [01]+[0101] ...

Pour détruire le chemin 1, on envoie la séquence [1100000000000] qui remet à zéro les Switchs appartenant au seul chemin 1, ainsi que le montre la figure VII.26.

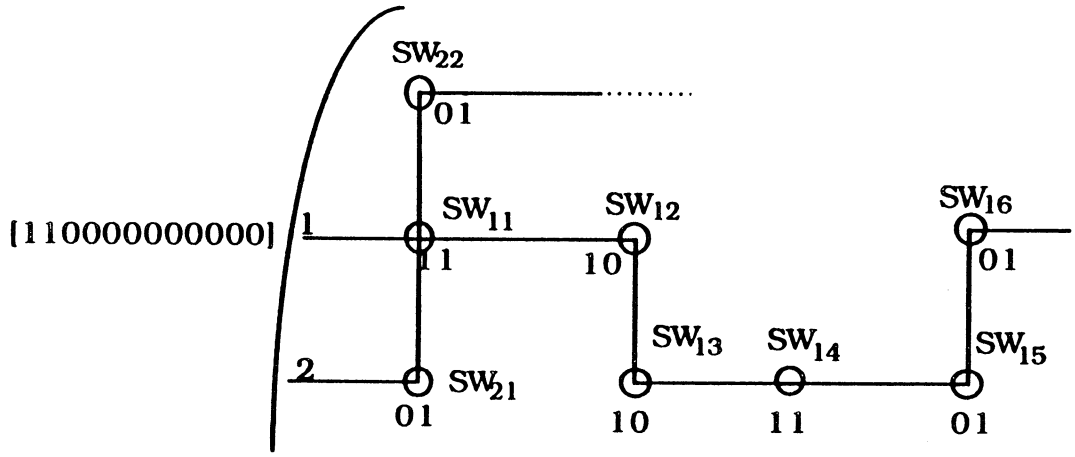


Figure VII.26 Exemple de l'annulation du chemin 1

On obtient alors la configuration suivante :

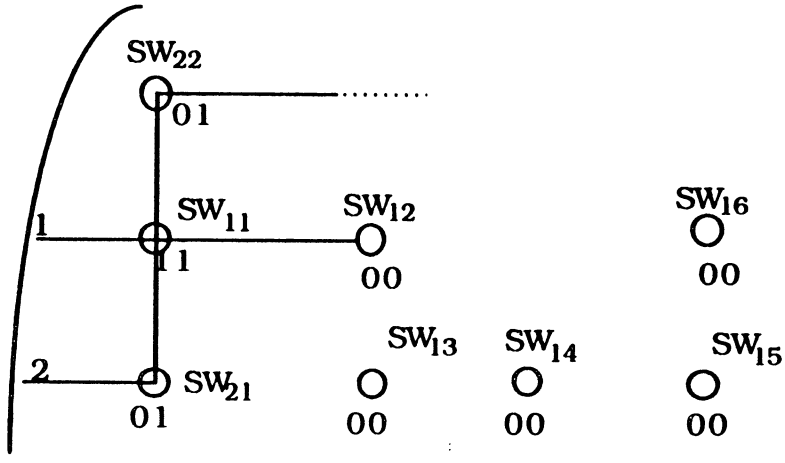


Figure VII.27 Résultat de l'annulation du chemin 1

Effacer le chemin 2 nécessite deux étapes :

- pour effacer SW_{21} et SW_{22}, il faut envoyer la séquence de programmation [0000],
- le Switch SW_{11} ne peut être effacé qu'à travers le chemin par lequel il a été programmé.

L'effacement du chemin 2 se fait alors par la programmation suivante.

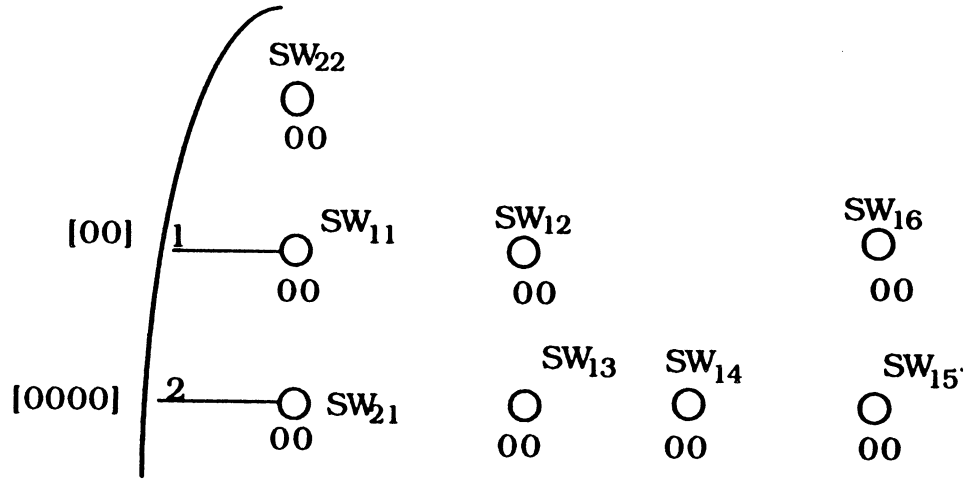
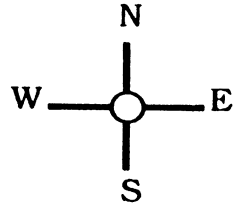


Figure VII.28 Annulation du chemin 2

III.4. Architecture du Switch

La méthode de programmation est complètement décrite. Ces spécifications de programmation ainsi que les contraintes liées à la reconfiguration permettent de présenter l'architecture du Switch.

Au début tous les Switchs sont configurés afin de déconnecter les fils N, S, E et W.



AB=00

Figure VII.29 Configuration initiale

Le signal RESET qui force cette configuration initiale doit aussi mettre à 0 tous les PROG.

Les registres A et B sont des bascules D avec Reset et les fils PROG sont mis à zéro par le dispositif nommé ACCRO.

Comme la programmation peut atteindre le Switch indifféremment par chacun des 4 côtés, le Switch doit détecter les fils qui amènent les signaux de programmation. Le dispositif SELECT sélectionne le fil PROG, le fil DATA d'où les données arrivent et celui où elles sortent.

On obtient alors le schéma logique suivant :

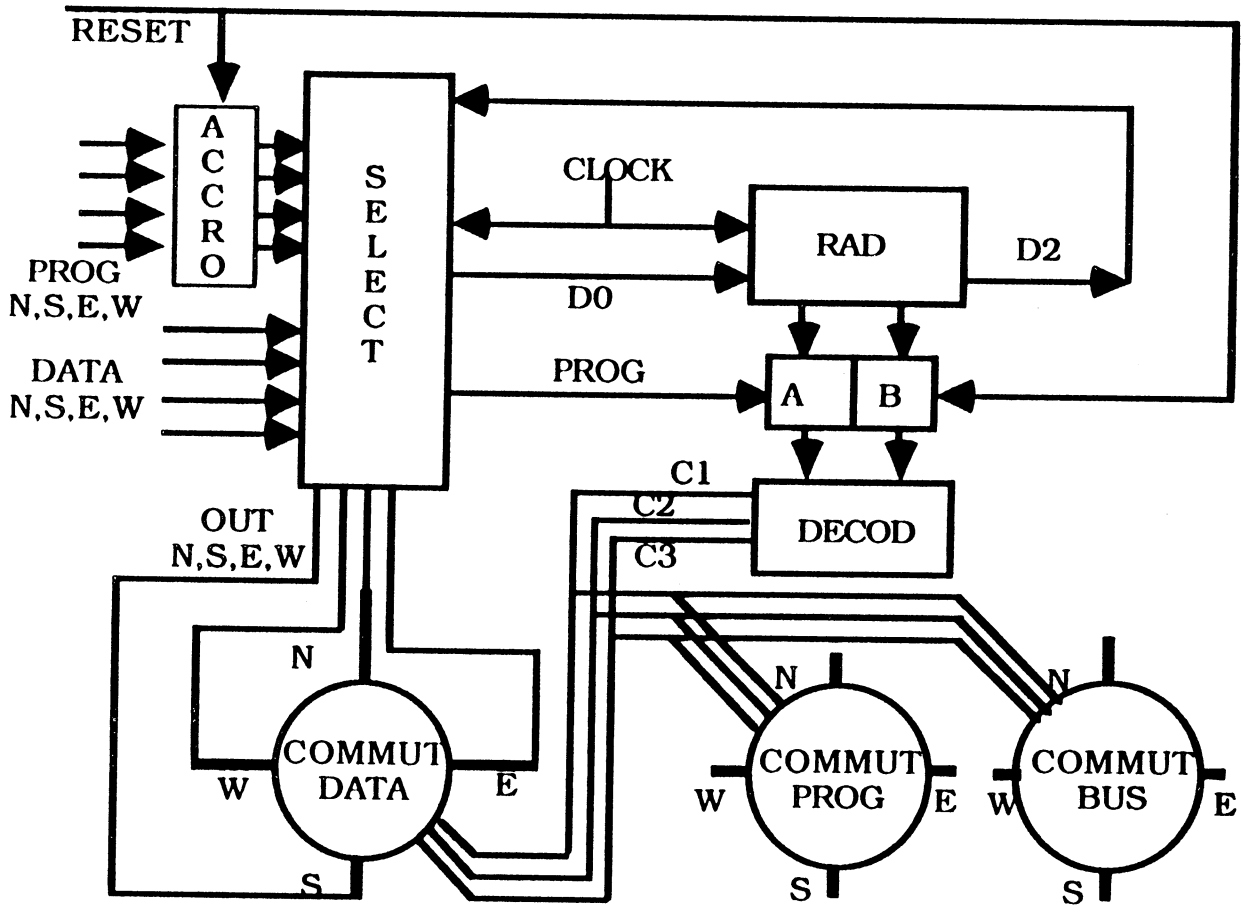


Figure VII.30 Architecture logique

Le dispositif SELECT rassemble les fils PROG qui arrivent sur chacune des extrémités du Switch : PROGN, PROGS, PROGE et PROGW. La première programmation est très importante car elle détermine le côté d'où viendront

les prochaines programmations. La première programmation est détectée quand un seul des signaux de programmation est au niveau haut :

XOR (PROGN, PROGS, PROGE, PROGW) =1.

SELECT doit mémoriser la source de la première programmation pour pouvoir sélectionner les programmations suivantes. Cette mémorisation a lieu dans 4 registres nommés FLAG. Quand XOR (PROGN, PROGS, PROGE, PROGW) vaut 1, les registres FLAG mémorisent l'état des signaux PROG. Ce chargement est synchronisé avec l'horloge afin d'éviter des mémorisations intempestives.

L'état des 4 registres FLAG permettent de choisir le fil PROG qui doit contrôler les registres AB et le fil DATA qui amène les données de programmation. Ces données sont dirigées vers le registre à décalage.

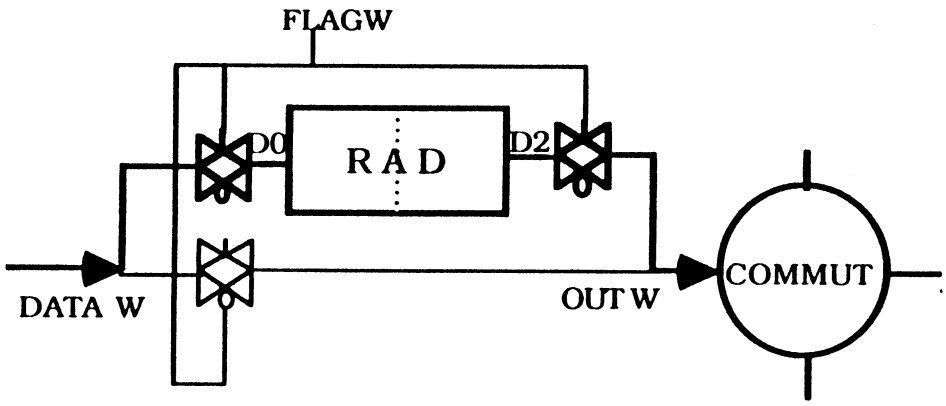


Figure VII.31 Chemin suivi par les données de programmation

Si la première programmation arrive par l'Ouest, FLAGW vaut 1. Dans ce cas, les données de programmation qui arrivent sur DATAW sont détournées vers le registre à décalage (elles suivent le chemin en gras). La sortie du registre à décalage est dirigée vers la partie Commutation afin que les données de programmation puissent traverser le Switch.

La partie Select est détaillée par la figure VII.32.

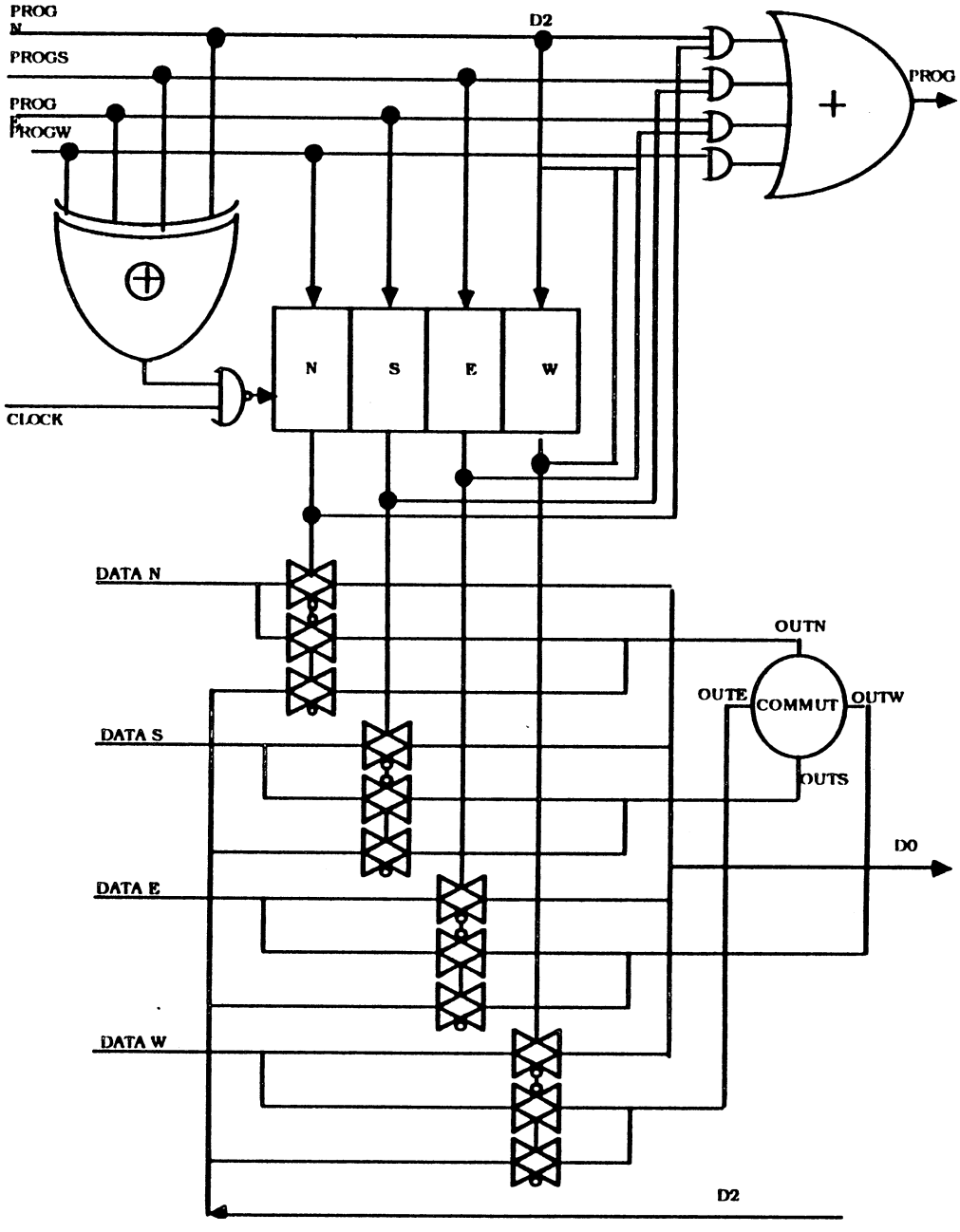


Figure VII.32 Partie SELECT

Cette partie reçoit les 4 signaux de programmation. La fonction XOR détecte la première programmation et mémorise son point d'entrée dans les registres FLAG. La fonction décrite par la figure VII.31 est réalisée par les 4 groupes de 3 portes de transfert. Le signal PROG qui contrôle le chargement des registres AB est engendré à partir des signaux PROGN,S,E,W et des registres FLAG.

III.5. Intégration du Switch

Le Switch doit être intégré dans un réseau double rail comme le montre la figure VII.33.

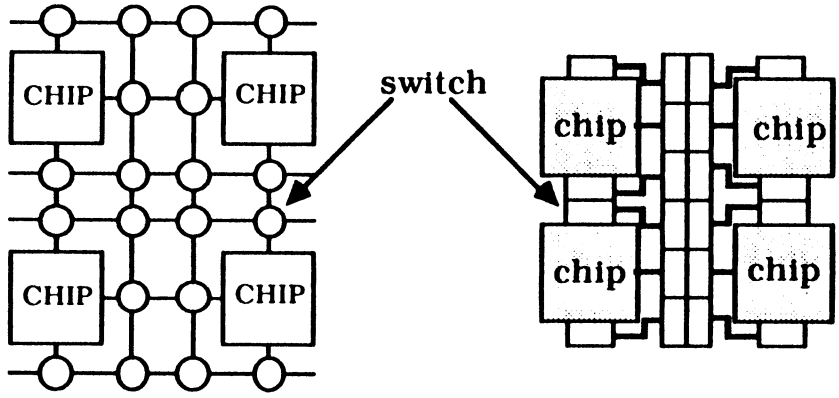


Figure VII.33 Contraintes topologiques

Aussi, le plan de masse doit être conçu en tenant compte de ces contraintes topologiques. On obtient ainsi un plan de masse rectangulaire où les différents éléments sont intégrés sous les transparences (PROG, DATA et BUS) comme le montre la figure VII.34.

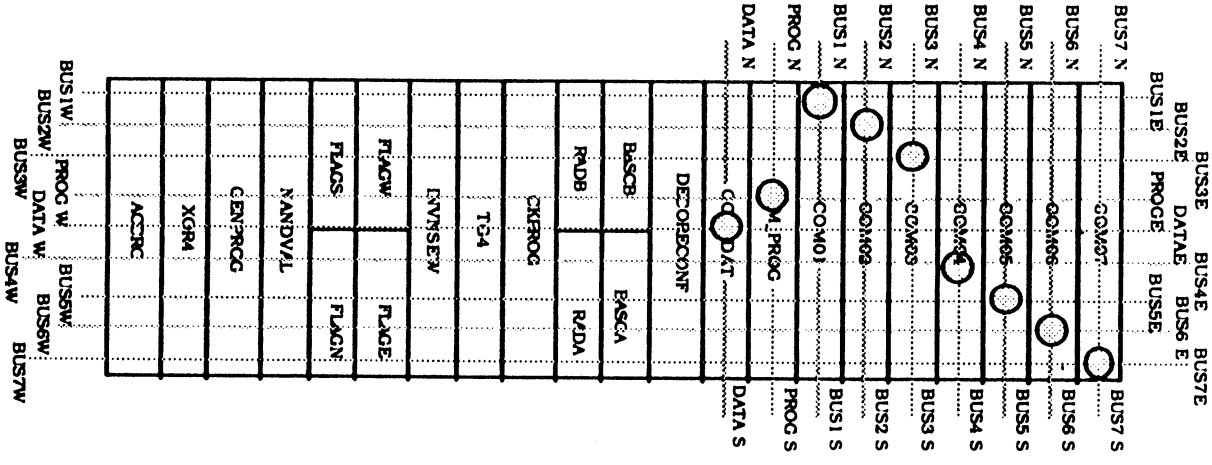


Figure VII.34 Plan de masse

Suivant ce plan de masse, une intégration a pu être réalisée. Elle a été menée de façon à ce que le Switch puisse se placer sous les bus de communication

entre chips et ainsi minimiser la place occupée par le dispositifs de commutation.

Les Switchs une fois intégrés autour des Chips doivent être programmés. Pour cela, il faut que les signaux de programmation émis sur le bord de la tranche puissent atteindre tous les Switchs. Les chips doivent donc être transparents aux signaux de programmation, c'est à dire :

- les signaux PROG et DATA des Switchs situés à l'Ouest d'un chip doivent être connectés aux signaux PROG et DATA des Switchs situés à l'Est,
- les signaux PROG et DATA des Switchs situés au Nord d'un chip doivent être connectés aux signaux PROG et DATA des Switchs situés au Sud.

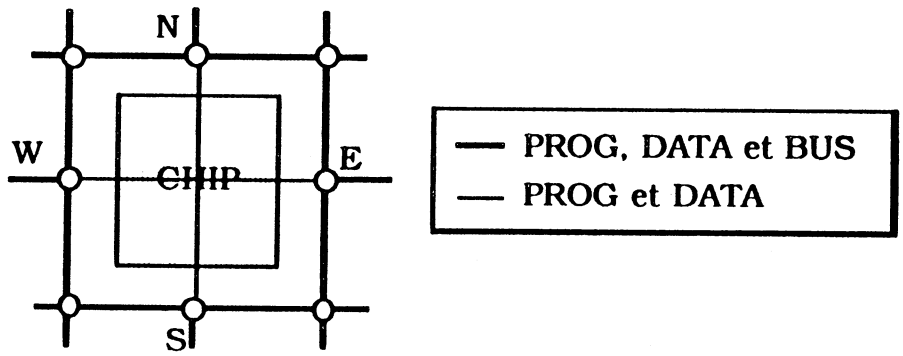


Figure VII.35 Transparence de PROG et DATA

Cet élément de commutation a été développé dans le but de minimiser les signaux de programmation. La programmation se fait en série de Switch en Switch. Des dispositifs d'amplification sont nécessaires bien qu'ils n'apparaissent pas dans cette étude. La taille de cet élément de commutation est difficile à évaluer car elle dépend du nombre de fils à commuter et de la technologie employée pour sa réalisation.

Cependant, la conception en vue de l'intégration dans un réseau de chips comme nous l'avons décrite, permet d'envisager une intégration optimale qui limite la place consacrée au réseau de reconfiguration.

IV. INTEGRATION SUR TRANCHE D'APLYSIE

Nous avons vu dans le chapitre précédent que l'intégration sur tranche de l'architecture systolique neuronique nécessitait des simplifications architecturales. Ces simplifications sont nécessaires pour conserver la tolérance aux pannes naturelle des réseaux de neurones. Elles interviennent principalement au niveau de la détection de la convergence et de la stratégie d'Entrée/Sortie.

Intégrer cette architecture simplifiée sur tranche nécessite évidemment des dispositifs de reconfiguration.
Le réseau de communication double rail que nous avons défini permet d'effectuer une reconfiguration à gros grain. Il faut alors regrouper les cellules pour constituer des chips. Chaque chip est alors immergé dans le réseau de communication double rail. La taille du chip doit être définie en fonction des données topologiques (rapport de taille entre le chip et les Switchs).
De plus, une étude plus approfondie devrait être effectuée en fonction de la technologie employée. Les données technologiques comme la densité de défauts, permettent de définir avec plus de précision la taille du grain de reconfiguration.
Enfin, selon le rendement espéré au niveau des chips, il faut éventuellement envisager une reconfiguration hiérarchisée. Chaque chip comprend alors de la redondance qui permet de remplacer localement une cellule défectueuse par une cellule redondante. Ce type de reconfiguration est tout à fait bien adapté aux réseaux systoliques rectangulaires [HUR87], [IVE87].

Pour cette étude de faisabilité, nous n'avons pas eu accès à des données technologiques. Nous avons supposé que le grain de reconfiguration est un circuit Aplysie 16, c'est à dire un sous réseau 16x16 cellules.

Nous avons basé notre étude sur la technologie 2µm d'ES2 pour évaluer la taille des différents éléments.

Un bloc de 16x16 cellules occupe une surface de 37 mm^2. Les éléments de commutation peuvent être intégrés sous les bus de communication entre chip. Comme il y a 32 fils de données par chip, les dimensions du chip sont augmentées, de chaque coté, de 200µm. On obtient alors un bloc de base de 42 mm^2.

Sur une tranche de 4 pouces, un tableau de 7x14 blocs peut être intégré comme le montre la figure VII.36. On obtient alors une tranche comportant 98 chips Aplysie 16. Parmi ces 98 chips, 64 chips seront choisis pour réaliser le réseau de 128 neurones.

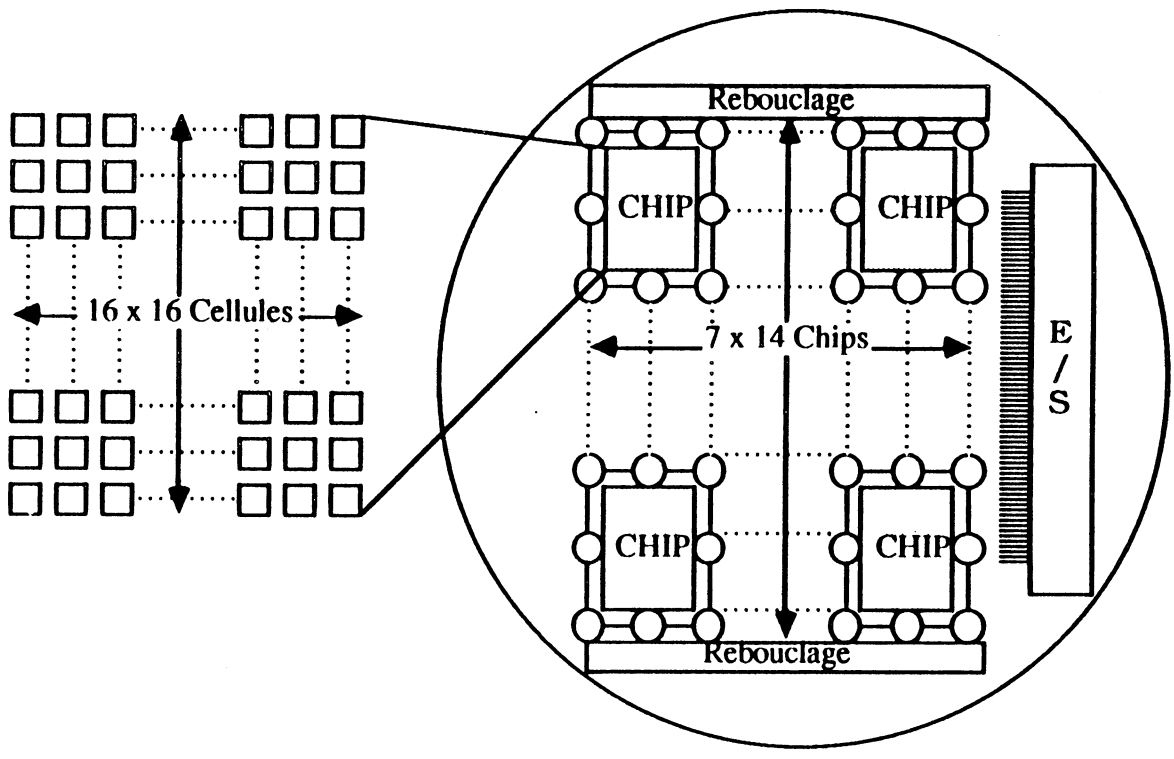


Figure VII.36 Intégration sur tranche d'Aplysie

V. CONCLUSION

Les défauts de fabrication résultant de l'intégration sur tranche, génèrent des cellules défectueuses. Afin de restaurer le fonctionnement du réseau systolique, l'immersion de la matrice cible sur les bonnes ressources de la tranche est effectuée via le réseau de communication reconfigurable.

Le réseau de communication double rail que nous avons défini précédemment, permet d'effectuer une moisson de bonne qualité sans nécessiter de trop longues interconnexions.

Cependant l'élément de commutation, base du réseau de communication, doit apporter la souplesse nécessaire à la reconfiguration. Comme les dispositifs laser et les portes à grille flottante nécessitent un environnement de programmation lourd, nous leur avons préféré les éléments de commutation à base de portes de transfert car ils sont programmables par logiciel. De plus, ils offrent des facilités de test [JAY86]. Leur souplesse de programmation permet aussi d'envisager une reconfiguration pendant la vie du circuit pour éviter les pannes éventuelles.

Conclusion

Lorsqu'on aborde la définition et la réalisation de machines neuroniques, il faut distinguer plusieurs approches. Les logiciels de simulation de réseaux de neurones sont extrêmement souples. Ils peuvent suivre aisément l'évolution des théories connexionistes, tant au niveau des modèles de neurones que des lois d'apprentissage. Cependant leur implantation sur des machines séquentielles est très lente et le traitement d'applications réalistes est rapidement impossible. Aussi, des cartes accélératrices peuvent être ajoutées pour réduire les temps de calcul prohibitifs. La réalisation de matériel spécialisé va encore plus loin avec la construction d'architectures orientées vers les réseaux de neurones. Les approches les plus courantes sont les architectures systoliques et parallèles qui offrent de grandes performances. Ces machines sont suffisamment générales pour pouvoir apporter la souplesse nécessaire à l'émulation de divers réseaux de neurones. Mais, ces nouvelles machines souffrent encore d'un environnement logiciel restreint et leur convivialité reste très limitée.

L'approche VLSI se démarque assez nettement de ces approches car un circuit ne peut apporter autant de souplesse et d'adaptabilité que des simulations. Les circuits VLSI peuvent être développés suivant plusieurs axes.

Si l'on veut conserver un aspect très général aux circuits neuroniques, on est amené à ne s'intéresser qu'aux calculs matriciels et principalement aux produits matrice-vecteur. Ces spécifications conduisent à réaliser des micro-processeurs du type traitement du signal. Les circuits existants peuvent alors remplir ce rôle.

L'intégration de circuits peut aussi être nécessaire dans le cadre de la réalisation d'une machine orientée vers la simulation des réseaux de neurones. On peut alors réaliser des circuits spécifiques qui auront un rôle d'accélérateur (par exemple des circuits implantant une fonction de décision complexe

comme la fonction sigmoïde). Ces circuits, une fois intégrés dans l'architecture finale permettent d'obtenir une machine plus performante.

La dernière approche fait appel à l'intégration "pure et dure". Il s'agit de réaliser un circuit VLSI, le plus autonome possible. Il peut alors servir de base à des applications pratiques et concrètes. Un circuit ne peut apporter la même souplesse qu'un logiciel ou qu'une architecture dédiée. Il faut alors restreindre l'étude à un modèle. Il faut choisir avec soin son domaine d'étude car le coût de l'intégration est élevé et l'algorithme, une fois implanté, n'est plus modifiable.

Parmi la multitude de modèles de neurones, de réseaux et de lois d'apprentissage, nous avons choisi d'intégrer un réseau de Hopfield pour la maturité de sa formalisation et son vaste champ d'application. Mettre sur le silicium une loi d'apprentissage réduisait encore la généralité du circuit. Nous avons préféré intégrer uniquement la phase de reconnaissance, tout en conservant la possibilité de charger la matrice synaptique. On obtient alors un circuit à matrice programmable qui permet d'effectuer la phase de reconnaissance avec un maximum de performance.

L'inconvénient majeur du réseau de Hopfield réside dans l'interconnexion complète des neurones. Nous avons résolu ce problème en systolisant les connexions ; les neurones sont fonctionnellement complètement interconnectés, mais les connexions physiques sont locales. Cette systolisation a été décrite sous la forme d'un système d'équations récurrentes uniformes et son adéquation à l'algorithme de reconnaissance a pu être formellement prouvée. Chaque cellule du circuit systolique effectue un calcul synaptique et la circulation adéquate des données permet d'enchaîner les pas de récurrence et de détecter la convergence de la phase de reconnaissance. La fonction de décision, simple pour le modèle de réseau choisi, est intégrée au circuit. Un dispositif d'Entrée/Sortie permet soit d'appliquer cette fonction et de reboucler les données pour poursuivre le calcul récurrent, soit d'effectuer une

Entrée/Sortie si la reconnaissance a convergé. Les performances du circuit sont encore améliorées par le traitement en mode "*pipeline*" de 2N vecteurs dans le réseau, sans aucune augmentation de la complexité.

Cette architecture systolique récurrente intègre donc toutes les fonctions nécessaires à la phase de reconnaissance : la fonction de décision, la détection de convergence et un dispositif d'Entrée/Sortie autonome. La matrice synaptique est programmable afin de conserver un maximum de généralité au circuit.

Nous avons commencé par intégrer une seule cellule synaptique pour valider l'architecture de la cellule. Ce circuit Aplysie 1, de 0,18 mm^2, a été testé avec succès jusqu'à 25 MHz.

Nous avons poursuivi l'intégration avec le circuit Aplysie 16 qui comprend un tableau carré de 256 cellules, soit l'équivalent de 16 neurones. Afin de limiter le nombre de plots qui est une contrainte importante quand on intègre une architecture systolique, nous avons procédé au multiplexage des plots d'Entrée/Sortie. Afin de conserver la régularité de l'architecture, chaque circuit intègre un dispositif de détection de convergence et un dispositif d'Entrée/Sortie. On obtient ainsi un circuit cascadable qui permet la construction de plus grands réseaux par interconnexion de plusieurs circuits Aplysie 16.

Un circuit Aplysie 16 permet d'effectuer la phase de reconnaissance sur une matrice synaptique 16x16 avec une performance de plus de 2 millions de produits matrice-vecteur par seconde, soit 0,6 milliard de calculs synaptiques par seconde.

Avec une implantation directe d'un réseau de 128 neurones, la phase de reconnaissance sur une matrice synaptique 128x128 s'effectue au rythme de 2 millions de produits matrice-vecteur par seconde, soit 19 milliards de calculs synaptiques par seconde.

Malgré la compacité du dessin du circuit de base Aplysie 16, la réalisation de réseaux de grande taille est assez coûteuse en terme de surface car le nombre de cellules croît avec le carré du nombre de neurones. Afin de réduire les connexions entre circuits, de limiter les problèmes de consommation et de place, nous avons envisagé l'intégration sur tranche entière (WSI). La grande répétitivité de l'architecture définie et la tolérance naturelle aux pannes due à la représentation distribuée de l'information font des réseaux de neurones de très bons candidats à l'intégration sur tranche entière.

Cependant, l'intégration à une telle échelle ne va pas sans problème. Pour conserver la tolérance aux pannes, des simplifications architecturales ont dû être apportées. De plus, les défauts de fin de fabrication sont inévitables et les éléments défectueux doivent être contournés afin de restaurer le fonctionnement global du circuit. Des algorithmes de reconfiguration sont alors mis en œuvre pour utiliser au maximum les bonnes ressources de la tranche. Ces algorithmes doivent optimiser le placement de l'architecture cible sur la tranche hôte et le routage des interconnexions. La reconfiguration ne pourra se faire avec succès sans une bonne adéquation entre l'algorithme de reconfiguration et l'architecture à reconfigurer.

Mais, il faut aussi que les dispositifs intégrés sur la tranche permettent de réaliser physiquement la reconfiguration. Pour cela, les cellules doivent être intégrées dans un réseau de communication reconfigurable. Ce sont les éléments de commutation qui permettent de modifier les interconnexions des cellules et ainsi relier les bonnes ressources de la tranche selon l'architecture ciblée.

Afin de satisfaire ces contraintes liées à la reconfigurabilité de l'architecture, nous avons été amenés à étudier différents réseaux de commutation et des éléments de commutation. Notre choix s'est porté vers un réseau de commutation classique : le réseau double rail, car il satisfait ces contraintes. Les dispositifs de commutation peuvent être de différente nature. Les

dispositifs matériels comme les fusibles laser ou les portes à grille flottante ne sont pas satisfaisants car leur environnement de programmation est lourd.
Aussi, nous avons été conduits à définir un dispositif de commutation "soft", programmable par logiciel, appelé Switch. Tous les Switchs ne peuvent pas être directement reliés au bord de la tranche, pour recevoir leurs données de programmation. La programmation fait alors appel à la notion de chemin : un chemin de Switchs est créé pas à pas, en ajoutant successivement les Switchs du chemin ; les données de programmation sont communiquées au Switch à ajouter via le début du chemin déjà construit.
Cette méthode de programmation a conduit à la définition architecturale complète du dispositif de commutation. Son intégration a été effectuée en vue de son immersion dans un réseau systolique carré ou rectangulaire.

L'ensemble de ce travail a abouti à la réalisation d'un circuit spécifique dédié à la phase de reconnaissance pour les réseaux de Hopfield et à la spécification d'un dispositif de reconfiguration. L'élément de commutation défini permet d'envisager l'intégration sur tranche entière de cette architecture et ainsi d'obtenir un réseau de neurones compact et très performant.

L'intégration de circuits intégrés VLSI basés sur des modèles connexionistes est généralement sévèrement limitée par l'implantation des connexions. L'approche systolique récurrente que nous avons adoptée, permet de résoudre ce problème crucial. Le travail effectué a montré que l'implantation VLSI d'un modèle connexioniste est viable en technologie digitale.
Dans le cadre défini en début d'étude - circuit spécialisé pour la phase de reconnaissance dans un réseau de Hopfield - la solution proposée présente de nombreux avantages : enchaînement des pas de récurrence, détection de la convergence, extensibilité, traitement pipeline, performances élevées.

Dans un proche avenir, le travail présenté ici pourra se poursuivre par la construction d'un système à base d'Aplysies pour une application spécialisée dans un des domaines d'application des réseaux de Hopfield. Toutefois, des études futures devront élargir le cadre de l'étude en intégrant l'apprentissage ou en explorant les possibilités d'intégration de nouveaux modèles.

Annexe 1 : Calcul des performances

Nous supposons que la fréquence de l'horloge de base est de 20MHz soit HC2 = 5.10^{-8} s.

I. 16 NEURONES SUR APLYSIE16.

I.1. Chargement normal sur 8 bits

Pour une matrice 16x16, le registre SP ne contient que 8 bits significatifs. On a donc : LC1 = 9 HC2.

Il faut 16 cycles d'horloge lente pour chaque bit + 16 pour le début:

T_{ch} = 9x16x9x5.10^{-8} = 64.10^{-6}s = **64 µs**.

I.2. Chargement optimisé sur 8 bits

Pour le multiplexage vertical, il faut au moins 4 cycles d'horloge rapide dans un cycle d'horloge lente. On a donc :

LC1 = 4HC2 sauf pour charger le D dans le registre C où LC1=9HC2 pour chaque bit.

Il faut 15 cycles d'horloge LC1=4HC2 plus 1 LC1=9HC2 pour chaque bit:

T_{ch} = 8x(15*4+9)x5.10^{-8} = 27.10^{6}s = **27µs**.

I.3. Reconnaissance

Pour une matrice 16x16, le registre SP ne contient que 8 bits significatifs. On a donc : LC1 = 9 HC2.

En mode stationnaire, à chaque LC1, 16 composantes sont évalués : un vecteur par LC1.

1 vecteur toutes les 45.10^{-8} s soit $2,22.10^{6}$ vec/s.

Nbvec = $2,22.10^{6}$ vec/s

II. 128 NEURONES SUR LE RESEAU COMPLET 128X128.

II.1. Chargement normal sur 8 bits

Pour une matrice 128x128, le registre SP ne contient que 16 bits significatifs. On a donc : LC1 = 17 HC2.

Il faut 128 cycles d'horloge lente pour chaque bit + 128 au début :

T_{ch} = 9x128x17x5.10^{-8} = 0,97.10^{-3}s = **1 ms**.

II.2. Chargement optimisé sur 8 bits

Pour le multiplexage vertical, il faut au moins 4 cycles d'horloge rapide dans un cycle d'horloge lente. On a donc : LC1 = 4HC2 sauf pour charger le D dans le registre C où LC1=17HC2 pour chaque bit.

Il faut 127 cycles d'horloge rapide plus 1 lent :

T_{ch} = 8x(127*4+17)x5.10^{-8} =210.10^{-6}s = **0,2 ms**.

II.3. Reconnaissance

Pour une matrice 128x128, le registre SP ne contient que 16 bits significatifs. On a donc : LC1 = 17 HC2.

En mode stationnaire, à chaque LC1, 128 composantes sont évalués : un vecteur par LC1.

1 vecteur toutes les 85.10^{-8} s soit 1,18.10^{6} vec/s.

Nbvec = 1,18.10^{6} vec/s

III. 128 NEURONES SUR UN CIRCUIT APLYSIE 16

Le calcul se fait par sous-matrice 16x16 de la matrice 128x128. Le calcul s'effectue donc en 64 pas composés de deux étapes :

- chargement de la sous-matrice
- calcul de X vecteurs.

Ces deux étapes vont se succéder suivant la nombre de vecteur à reconnaître (X).

III.1. Chargement normal sur 8 bits

Pour une matrice 128x128, le registre SP ne contient que 16 bits significatifs.

On a donc : LC1 = 17 HC2.

Il faut 16 cycles d'horloge lente pour chaque bit :

$T_{ch} = 8x16x17x5.10^{-8} = 108.10^{-6}s = 108$ ms.

III.2. Reconnaissance

pour 1 vecteur : 2x16xLC1

T_{rec} (1) = $2x16x17x5.10^{-8} = 2720.10^{-8} = 27\mu s$

pour X vecteur : (2x16+X-1).LC1

T_{rec} (X) = $(2x16+X-1)x17x5.10^{-8} = 26,35 + 0,85X\ \mu s$

$T_{rec}(1) = 27\mu s$

$T_{rec}(32) = 53\ \mu s$

$T_{rec}(100) = 111\ \mu s$

$T_{rec}(256) = 244\ \mu s$

III.3. Temps total de traitement

Le temps total $T_{tot}(X)$ pour une reconnaissance par lot de X vecteur se calcul par :

$T_{tot}(X) = 64(T_{ch} + T_{rec}(X))$

Le nombre de vecteur reconnus par lot de X vecteurs s'obtient par :

$NB(X) = \frac{X}{T_{tot}(X)}$

Application numérique :

$T_{tot}(1) = 64(T_{ch} + T_{rec}(1)) = 64 \times 135\mu s = 8,6$ ms

Nbvec (1) = $\frac{1}{T_{tot}(1)}$ = 116 vec/s

$T_{tot}(32) = 64(T_{ch} + T_{rec}(32)) = 64 \times 161\mu s = 10$ ms

Nbvec (32) = $\frac{32}{T_{tot}(32)}$ = 3200 vec/s

$T_{tot}(100) = 64(T_{ch} + T_{rec}(100)) = 64 \times 220\ \mu s = 14$ ms

Nbvec (100) = $\frac{100}{T_{tot}(100)}$ = 7124 vec/s

$T_{tot}(256) = 64(T_{ch} + T_{rec}(256)) = 64 \times 352\ \mu s = 22,5$ ms

Nbvec (256) = $\frac{256}{T_{tot}(256)}$ = 11 377 vec/s

IV. 1024 NEURONES SUR LE RESEAU COMPLET 1024X1024 SYNAPSES

Le réseau de 1024 neurones (1KN) est directement implanté sur un ensemble de 64x64 circuits Aplysie16.

IV.1. Chargement normal sur 8 bits

Pour une matrice 128x128, le registre SP ne contient que 16 bits significatifs. On a donc : LC1 = 17 HC2.

Il faut 1024 cycles d'horloge lente pour chaque bit (8) + 1024 au début :

$T_{ch} = 9x1024x17x5.10^{-8} = 7,8.10^{-3}s$ = **7,8 ms**.

II.2. Chargement optimisé sur 8 bits

Pour le multiplexage vertical, il faut au moins 4 cycles d'horloge rapide dans un cycle d'horloge lente. On a donc : LC1 = 4HC2 sauf pour charger le D dans le registre C où LC1=17HC2 pour chaque bit.

Il faut 1023 cycles d'horloge rapide plus 1 lent :

T_{ch} = 8x(1023*4+17)x5.10^{-8} =210.10^{-6}s = **1,6 ms**.

II.3. Reconnaissance

Pour une matrice 128x128, le registre SP ne contient que 16 bits significatifs.

On a donc : LC1 = 17 HC2.

En mode stationnaire, à chaque LC1, 1024 composantes sont évalués : un vecteur par LC1.

1 vecteur toutes les 85.10^{-8} s soit 1,18.10^{6} vec/s.

Nbvec = 1,18.10^{6} vec/s

V. 1024 NEURONES SUR LE RESEAU COMPLET 128X128 SYNAPSES

1KN = 1024 neurones.

64 sous matrices

T_{ch}= 1ms

pour X vecteur : (2.128+X-1).LC1

T_{rec} (X) = (2.128+X-1)x17x5.10^{-8} = 216 + 0,85X µs

T_{rec} (1) = 217 µs

T_{tot} (1) = 64(1000+217) µs = 77 ms

Nbvec (1) = 13 vec/s

T_{rec} (256) = 433 µs

T_{tot} (256) = 64(1000+433) µs = 91 ms

Nbvec (256) = 2813 vec/s

T_{rec} (1024) = 1086 µs

T_{tot} (1024) = 64(1000+1086) µs = 133 ms

Nbvec (1024) = 7700 vec/s

T_{rec} (2048) = 1956 µs

T_{tot} (2048) = 64(1000+1956) µs = 189 ms

Nbvec (2048) = 10 835 vec/s

Annexe 2 : Réalisation d'un réseau 128x128

Un réseau 128x128 est réalisé par l'interconnexion de 64 circuits Aplysie 16, organisés en matrice 8x8.

Afin de réaliser un réseau fonctionnellement correct, il faut programmer les différents circuits en fonction de leur localisation dans le réseau. La figure suivante distingue 10 localisations pour lesquelles les circuits doivent être différenciés.

NORD

OUEST								ESTT
	6	3	3	3	3	3	3	7
	2	1	0	0	0	0	0	4
	2	0	1	0	0	0	0	4
	2	0	0	1	0	0	0	4
	2	0	0	0	1	0	0	4
	2	0	0	0	0	1	0	4
	2	0	0	0	0	0	1	4
	9	5	5	5	5	5	5	8

SUD

Figure Localisations des circuits dans le réseau final

Nous définissons ci-dessous les 10 types de localisation. Tous les circuits sont identiques, mais une programmation adéquate permet de spécifier le comportement des circuits en fonction de leur localisation.

- <u>type 0</u> : autre localisation.

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = "0".

- <u>type 1</u> : sur la diagonale (i=j),

  DIAG = "1" pour programmer les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = "0".

- <u>type 2</u> : sur le bord Ouest (j=1),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = "0" initialisation de la somme partielle,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin(i,j) = CVout(i-1,j) propagation de la détection de la convergence.

- <u>type 3</u> : sur le bord Nord (i=1),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sN(i,j) rebouclage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = "0".

- type 4 : sur le bord Est (j=N),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = interface,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = IntroOut(i-1,j) propagation de la "fermeture Eclair",

  CVin = "0".

- type 5 : sur le bord Sud (i=N),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sS(i,j) rebouclage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = "0".

- type 6 : angle Ouest-Nord (i=1, j=1),

  DIAG = "1" pour programmer les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sN(i,j) rebouclage,

  eW(i,j) = "0" initialisation de la somme partielle,

  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = "0" initialisation de la détection de convergence.

- type 7 : angle Nord-Est (i=1, j=N),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sN(i+1,j) connexion par voisinage,

  eE(i,j) = interface,

  eN(i,j) = sN(i,j) rebouclage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn à initialiser selon la gestion d'Entrée/Sortie,

  CVin = "0".

- type 8 : angle Est-Sud (i=N, j=N),

  DIAG = "1" pour programmer les cellules diagonales,

  eS(i,j) = sS(i,j) rebouclage,

  eE(i,j) = interface,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = sE(i,j-1) connexion par voisinage,

  IntroIn = IntroOut(i-1,j) propagation de la "fermeture Eclair",

  CVin = "0".

- type 9 : angle Sud-Est (i=N, j=1),

  DIAG = "0" pour inhiber les cellules diagonales,

  eS(i,j) = sS(i,j) rebouclage,

  eE(i,j) = sW(i,j+1) connexion par voisinage,

  eN(i,j) = sS(i-1,j) connexion par voisinage,

  eW(i,j) = "0" initialisation de la somme partielle,

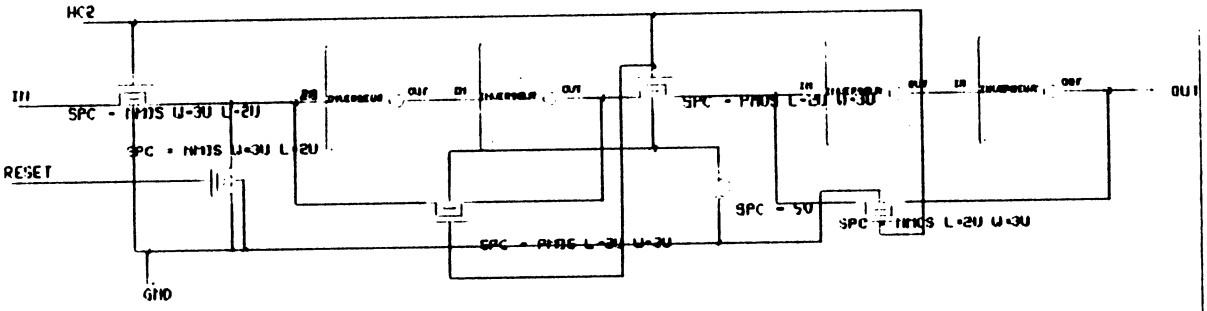
  IntroIn = "1" inhibition du rebouclage de la "fermeture Eclair",

  CVin = CVout(i.,j) propagation de la détection de convergence.

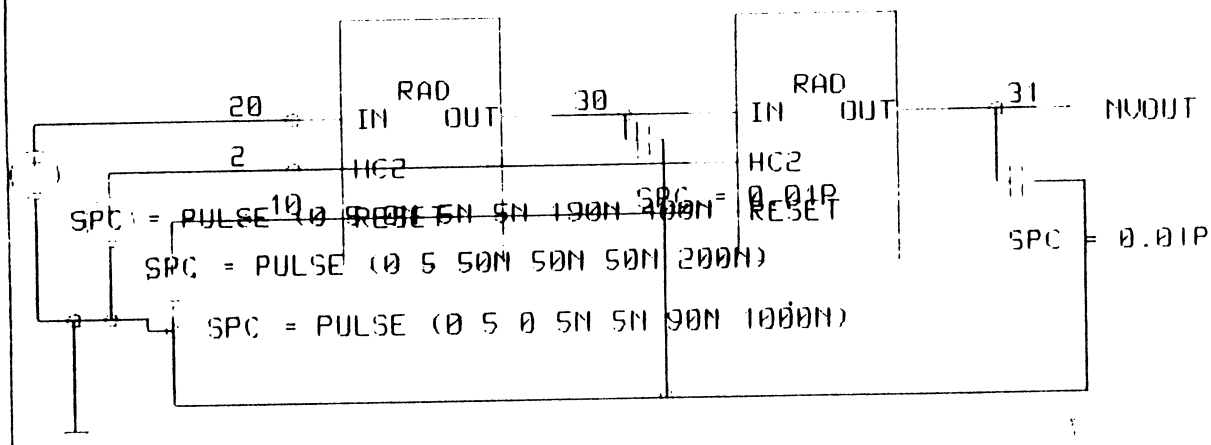
Les 10 localisations sont facilement différentiées grâce à la programmation adéquate de la fermeture Eclair, du dispositif de détection de la convergence et du signal DIAG.

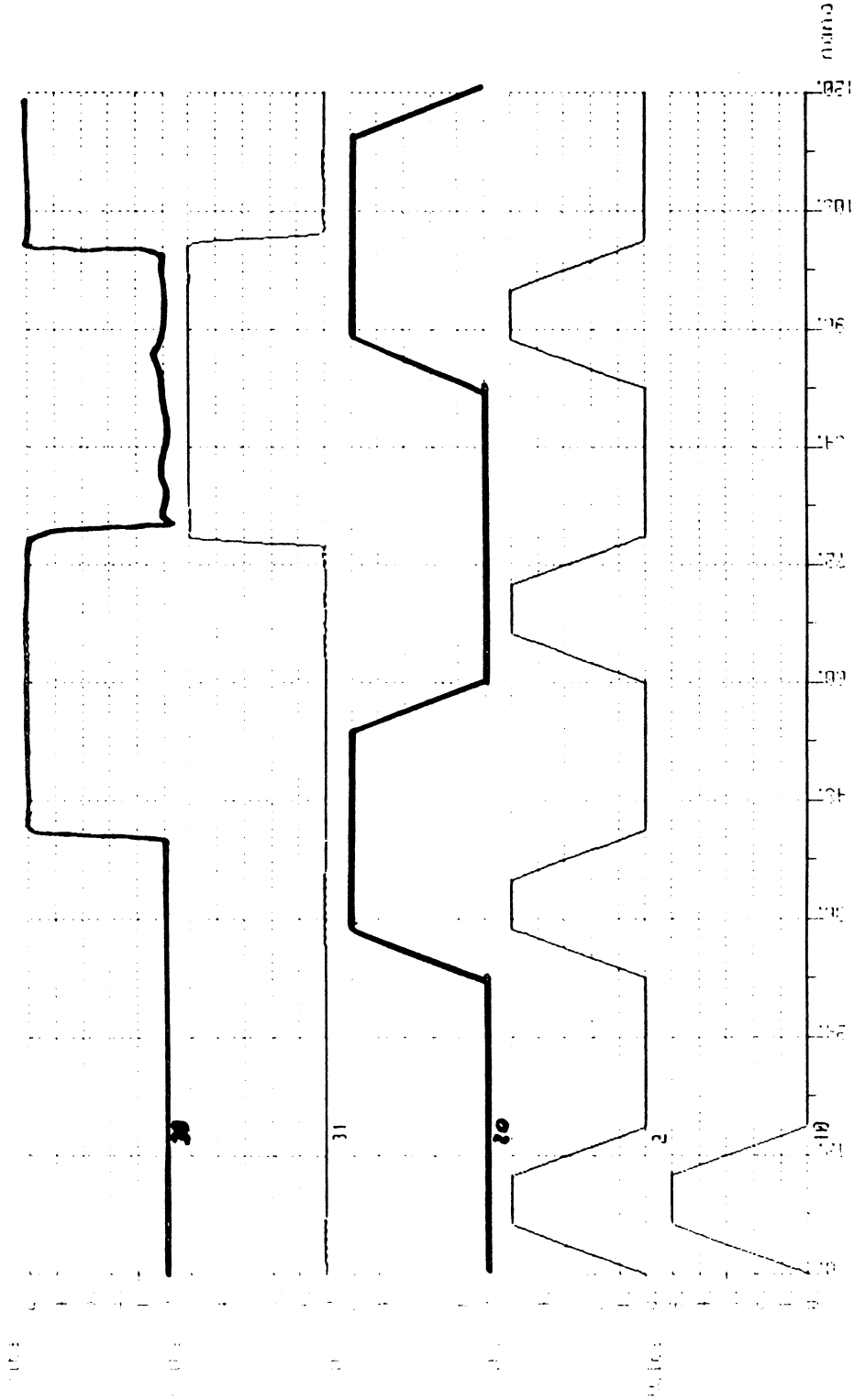
Annexe 3 : Implantation d'Aplysie

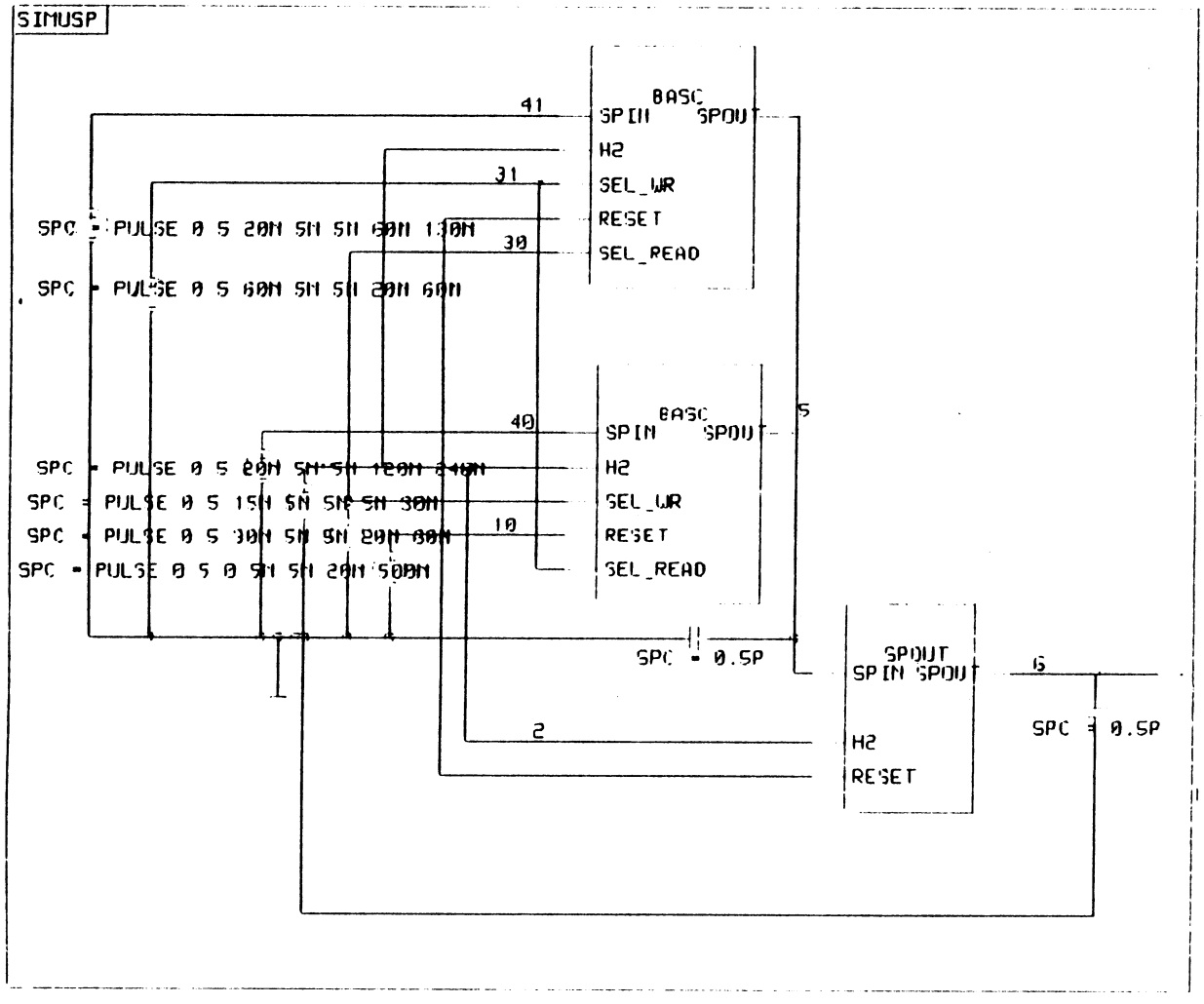
RAD

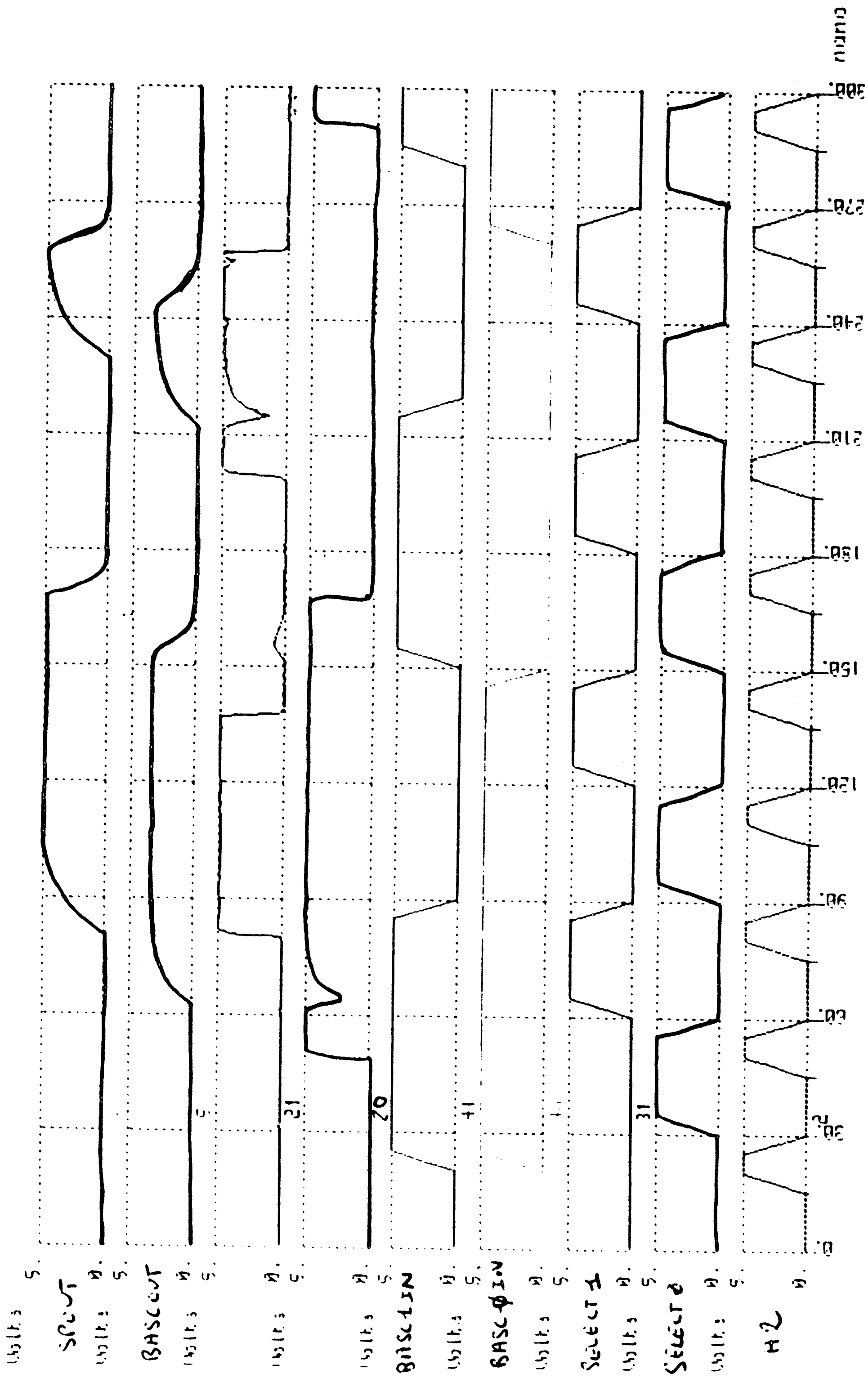

HC2
IN
SPC = NMOS L=2U
SPC = NMOS L=3U L=2U
RESET
GND
SPC = 5V
OUT

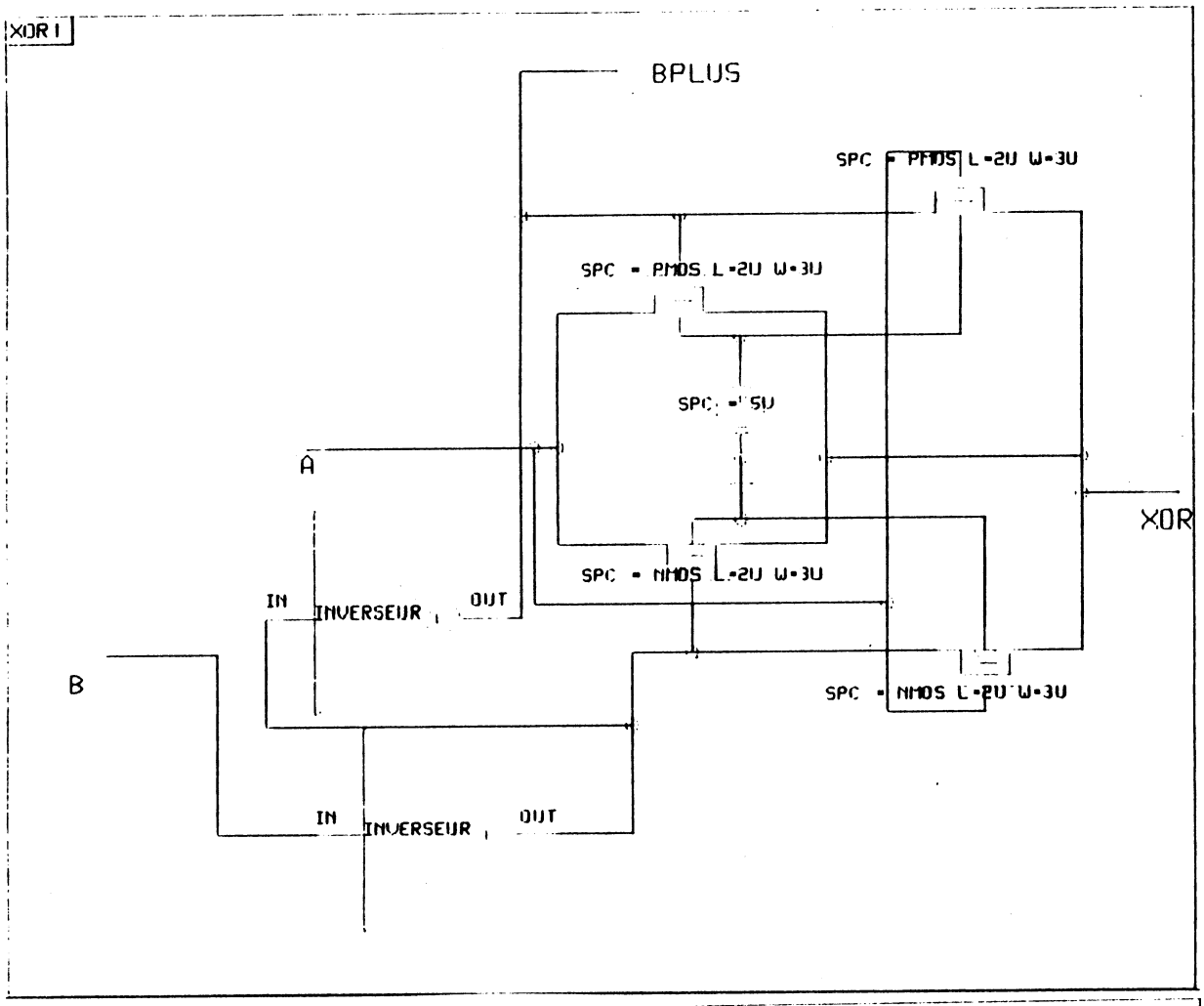
SIMURAD

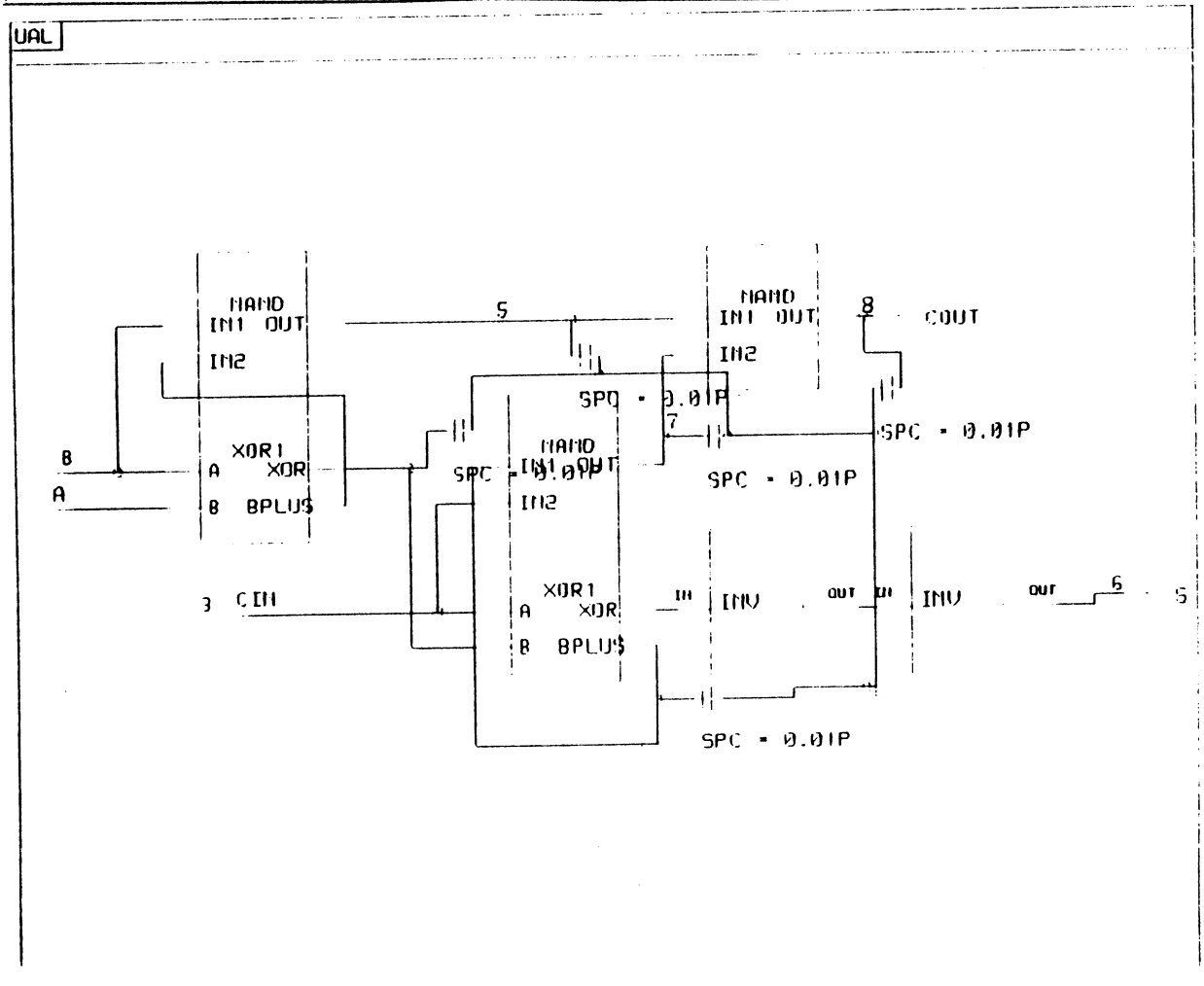

20
2
10
RAD
IN OUT
30
RAD
IN OUT
31
NVOUT
HC2
RESET
SPC = 0.01P
SPC = PULSE (0 5 50N 50N 50N 200N)
SPC = PULSE (0 5 0 5N 5N 90N 1000N)

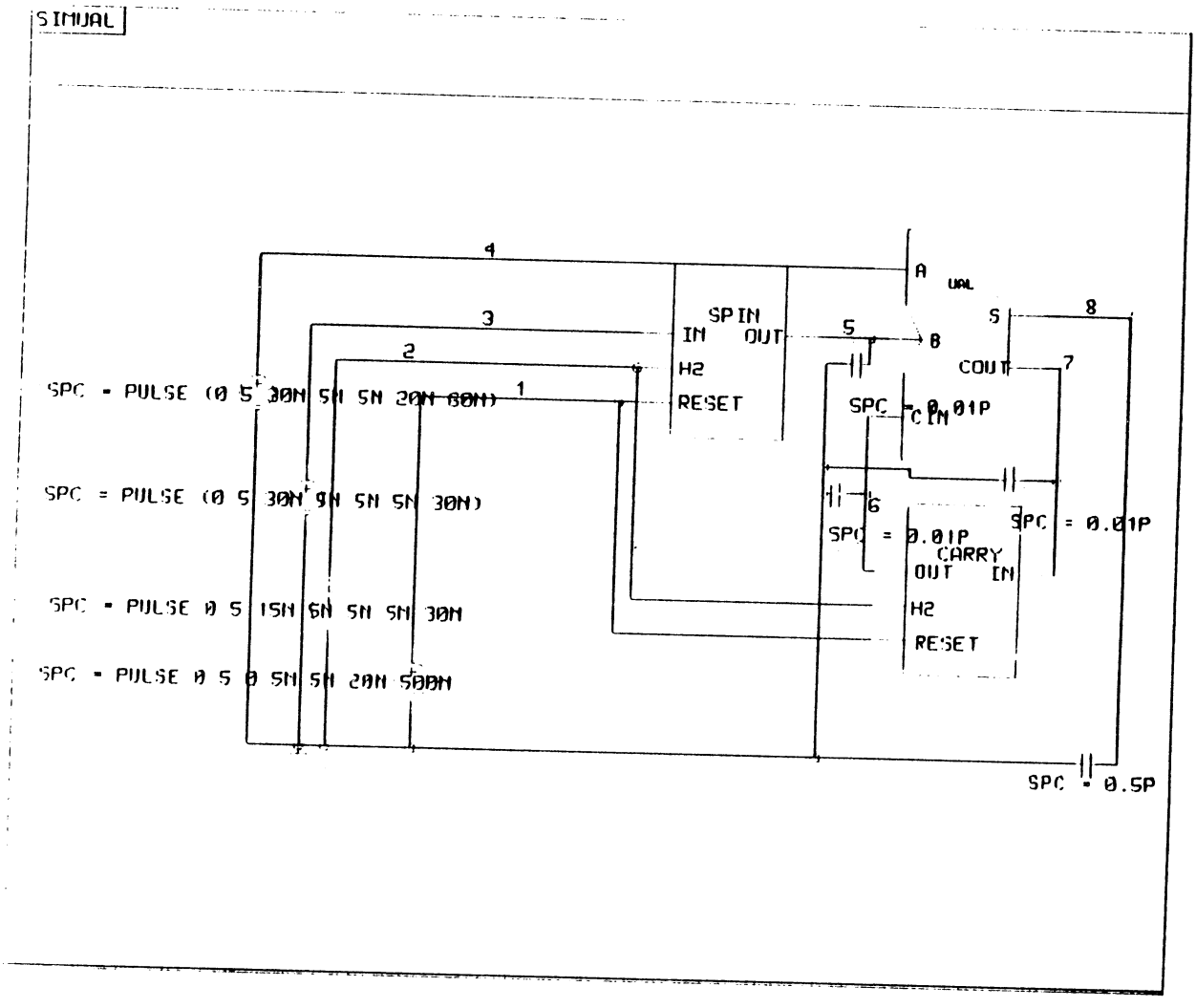


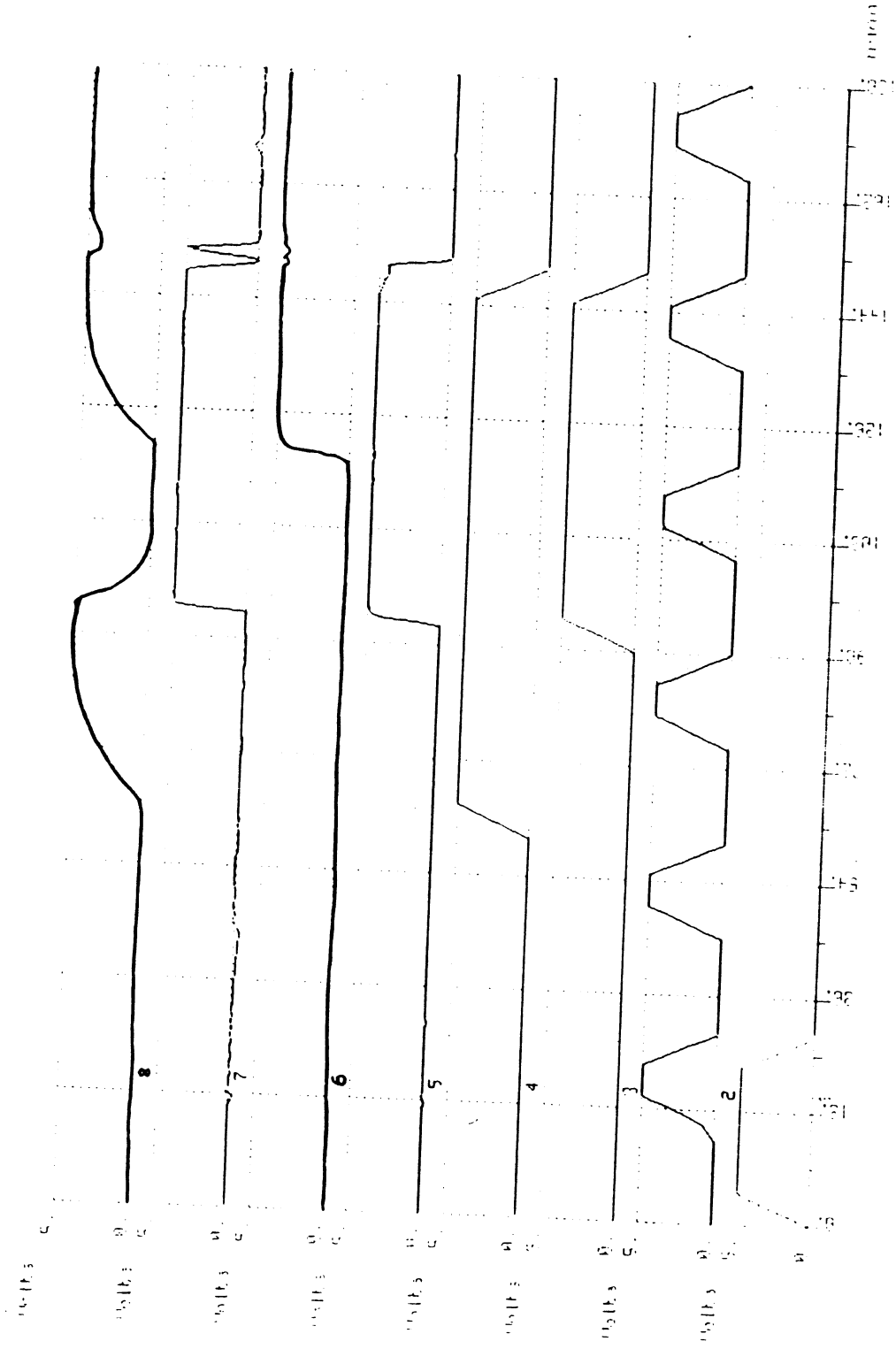

SIMUSP
BASC
SPIN
SPOUT
H2
SEL_WR
RESET
SEL_READ
41
31
30
40
10
2
5
6
SPC = PULSE 0 5 20N 5N 5N 60N 130N
SPC = PULSE 0 5 60N 5N 5N 20N 60N
SPC = PULSE 0 5 20N 5N 5N 120N 240N
SPC = PULSE 0 5 15N 5N 5N 5N 30N
SPC = PULSE 0 5 30N 5N 5N 20N 80N
SPC = PULSE 0 5 0 5N 5N 20N 50BN
SPC = 0.5P
SPC = 0.5P

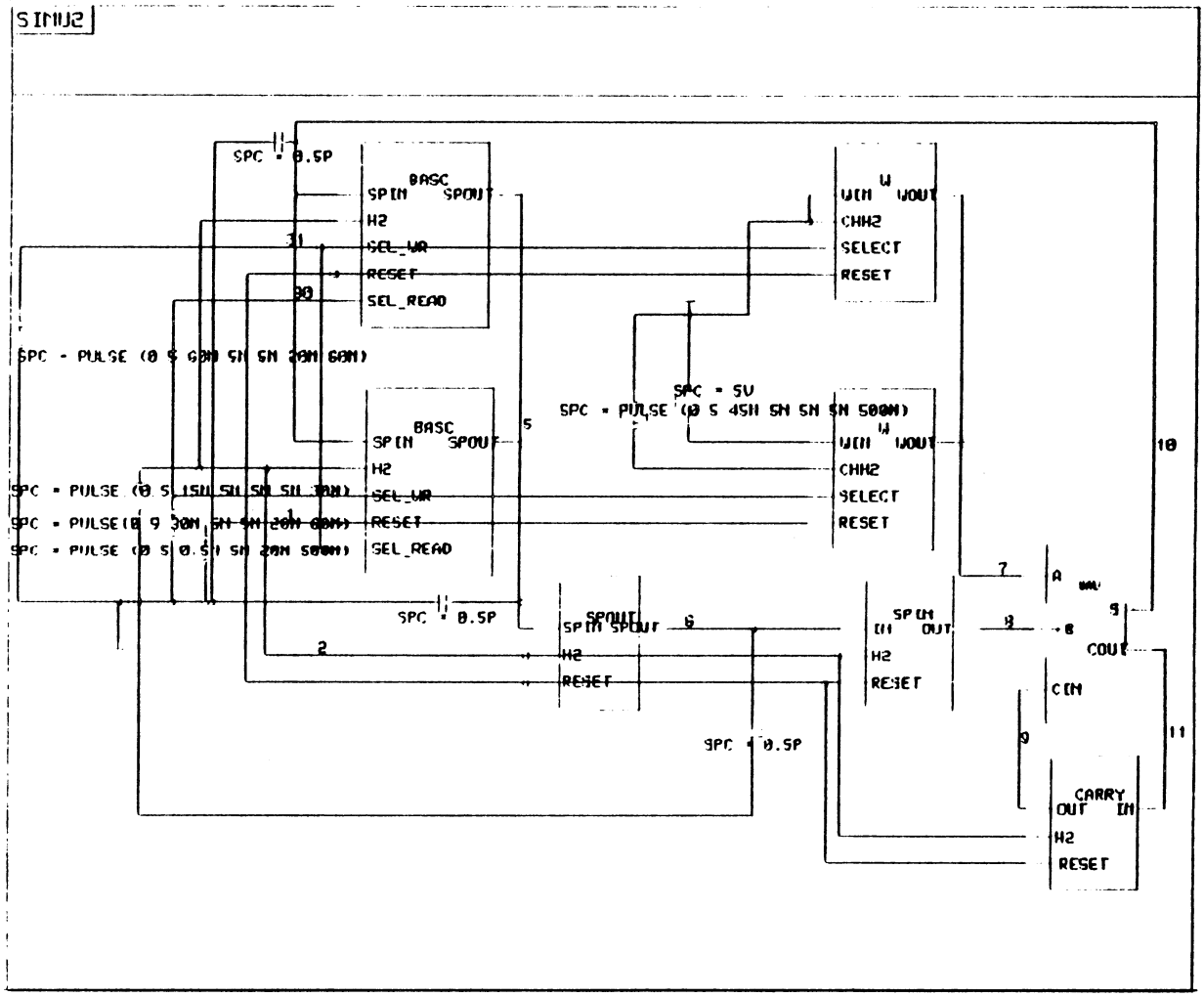

BASCOUT
BASC1IN
SELECT 1
SELECT 2
H2
0.
30.
60.
90.
120.
150.
180.
210.
240.
270.
300.

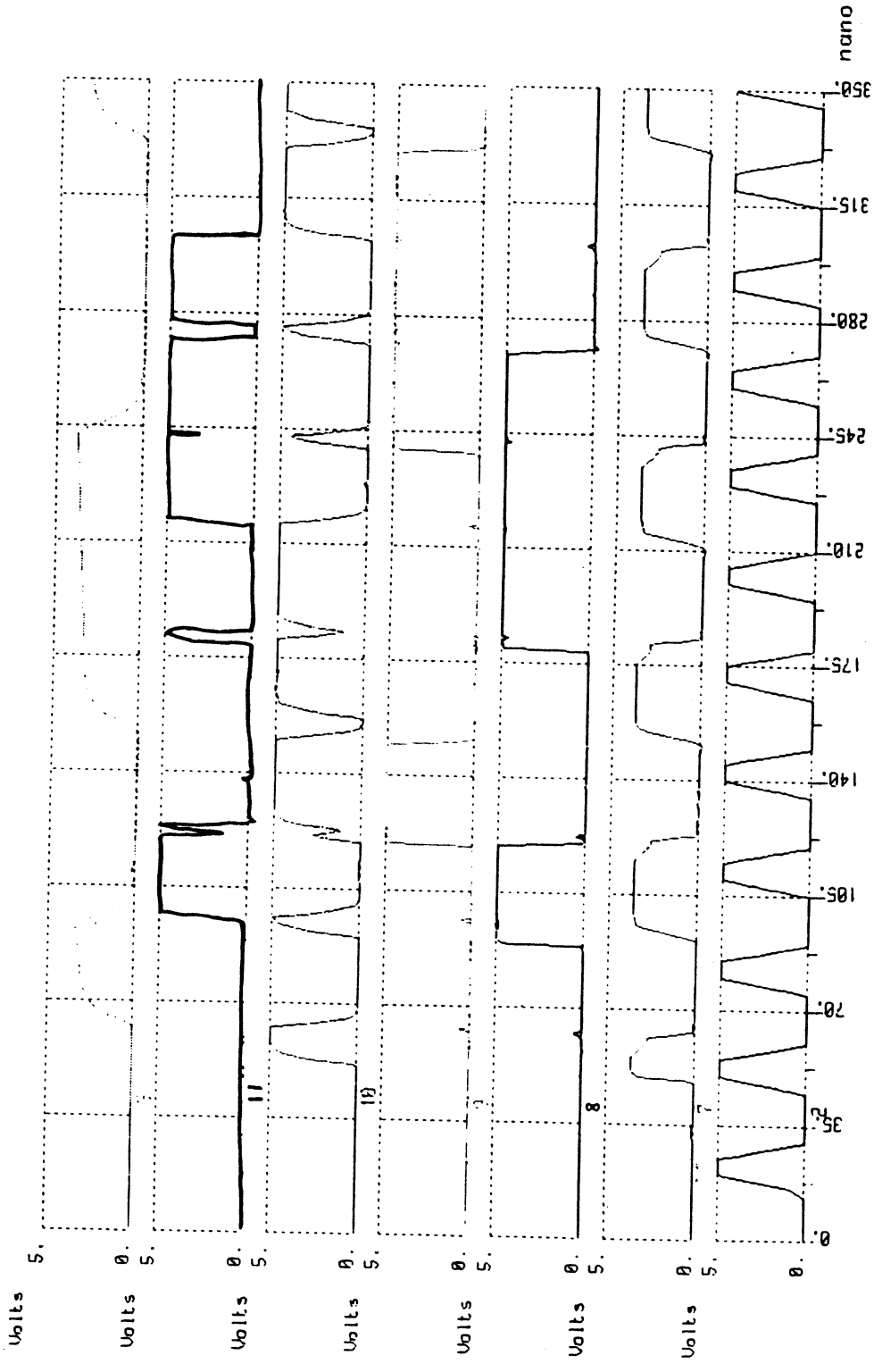

XOR1
BPLUS
SPC = PMOS L=2U W=3U
SPC = PMOS L=2U W=3U
SPC = 1.5U
A
XOR
SPC = NMOS L=2U W=3U
IN INVERSEUR OUT
B
SPC = NMOS L=2U W=3U
IN INVERSEUR OUT

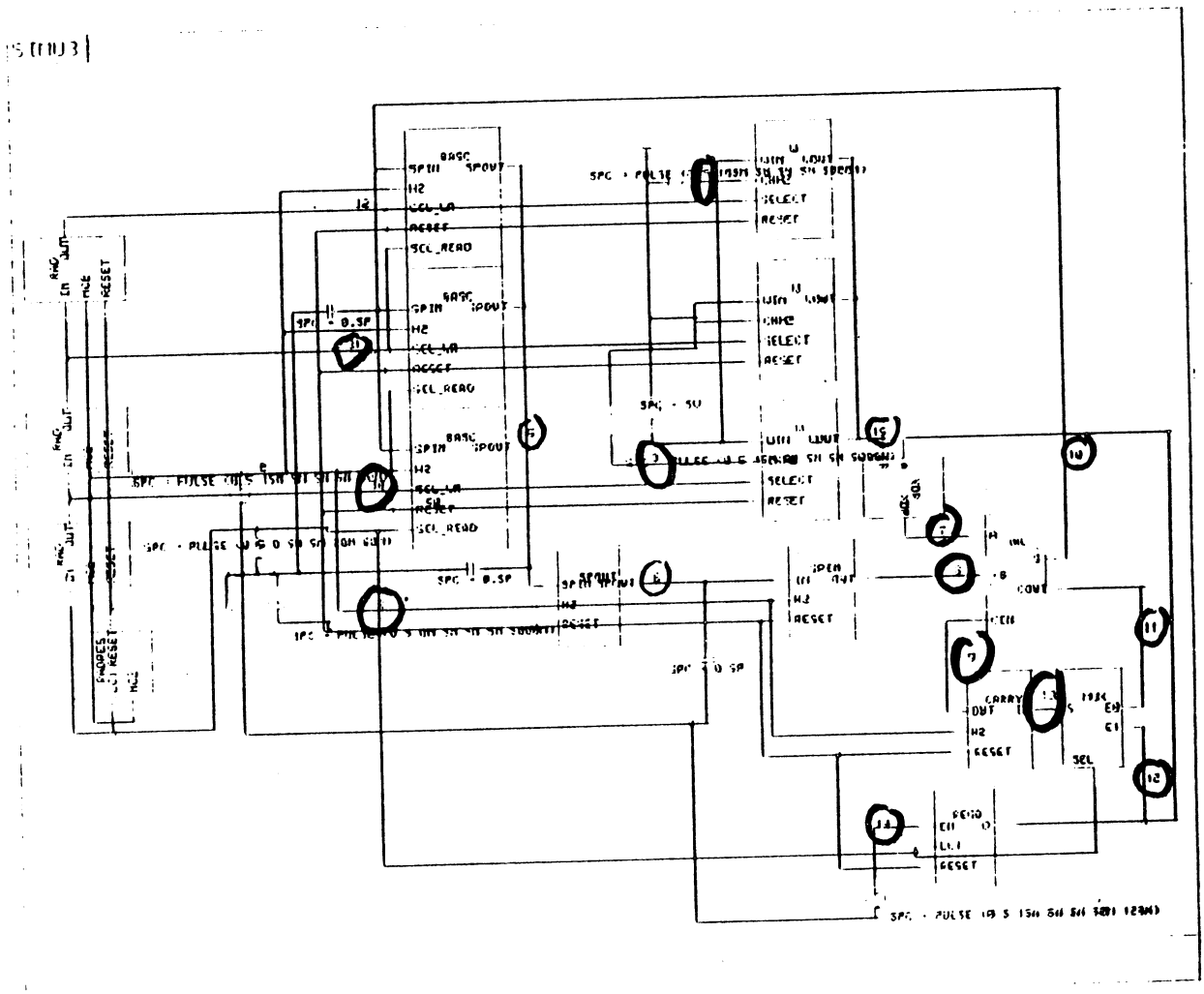

UAL
NAND
IN1 OUT
IN2
XOR1
A XOR
B BPLUS
B
A
5
NAND
IN1 OUT
IN2
8
COUT
SPC = 0.01P
7
SPC = 0.01P
SPC = 0.01P
NAND
IN1 OUT
IN2
SPC = 0.01P
3 CIN
XOR1
A XOR
B BPLUS
IN INV OUT
IN INV OUT
6
S
SPC = 0.01P

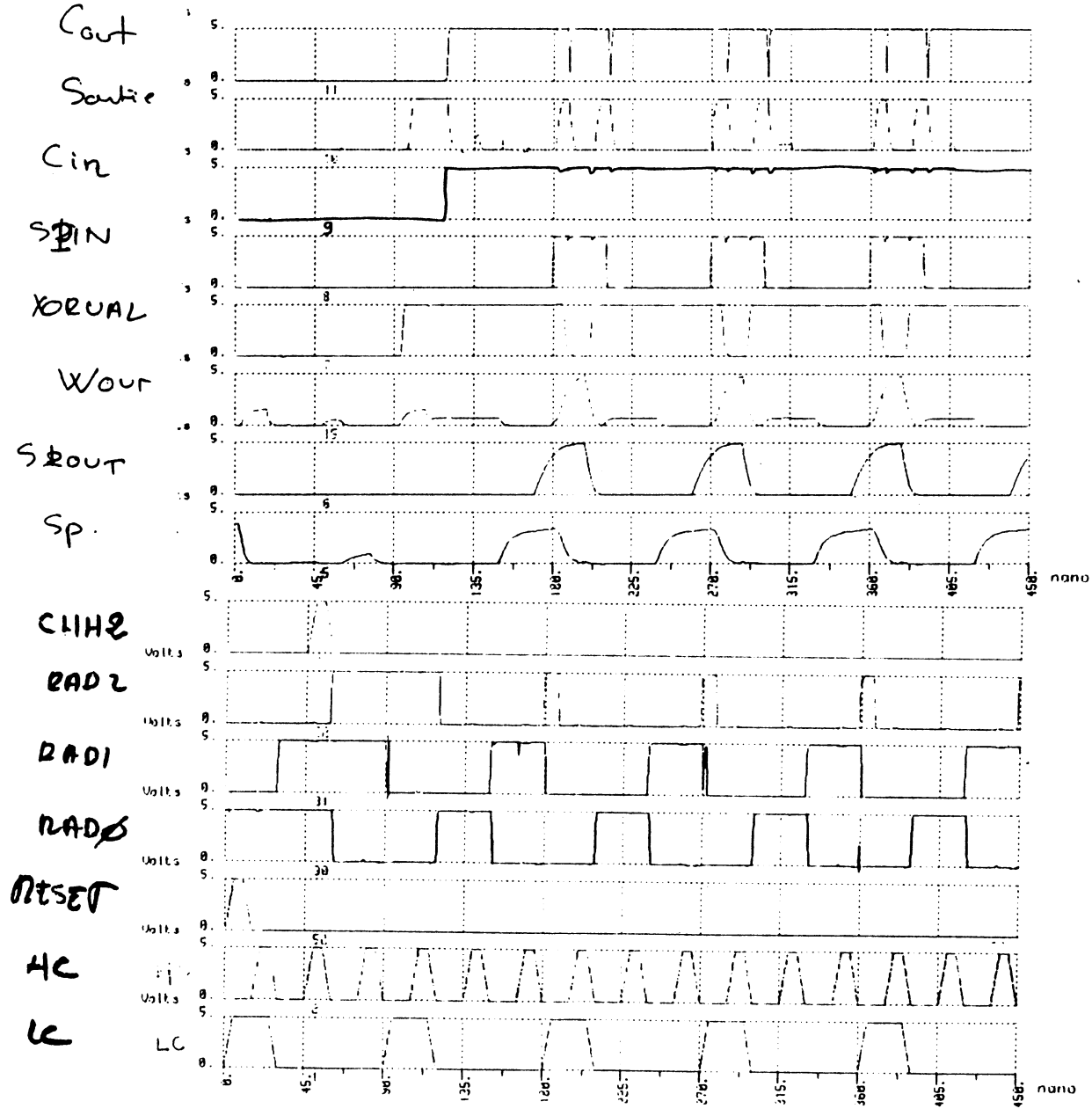

SIMUAL
4
3
2
1
SPC = PULSE (0 5 30N 5N 5N 20N 60N)
SPC = PULSE (0 5 30N 5N 5N 5N 30N)
SPC = PULSE 0 5 15N 5N 5N 5N 30N
SPC = PULSE 0 5 0 5N 5N 20N 50ON
SP IN
IN OUT
H2
RESET
A
UAL
B
5
S
8
COUT
7
SPC = 0.01P
CIN
6
SPC = 0.01P
CARRY
OUT IN
H2
RESET
SPC = 0.01P
SPC = 0.5P

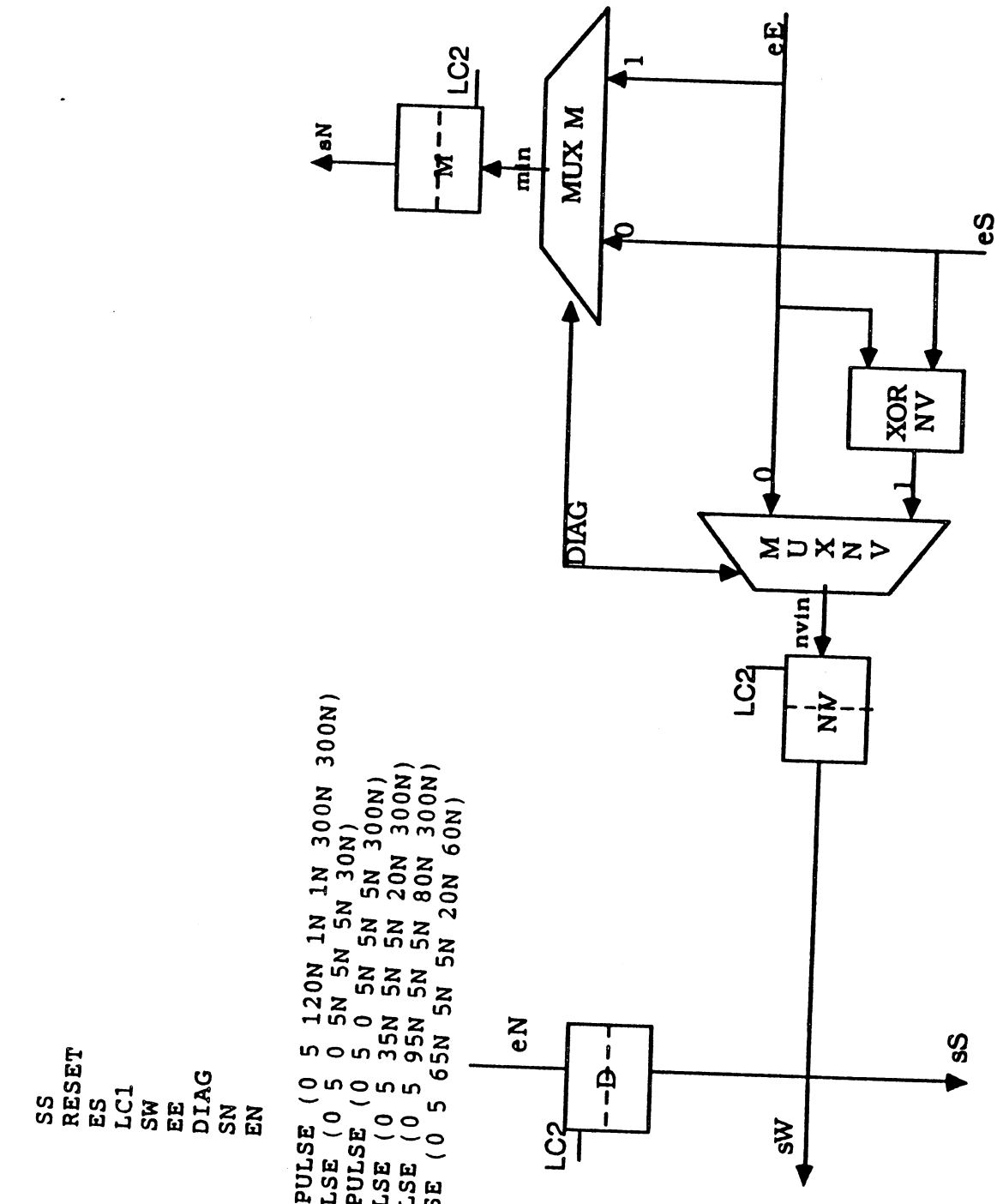

8
7
6
5
4
3
2

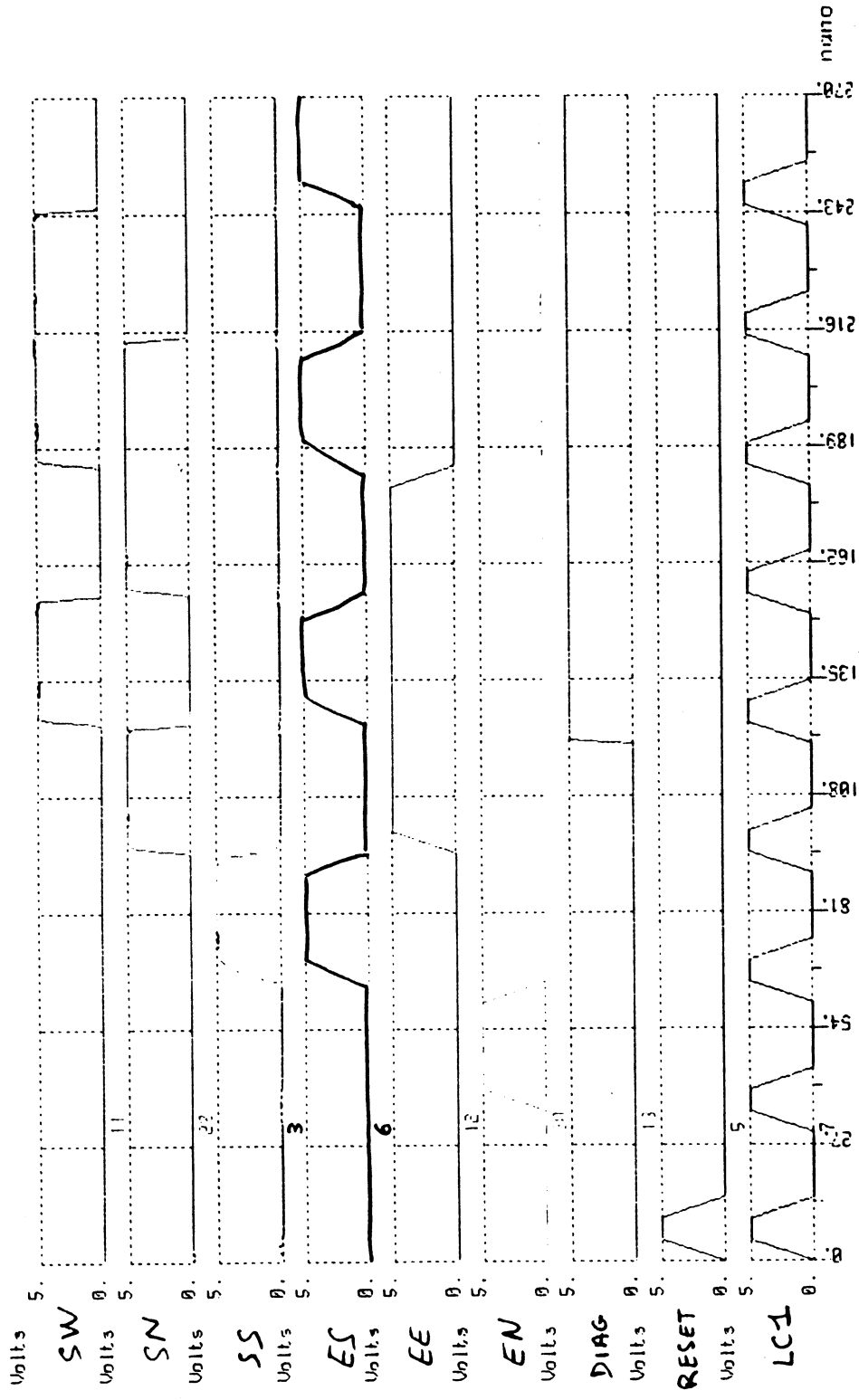

SINUS
SPC = 0.5P
BASC
SPIN SPOUT
H2
SEL_WR
RESET
SEL_READ
W
WIN WOUT
CHH2
SELECT
RESET
SPC = PULSE (0 5 60N 5N 5N 20N 60N)
SPC = 5V
SPC = PULSE (0 5 45N 5N 5N 5N 500N)
SPC = PULSE (0 5 15N 5N 5N 5N 30N)
SPC = PULSE (0 5 30N 5N 5N 20N 60N)
SPC = PULSE (0 5 0.5N 5N 20N 500N)
SPOUT
SPIN SPOUT
SPIN
IN OUT
A
B
COUT
CIN
CARRY
OUT IN
1
2
5
6
7
8
9
10
11

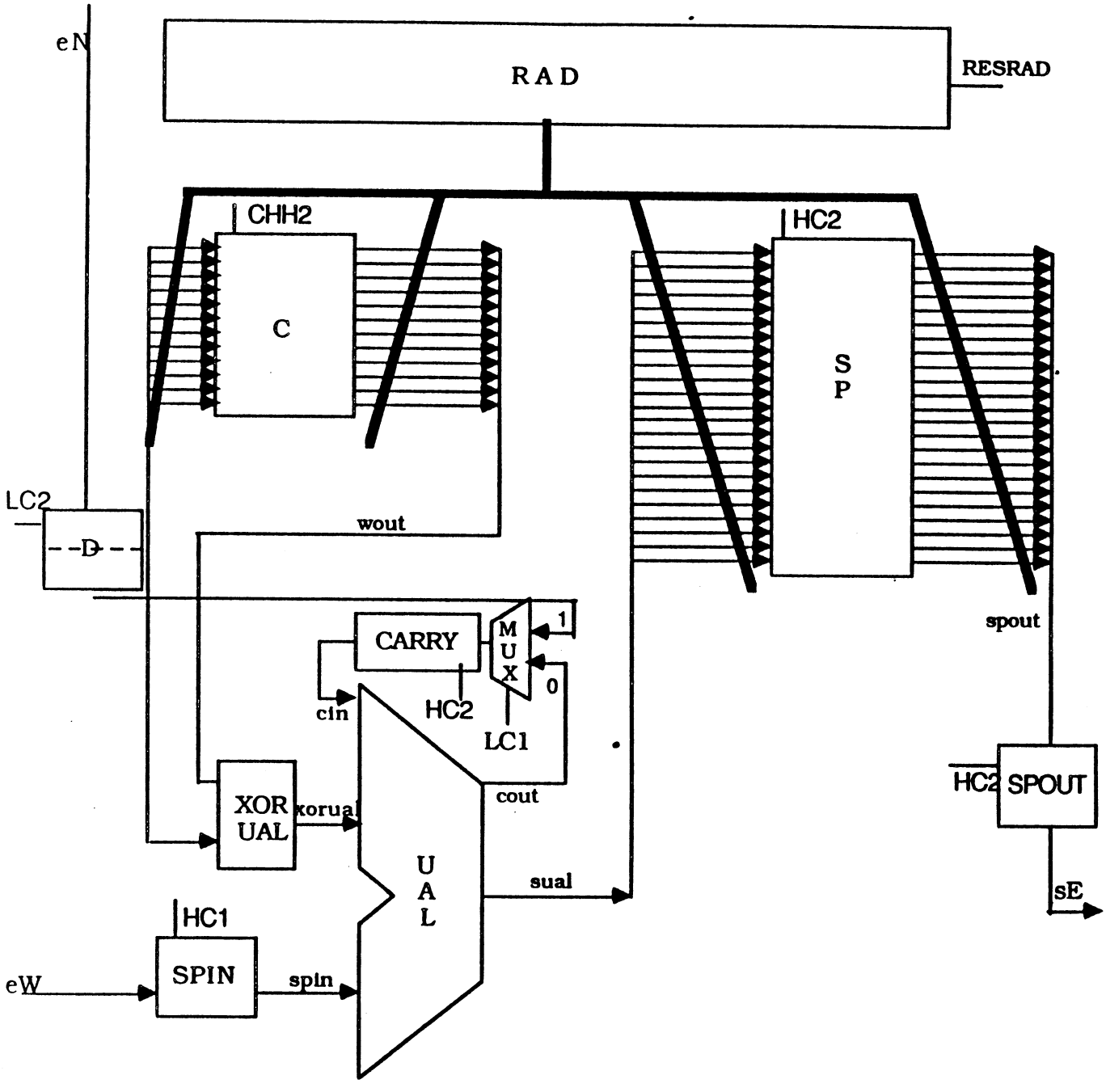

nano
350.
315.
280.
245.
210.
175.
140.
105.
70.
35.
0.
Volts
5.
0.
11
10
9
8
7

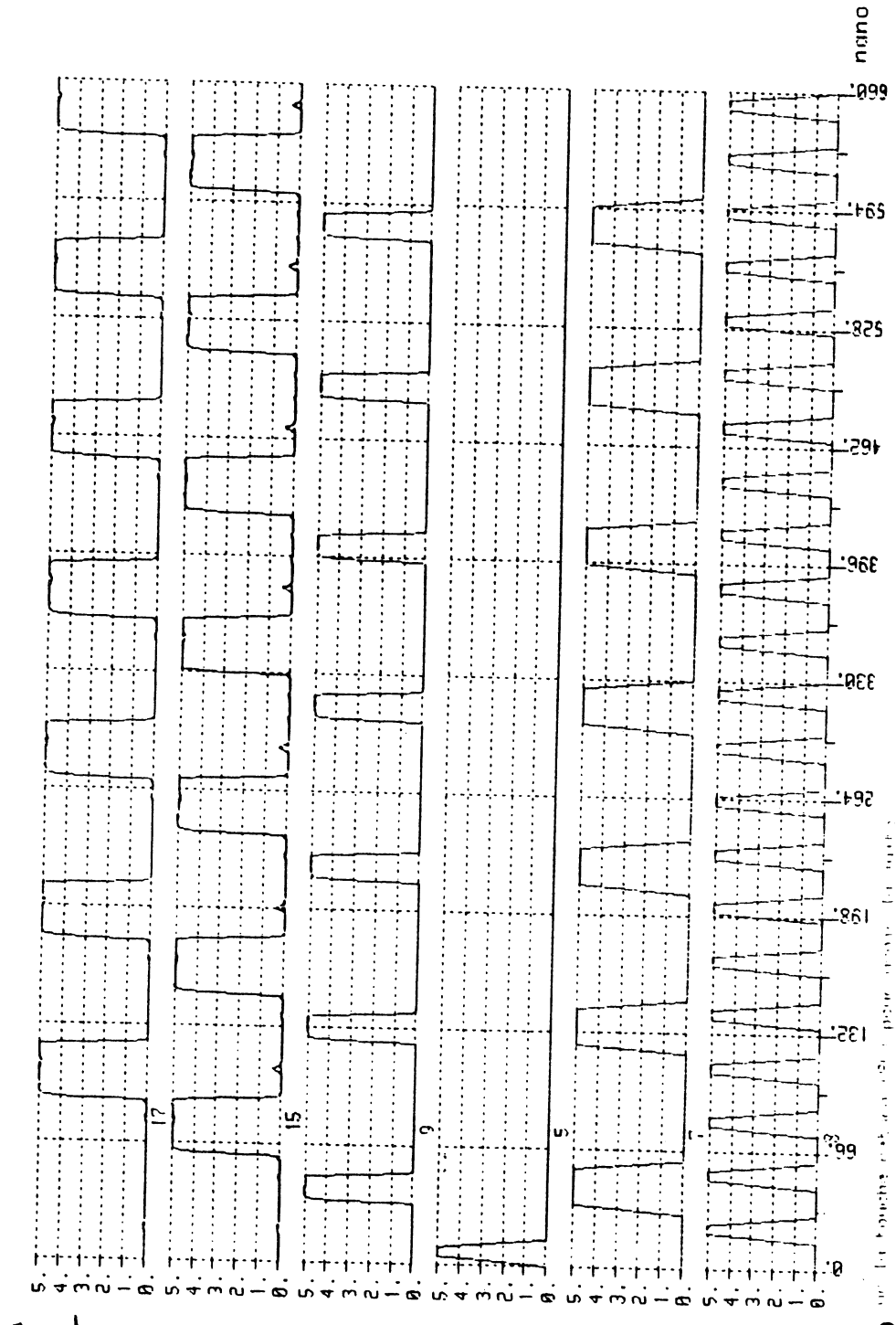


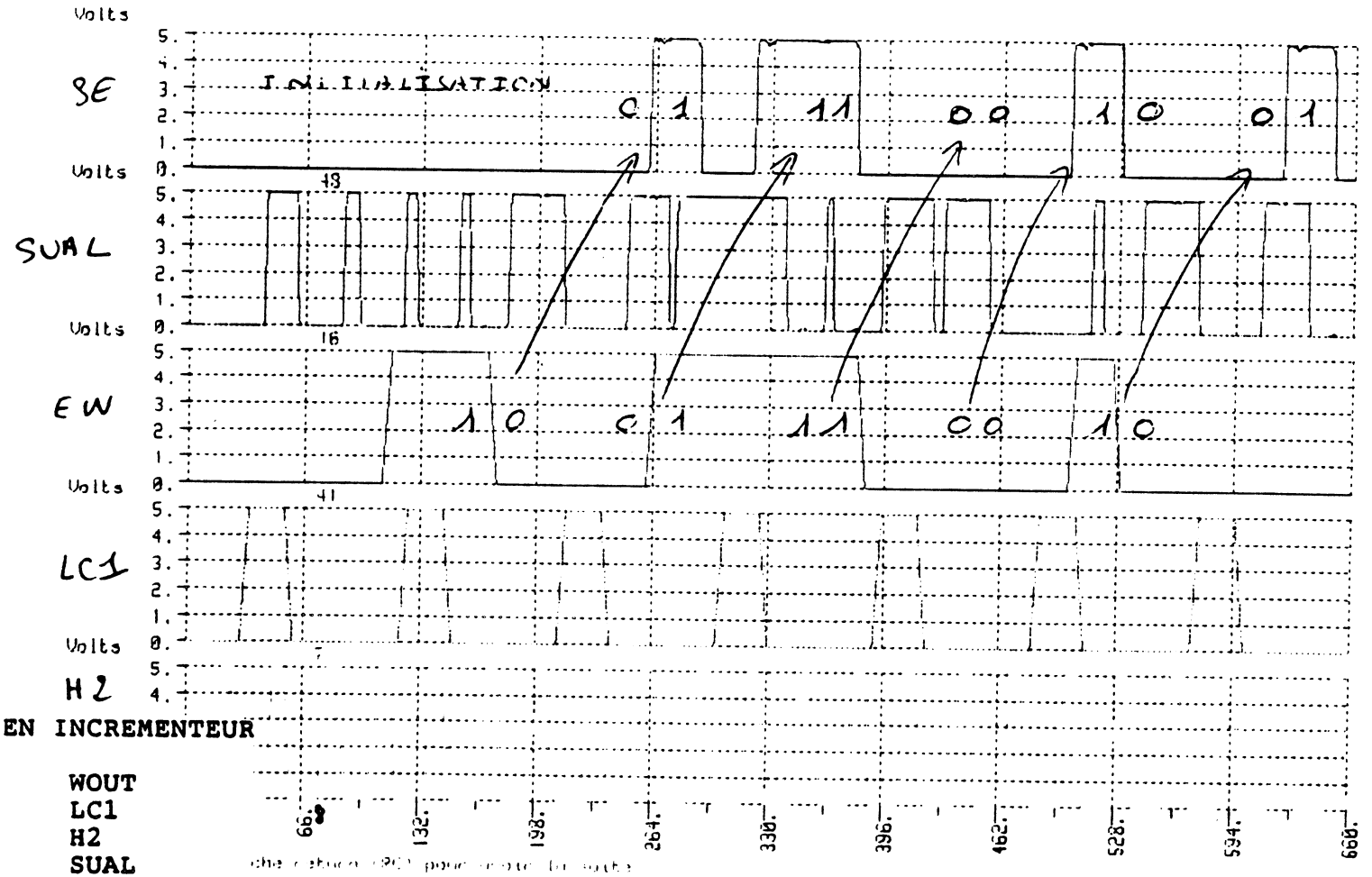

Cout
Sortie
Cin
SPIN
XORUAL
WOUT
SROUT
Sp.
nano
CLIH2
RAD2
RAD1
RAD0
RESET
HC
LC
Volts
nano


SS
RESET
ES
LC1
SW
EE
DIAG
SN
EN
* 3
* 5
* 6
* 7
* 11
* 12
* 13
* 28
* 30
VCC 1 0 5V
VDIAG 13 0 PULSE (0 5 120N 1N 1N 300N 300N)
VLC1 7 0 PULSE (0 5 0 5N 5N 30N)
VRESET 5 0 PULSE (0 5 0 5N 5N 300N)
VEN 30 0 PULSE (0 5 35N 5N 5N 20N 300N)
VEE 12 0 PULSE (0 5 95N 5N 5N 80N 300N)
VES 6 0 PULSE (0 5 65N 5N 5N 20N 60N)
LC2
M
eN
sN
min
MUX M
1
0
eE
eS
XOR NV
DIAG
MUX NV
nvin
NV
sW
D
sS


Volts
SW
SN
SS
ES
EE
EN
DIAG
RESET
LC1
5.
0.
11
22
3
6
12
13
5
0
27
54
81
108
135
162
189
216
243
270
nano

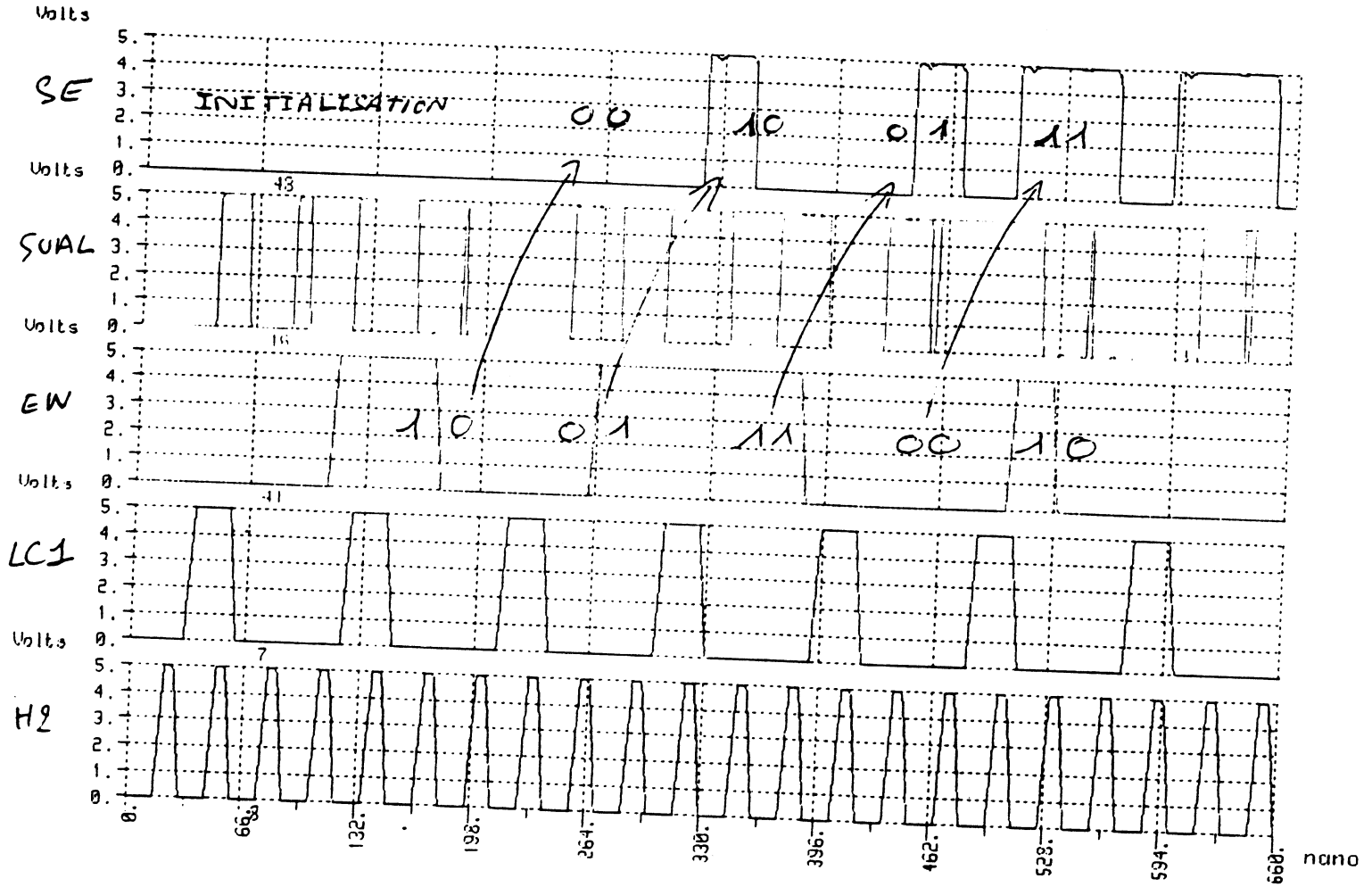

eN
RAD
RESRAD
CHH2
C
HC2
SP
LC2
D
wout
spout
CARRY
MUX
1
0
cin
HC2
LC1
cout
XOR
UAL
xorual
UAL
sual
HC2
SPOUT
sE
HC1
SPIN
eW
spin


SELECT1
SELECT0
RESRAD
RESET
LC1
H2
Volts
nano
0.
66.
132.
198.
264.
330.
396.
462.
528.
594.
660.
17
15
9
5
1
RESET
LC1
H2
RESRAD
SELECT0
SELECT1
5
7
8
9
15
17
CC 1 0 5V
H2 8 0 PULSE (0 5 15N 5N 5N 30N)
LC1 7 0 PULSE (0 5 30N 5N 5N 20N 90N)
RESET 5 0 PULSE (0 5 0 5N 5N 5N 2000N)



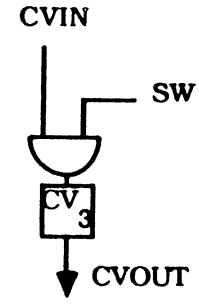
SIMULATION EN INCREMENTEUR

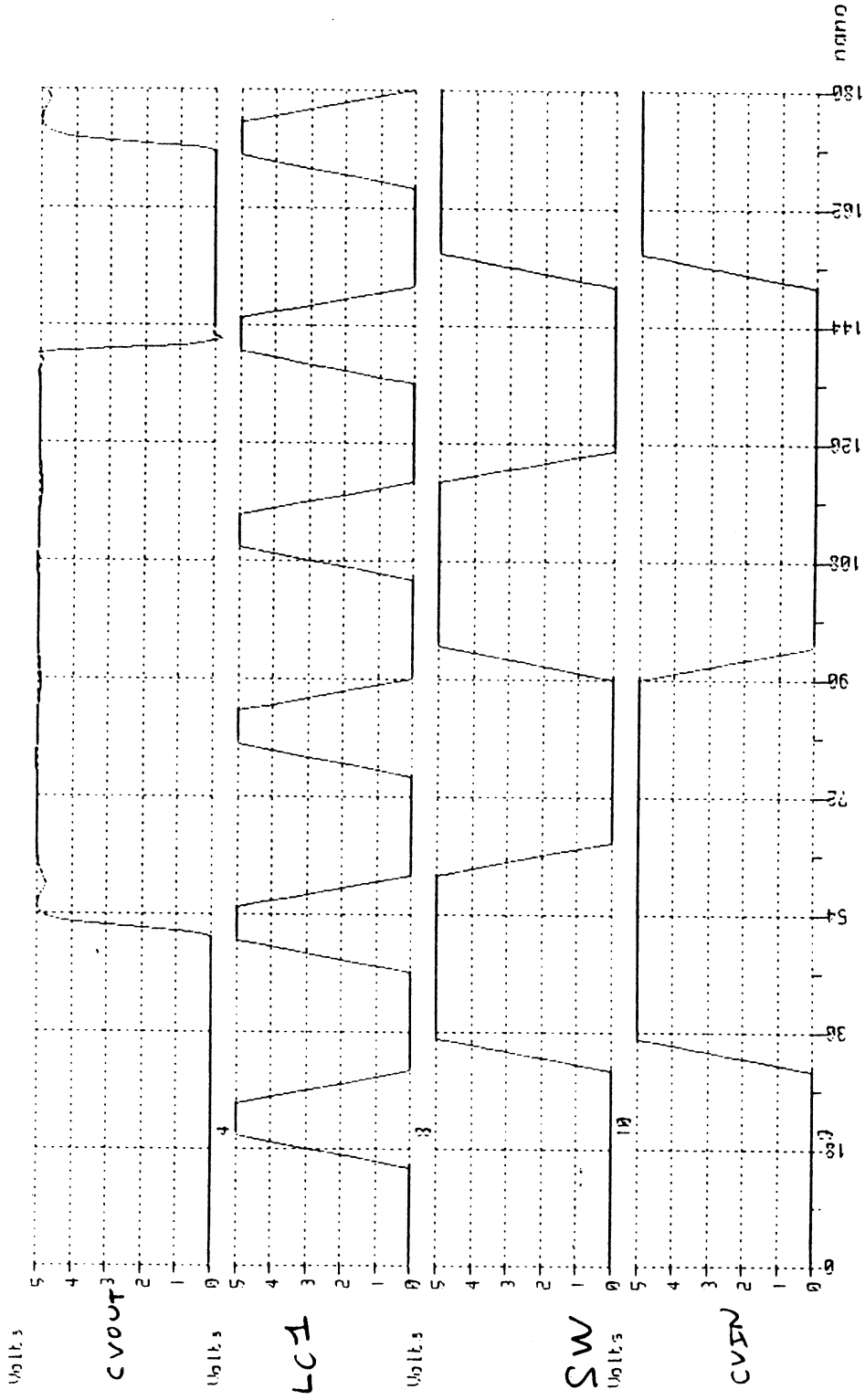
```
* 4              WOUT
* 7              LC1
* 8              H2
* 16             SUAL
* 41             EW
* 48             SE
VCC 1 0 5v
VH2 8 0 PULSE (0 5 15N 5N 5N 5N 30N)
VLC1 7 0 PULSE (0 5 30N 5N 5N 20N 90N)
VEW 41 0 PWL (0 0 110N 0 115N 5 170N 5 175N 0 260N 0 265N 5 380N 5 385N 0
+ 500N 0 505N 5 525N 5 530N 0V 600N 0)
```

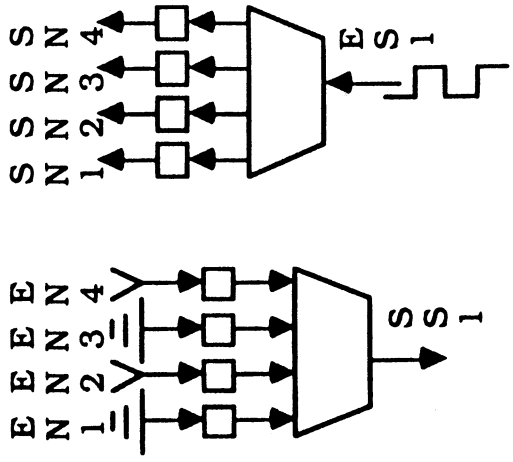


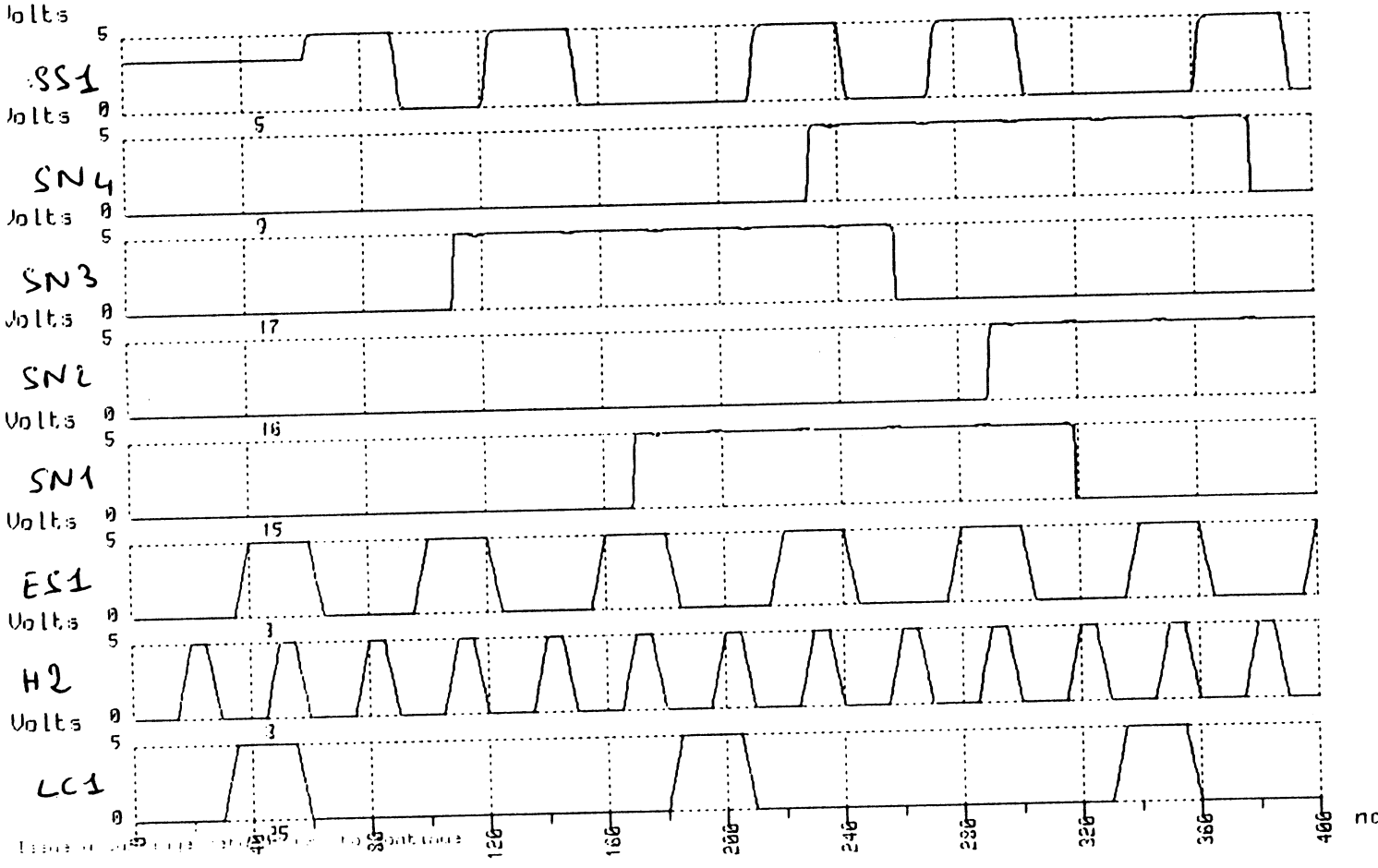
SIMULATION EN DECREMENTEUR

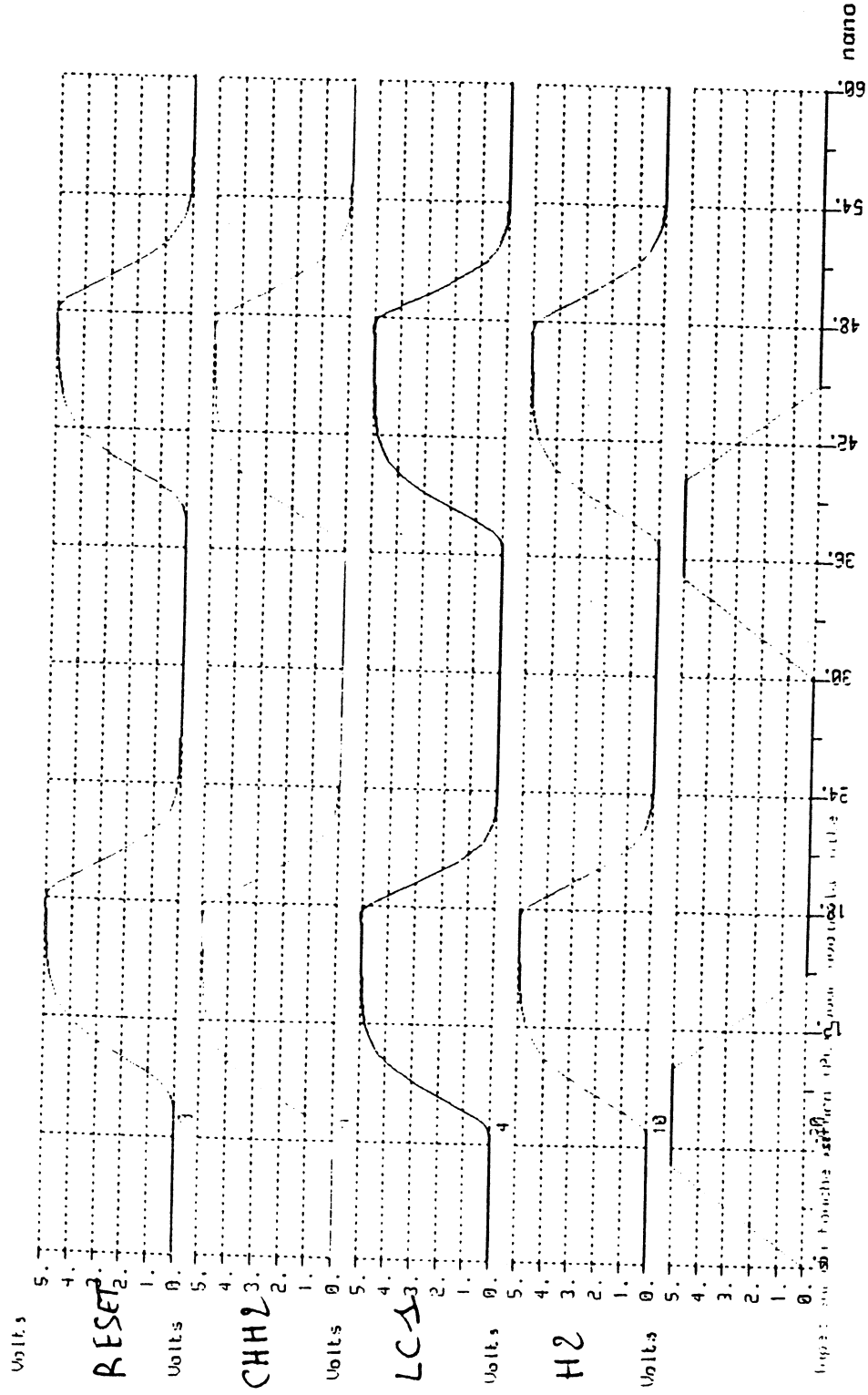
```
* 4              WOUT
* 7              LC1
* 8              H2
* 16             SUAL
* 41             EW
* 48             SE
VCC 1 0 5V
VH2 8 0 PULSE (0 5 15N 5N 5N 5N 30N)
VLC1 7 0 PULSE (0 5 30N 5N 5N 20N 90N)
VEW 41 0 PWL (0 0 110N 0 115N 5 170N 5 175N 0 260N 0 265N 5 380N 5 385N 0
+ 500N 0 505N 5 525N 5 530N 0V 600N 0)
```

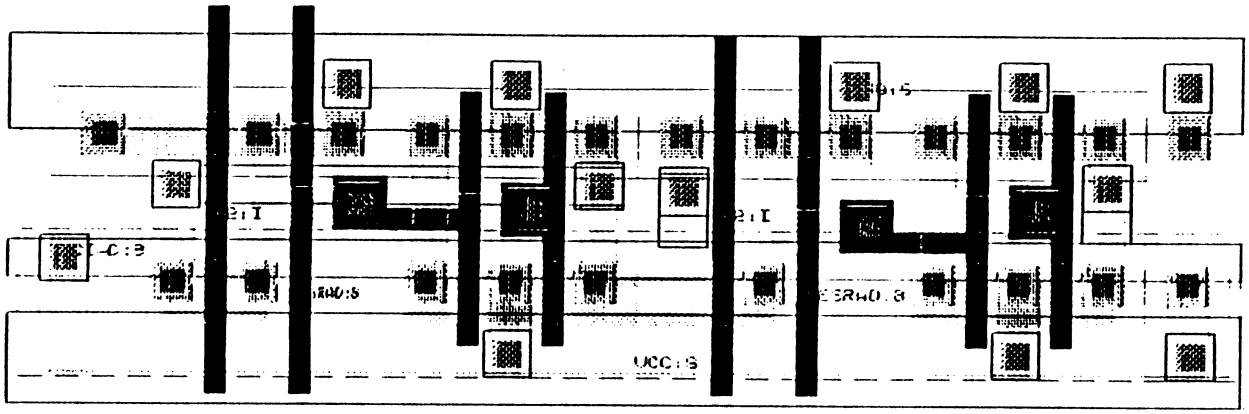

CVIN
SW
CV
3
CVOUT

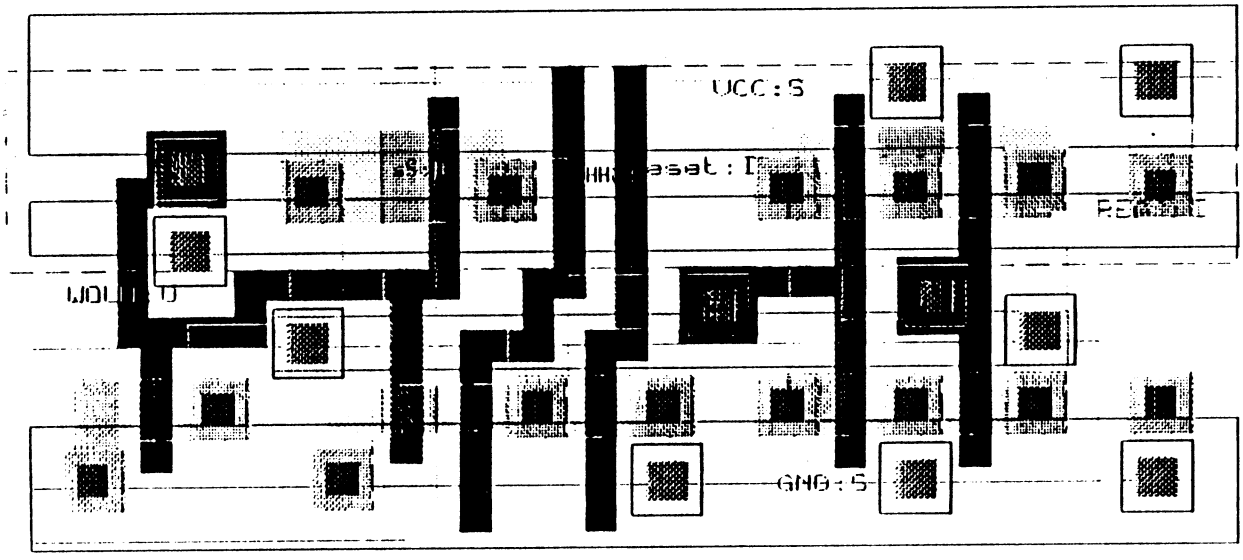

nano
180
162
144
126
108
90
72
54
36
18
0
CVOUT
Volts
LC1
Volts
SW
Volts
CVIN

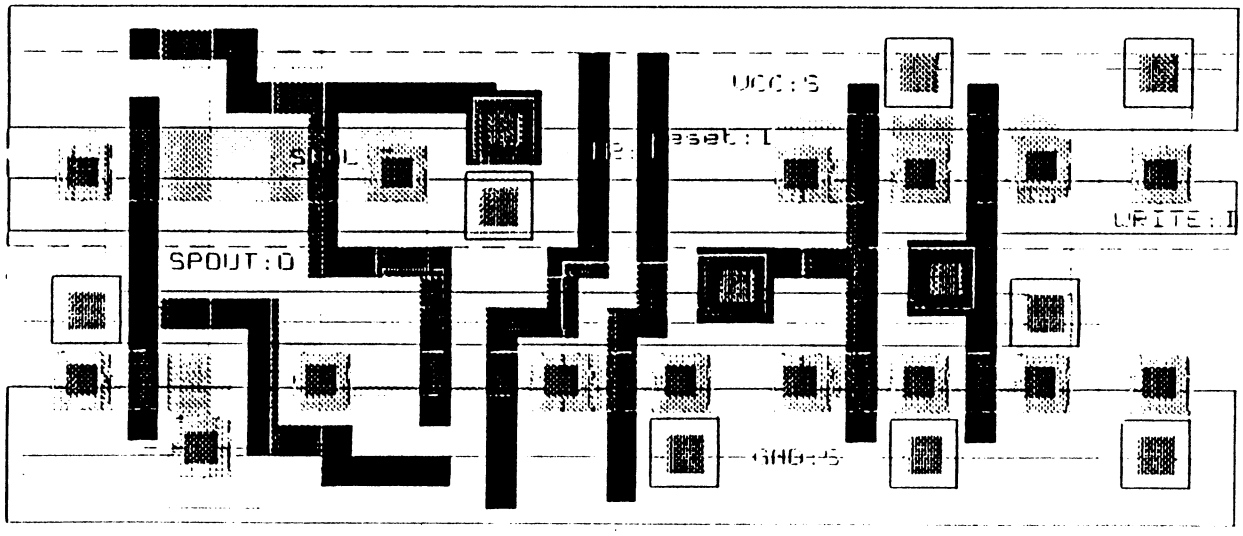

S N N S 4
S N N S 3
S N N S 2
S N N S 1
E S 1
E N N E 4
E N N E 3
E N N E 2
E N N E 1
S S 1

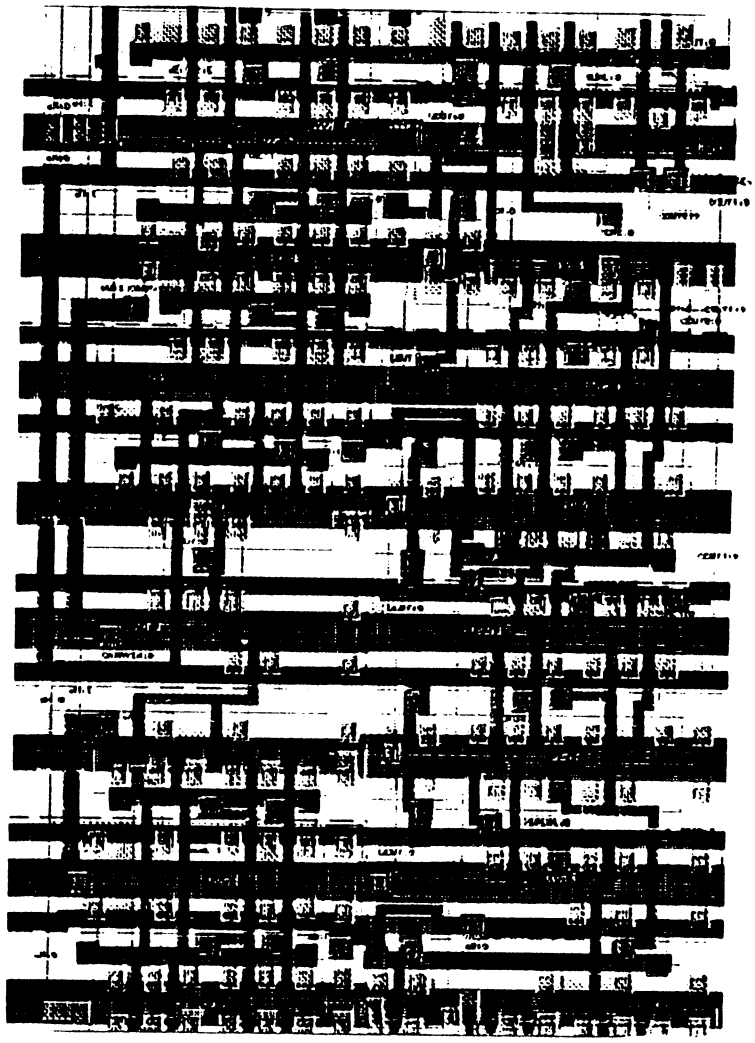

Volts
5
SS1
Volts
0
5
SN4
Volts
0
5
SN3
Volts
0
5
SN2
Volts
0
5
SN1
Volts
0
5
ES1
Volts
0
5
H2
Volts
0
5
LC1
0
5
9
17
16
15
3
3
126
166
206
246
286
326
366
406
no

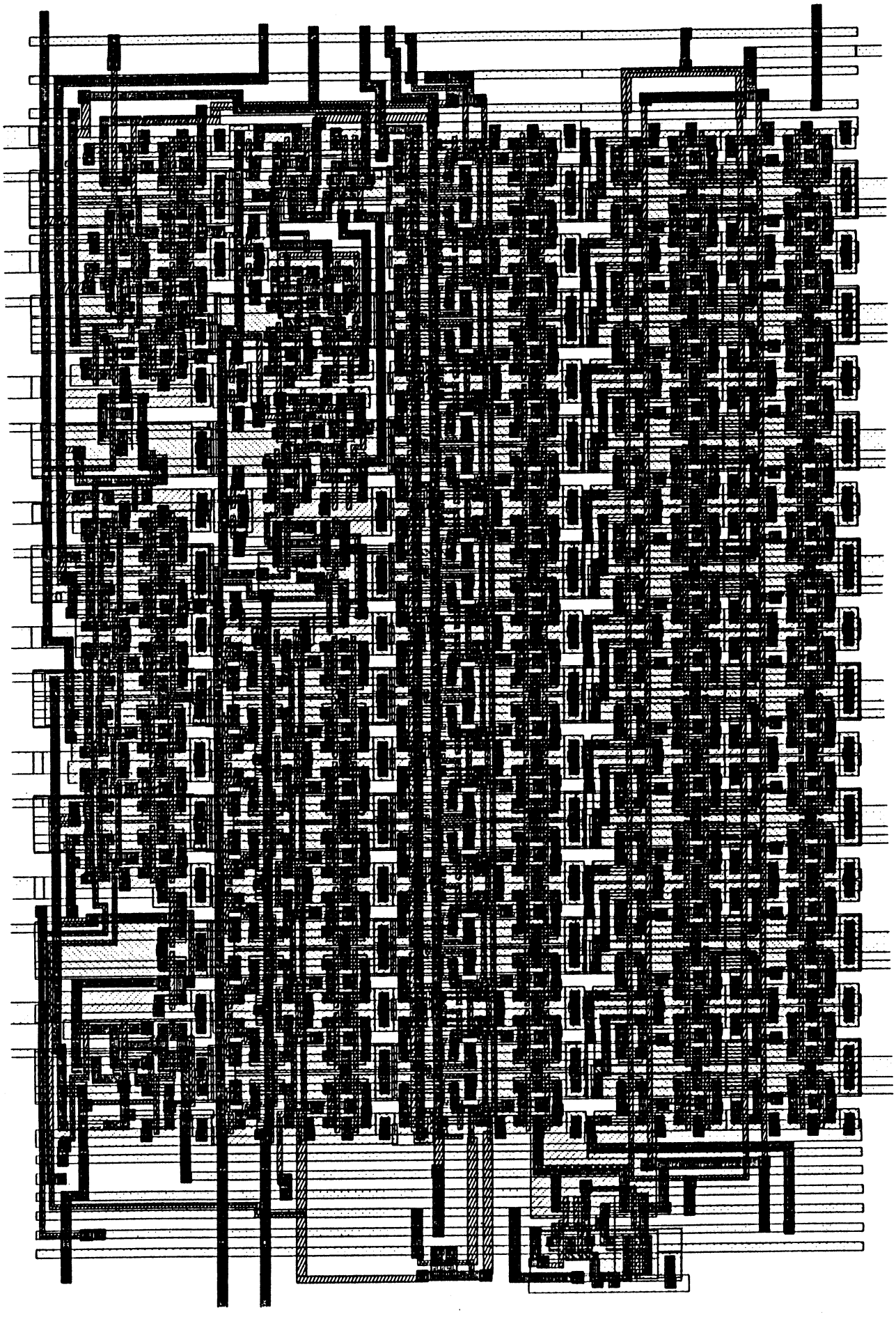

nano
60.
54.
48.
42.
36.
30.
24.
18.
12.
6.
0.
Volts
RESET
CHH2
LCX
H2
10

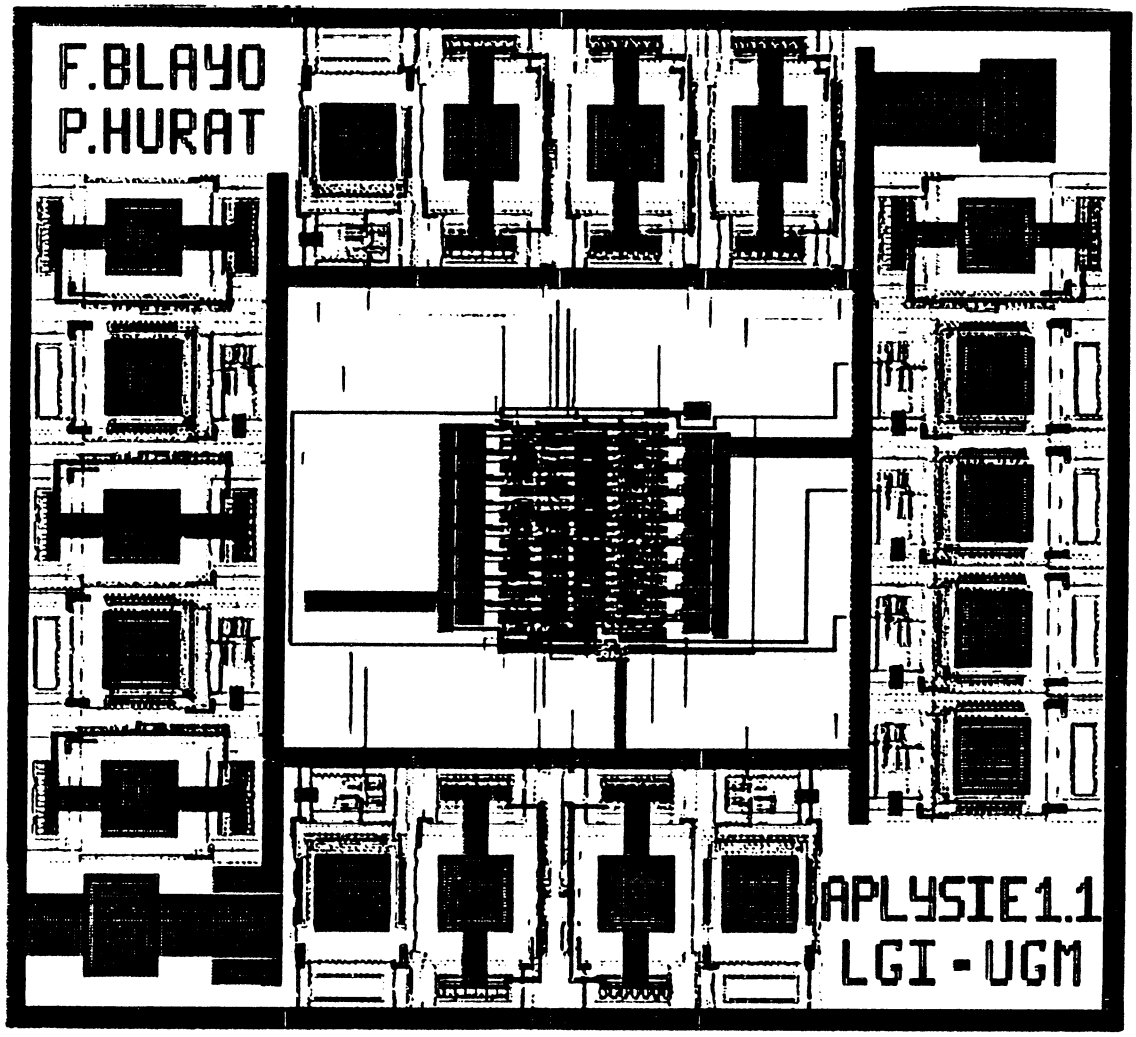

VCC:S
GND:S

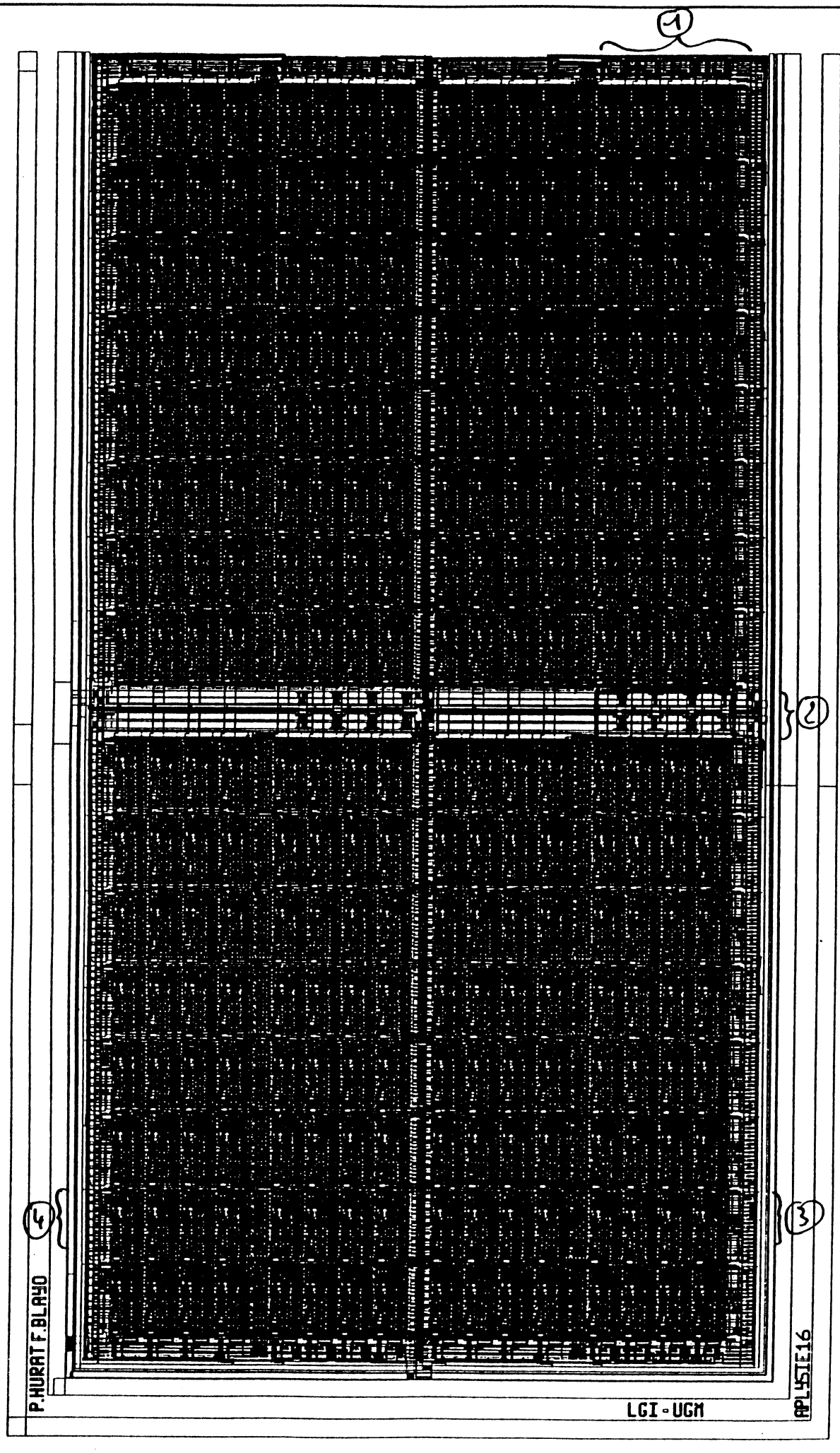

VCC:S
GND:S

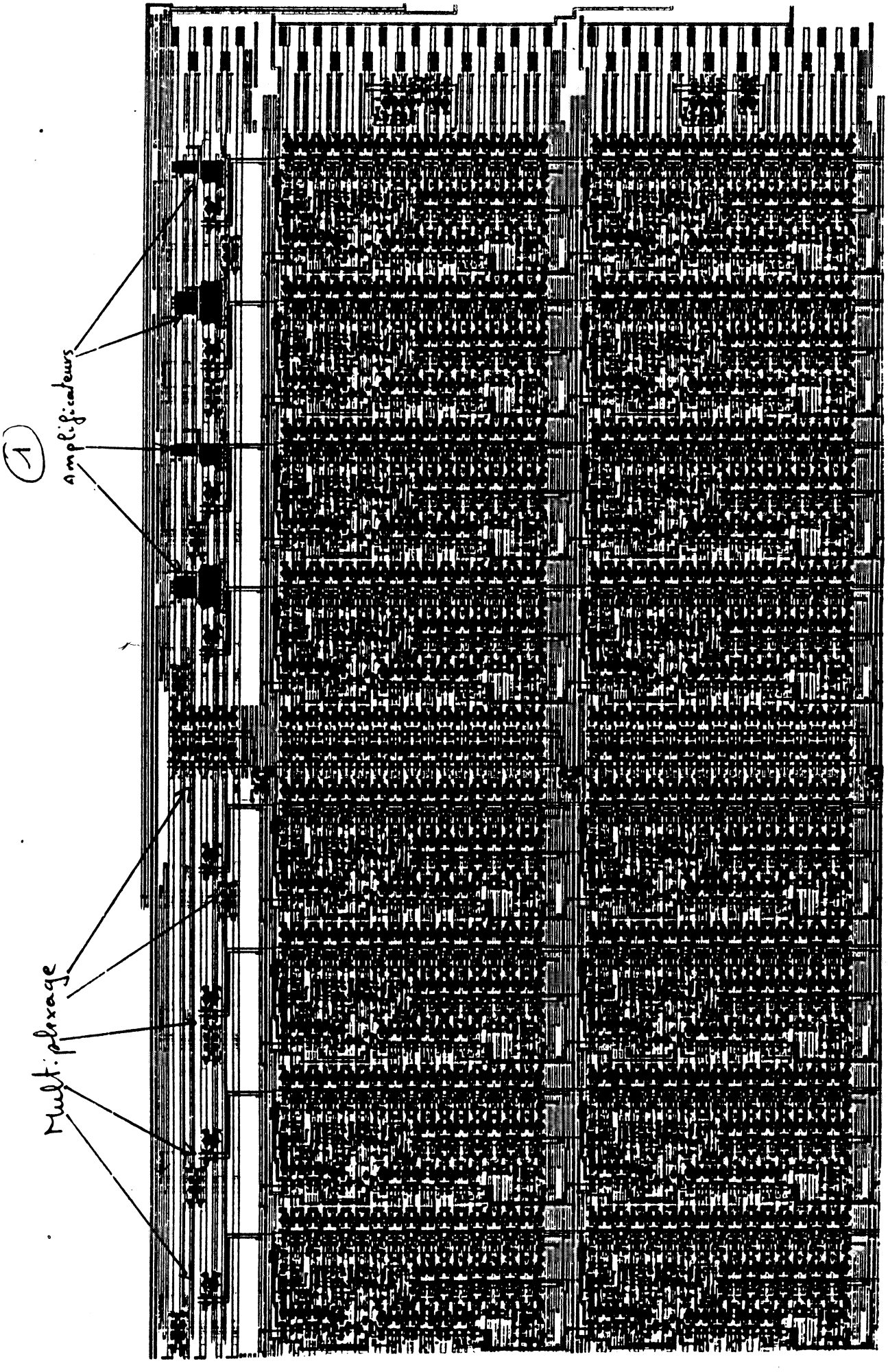

VCC:S
SPOUT:O
GND:S

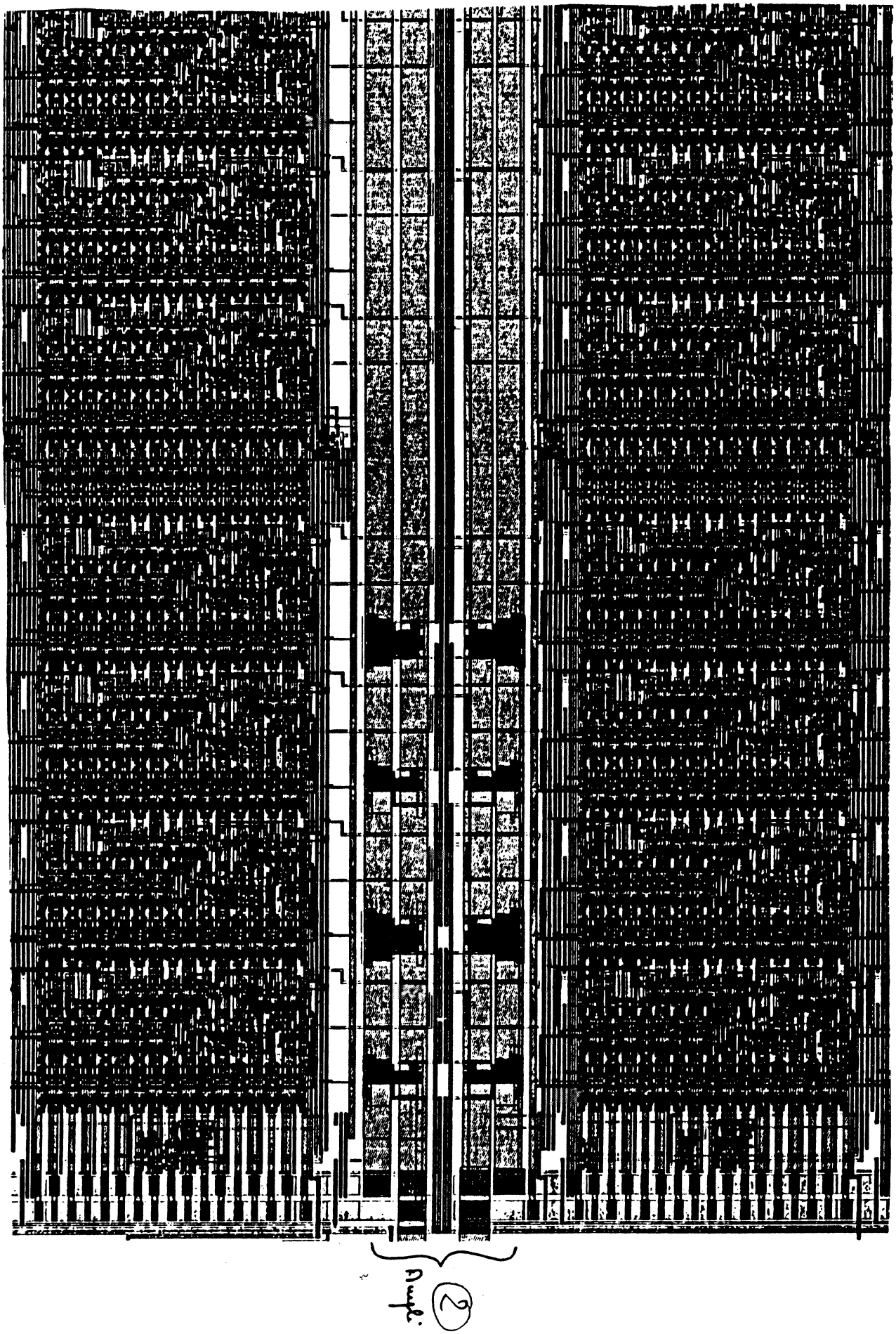


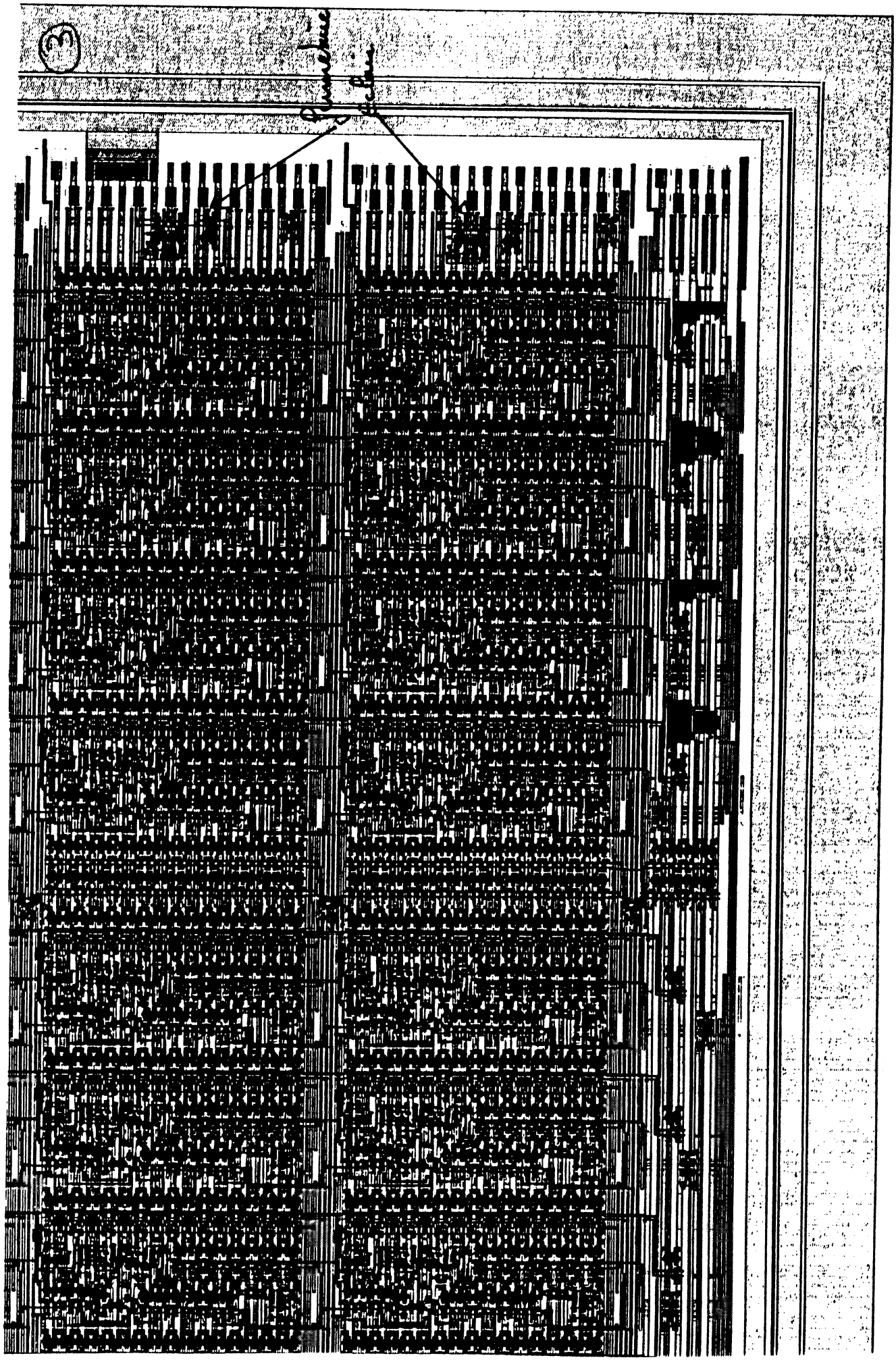


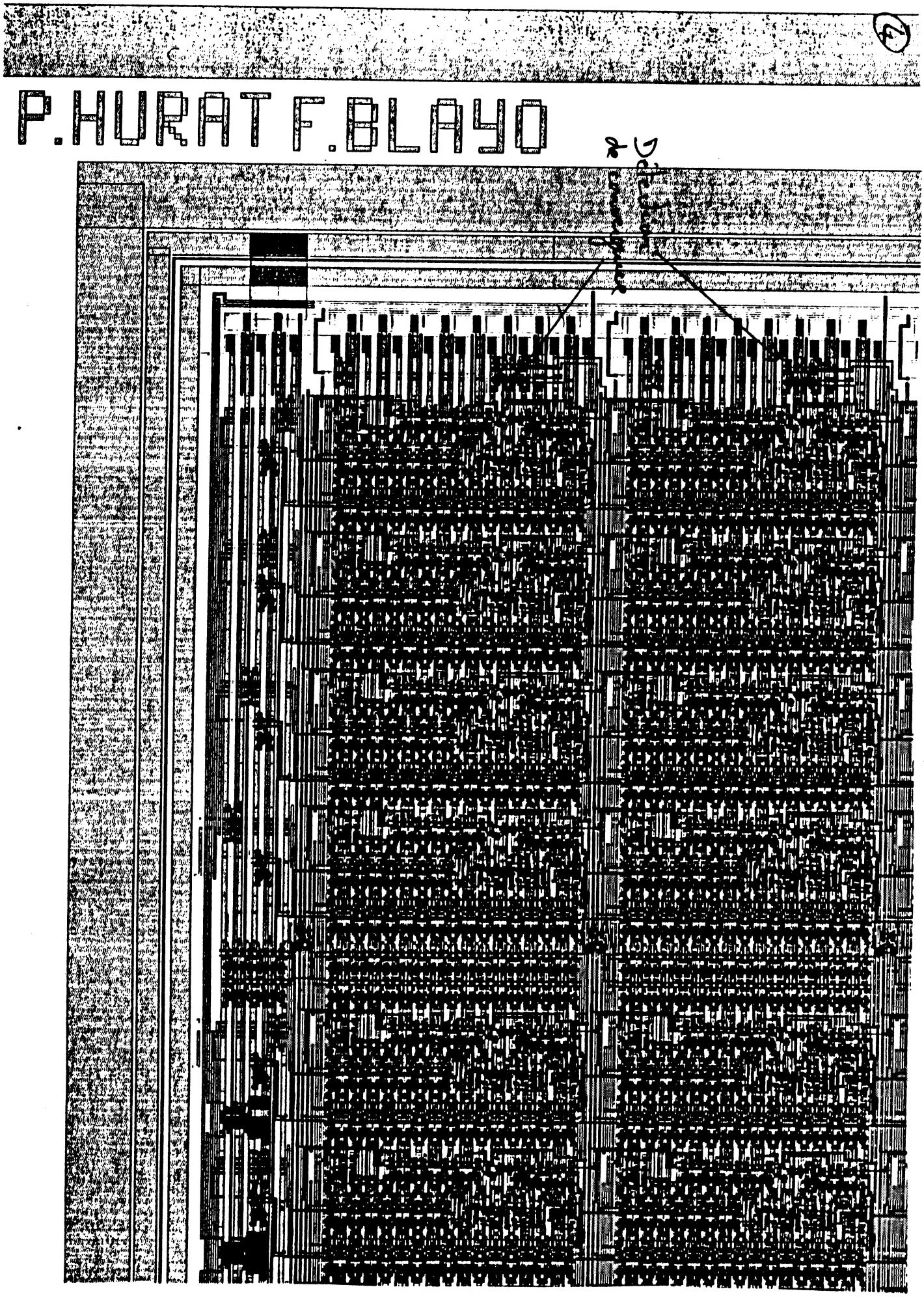

F.BLAYO
P.HURAT
APLYSIE1.1
LGI-UGM


1
2
3
4
P.HURAT F.BLAYO
LGI-UGM
APLUSIE16


(1)
amplificateurs
Mult. plexage






P.HURAT F.BLAYO

Bibliographie

[AMA83] AMARI S.I.
Field Theory of Self-Organizing Neural Nets
IEEE Transactions on Systems, Man, and cybernetics, Vol.SMC-13 No5, Septembre/Octobre 1983.

[BAU86] BAUM E.B.
Towards Practical "Neural" Computation for Combinatorial Optimization Problems
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.53-58.

[BLA87] BLAYO F., HURAT Ph.
Approche Systolique pour une Architecture Neuro-Mimétique
IMAG, LGI/UGM, Rapport Technique 29, Grenoble, France, Octobre 1987.

[BLA88a] BLAYO F., HURAT Ph.
Intégration d'une Architecture Systolique Neuro-mimétique
Journée Internationale, "Automates Cellulaires et Neurones Artificiels", LAMI, EPFL, Lausanne, Suisse, Mars 1988.

[BLA88b] BLAYO F., HURAT Ph.
A Systolic Architecture Dedicated to Neural Networks
N'euro 88, Paris, France, Juin 1988.

[BLA88c] BLAYO F., HURAT Ph.
A VLSI Systolic Array Dedicated to Hopfield Neural Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[BLA88d] BLAYO F., HURAT Ph.
Définition et Intégration d'un Réseau Systolique Neuro-mimétique
Neuro-Nîmes 88, Journées Internationales "Les réseaux neuro-mimétiques et leurs applications", Nîmes, France, Novembre 1988.

[BLA89] BLAYO F., HURAT Ph.
A Reconfigurable WSI Neural Network
IEEE International Conference on Wafer Scale Integration, San Francisco, USA, Janvier 1989.

[BRU86] BRUCE A.D., CANNING A., FORREST B., GARDNER E., WALLACE DJ.
Learning and Memory Properties in Fully Connected Networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.65-70.

[BUT88] BUTLER Z.F., MURRAY A.F., SMITH A.V.W.
VLSI Bit-Serial Neural Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[CAR86] CARPENTER G.A., GROSSBERG S.
Absolutely Stable Learning of Recognition Codes by a self-organizing Neural Network
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.77-85.

[CAS88] CASPI P.
Etude des Résaux de Hopfield par Bijection avec un Problème d'Optimisation Quasi-Linéaire
Publication interne, IMAG, LGI/UGM, Grenoble, France 1988.

[CHE85] CHEVALIER G., SAUCIER G.
A Programmable Switch for Fault Tolerant WSI of Processor Arrays
International Workshop on WSI, Southampton, England, 1985.

[COR88] CORNU-EMIEUX R.
Réseau de Cellules Intégré : Etudes d'Architectures pour des Applications de C.A.O. de V.L.S.I
Thèse de doctorat en micro-électronique, INPG, Grenoble, France, Septembre 1988.

[COT88] COTTRELL M.
Stability and Attractivity in Associative Memory Networks
Biological Cybernetics 58, Springer-Verlag, 1988, pp.129-139.

[DEL87] DELORI H., GUYOT A., PAILLOTIN J.F.
French MPC Activity Report
IMAG, TIM3, Rapport de Recherche 7211, Grenoble, France, Avril 1987.

[DIS88] DISTANTE F., LOMBARDI F., SCUITO D.
Array Partionning : a Methodology for Reconfigurability and Reconfiguration Problems
ICCD'88, Mai 1988, pp.564-567.

[EVA85] EVANS R. A.
A Self -Organising Fault-Tolerant 2 Dimensional Array
VLSI 85 International Conference, Tokyo, Japan, Août 1985.

[FAH87] FAHLMAN S.E., HINTON G.
Connectionist Architectures for Artificial Intelligence
IEEE Computer, 1987, pp.100-108.

[FAU88] FAURE B., MAZARE G.
A VLSI Implementation of Multi-Layered Neural Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[FRA86] FRANZON P. D.
Yield Modeling for Fault Tolerant Arrays
International Workshop on Systolic Arrays, University of Oxford, England, Juillet 1986.

[FRA87] FRANZON P. D., TEWKSBURY S. K.
'Chip Frame' Scheme for Reconfigurable Mesh-Connected Arrays
IFIP Workshop on WSI, Brunel University, England, Septembre 1987.

[FRI85] FRIED J.
An Analysis of Power and Clock Distribution for WSI Systems
International Workshop on WSI, Southampton, England, 1985.

[FUK75] FUKUSHIMA K.
Cognitron : A Self-organizing Multilayered Neural Network
Biological Cybernetics 20, Springer Verlag, 1975, pp.121-136.

[GAR88] GARTH S.
A specialist Parallel Processor for Neural Network Simulations
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[GEN87] GENESTIER P.
Conception de Micropocesseurs à Haut Rendement
Thèse de Docteur en microélectronique, INPG, Grenoble, France, 1987.

[GRA86] GRAF H.P., JACKEL D., HOWARD R.E., STRAUGHN B., DENKER J.S., HUBBARD W., TENNANT D.M., SCHWARTZ D.
VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.182-187.

[GRA87] GRAF H.P., VEGVAR P.
A CMOS Associative Memory Chip Based on Neural Networks
IEEE International Solid State Circuits, Février 1987.

[GRA88] GRAF H.P., JACKEL L.D., HUBBARD W.E.
VLSI Implementation of a Neural Network Model
IEEE Computer, Mars 1988, pp.41-49.

[GRO76] GROSSBERG S.
Adaptative Pattern Classification and Universal Recoding : I. Parallel Development and Coding of Neural Feature Detectors
Biological Cybernetics, 1976, Springer-Verlag, pp.121-134.

[GRZ86] GRZYWACX N.M.,YUILLE A.L.
Motion Correspondence and Analog networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.200-205.

[GUE87] GUERIN A.
Un Calculateur de Réseaux Adaptatifs SYstolique : Application au Calcul Neuromimétique
Thèse de docteur ingénieur en Electronique, INPG, Grenoble, France, Juillet 1987.

[HED82] HEDLUND K. S.
Wafer Scale Integration of Parralel Processors
PhD dissertation, Purdue University, West Laffayette, Novembre 1982.

[HIN85] HINTON G.E., SEJNOWSKI T.J.
Learning in Boltzmann Machines
Cognitiva 85, Paris, France, Juin 1985, pp.283-290.

[HIN86] HINTON G.E., SEJNOVSKI T.J.
Learning in Boltzmann Machines
in D.E. Rumelhart, J.L. McClelland, & the PDP research group (Eds.), Parallel Distributed Processing : Explorations in the microstructure of cognition , Vol.1, Cambridge, MA, USA, Bradford Books, 1986, pp.282-317.

[HIN87] HINTON G.E.
Connectionist Learning Procedures
Technical Report CMU-CS-87-115, Juin 1987.

[HOP82] HOPFIELD J.J.
Neural Networks and Physical systems with emergent collective computational abilities
California Institute of Technology, Pasadena, California, USA, Janvier 1982, pp.2554-2558.

[HOP84] HOPFIELD J.J.
Neurons with graded response have collective computational properties like those of two state neurons
California Institute of Technology, Pasadena, California, USA, Fevrier 1984, pp.3088-3092.

[HOP85] HOPFIELD J.J., TANK D.W.
"Neural" Computation of Decisions in Optimization Problems
Biological Cybernetics, Springer-Verlag 1985, pp.141-152.

[HUB86] HUBBARD W., SCHWARTZ J., DENKER J., GRAF H.P., HOWARD R., JACKEL L., STRAUGHN B., TENNANT D.
Electronic Neural Networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.227-234.

[HUR87] HURAT Ph., SAUCIER G.
Architecture Systolique sur Tranche
Journées Architectures Parallèles du C3, Nice, France, 1987.

[IVE87] IVEY P.A., HUCH M., MIDWINTER T., HURAT Ph., GLESNER M.
Design of a Large SIMD Array in Wafer Scale Technology
IFIP Workshop on Wafer Scale Integration, Brunel University, England, 1987.

[JAY86] JAY C., EYNARD J.P., SAUCIER G.
Test Facilities in a 2-D WSI Processor Array
IFIP Workshop on WSI, Grenoble, France, Mars 1986.

[JEF86] JEFFREY W., ROSNER R.
Neural Networks Processing as a Tool for Function Optimization
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.241-246.

[JOI87] JOINNAULT B., GACHET P.
Conception d'Algorithmes et d'Architectures Systoliques.
Thèse de Docteur d'Université, Rennes, France, 1987.

[JOR86] JORDAN M.I.
An Introduction to Linear Algebra in Parallel Distributed Processing in D.E. Rumelhart, J.L. McClelland, & the PDP research group (Eds.), Parallel Distributed Processing : Explorations in the microstructure of cognition , Vol.1, Cambridge, MA, USA, Bradford Books, 1986, pp.366-443.

[JOS87] JOSIN G.
Neural-Network Heuristics - Three heuristic algorithms that learn from experience
Heuristic algorithms, BYTE, Octobre 1987, pp.183-192.

[KAT86] KATEVENIS M.
Switch Design for Soft-Configurable WSI Systems
IFIP Workshop on WSI, Grenoble, France, Mars 1986.

[KOH85] KOHONEN T.
Representation of Sensory Information in Self-organizing Maps
Cognitiva 85, Paris, Juin 1985, pp.586-591.

[KOH88] KOHONEN T.
A "Neural" Phonetic Type writer
IEEE Computer, Mars 1988, pp.11.

[KOU88] KOUKA E.F.M.
Reconfiguration de Circuits Intégrés sur Tranche : une Approche Expérimentale
CSI-INPG, Rapport de Recherche 1/88, Grenoble, France, Novembre 1988.

[KUN80] KUNG H.T.
The Structure of Parallel Algorithms
Advances in Computers, Vol.19, Academic Press, 1980, pp.65-112.

[KUN82] KUNG H.T.
Why Systolic Architecture ?
IEEE Computer, Janvier 1982, pp.37-46.

[LAT88] LATTARD D., MAZARE G.
Parallel Image Reconstruction by Using a Dedicated Asynchronous Cellular Array
Proceedings of the International Conference on Parallel Processing for Computer Vision and Display, Leeds, England, Janvier 1988.

[LEC87] LECUN Y.
Modèles Connexionistes de l'Apprentissage
Thèse de Doctorat en Informatique, Université Paris 6, France, Juin 1987.

[LEI86] LEIGHTON T., LEISERSON C.
A Survey of Algorithms for Integrated Wafer-Scale Systolic Arrays
MIT/LCS/TM-302, Mai 1986.

[LIN86] LINSKER R.
From Basic Network Principles to Neural Architecture : Emergence of Spatial-opponent Cells
Proc. Natl. Acad. Sci. USA, Vol 83, Octobre 1986, pp.7508-7512.

[LIN88] LINSKER R.
Self Organization in a Perceptual Network
IEEE Computers, Mars 1988, pp.105-117.

[LIP87] LIPPMANN R.P.
An Introduction to Computing with Neural Nets
IEEE ASSP magazine, Avril 1987, pp.4-22.

[LOM87] LOMBARDI F., SAMI M.G., STEFFANELI R.
Reconfiguration of VLSI Arrays : an Index Mapping Approach
IEEE Compeuro 87, Hamburg, RFA, 1987, pp.62-65.

[MIY86] MIYAKE S., FUKUSHIMA K.
A Neural Network Model for the Mechanism of Pattern Information Processing
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.305-308.

[MOO85] MOORE W.R., MAHAT R.
Fault-Tolerant Communications for WSI of a Processor Array
Microelectronics and Reliability, Vol. 25, No 2, 1985, pp.291-294.

[McK86] McKIRDY R., LEA R.
Physical Design Issues for WSI
IFIP Workshop on WSI, Grenoble, France, Mars 1986.

[NEG85] NEGRINI R., STEFFANELLI R.
Algorithms for Self Reconfiguration of Wafer Scale Regular Arrays
Proceedings of International Conference on Circuits and Systems, IEEE, Beijing, Chine, 1985.

[NIC86] NICOLAS G.
Technical and Economical Aspect of laser Repair for WSI Memory
IFIP Workshop on WSI, Grenoble, France, Mars 1986.

[OBJ88] OBJOIS Ph.
Réseau de Cellules Intégré : Mécanisme de Communication Intercellulaire et Application à la Simulation Logique
Thèse de Docteur en Micro-électronique, INPG, Grenoble, France, Septembre 1988.

[PEN86] PENZ A. P., WIGGINS R.
Digital Signal Processor Accelerators for Neural Network Simulations
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.345-355.

[PER85] PERSONNAZ L., GUYON I., RONNET J.C., DREYFUS G.
Character Recognition, Neural Networks and Statistical Physics
Cognitiva 85, Paris, France, Juin 1985.

[PER86a] PERSONNAZ L., GUYON I., DREYFUS G., TOULOUSE G.
A Biologically constrained learning mechanism in networks of formal neurons
Journal of Statistical Physics, Mai 1986.

[PER86b] PERSONNAZ L., GUYON I., DREYFUS G.
Designing a Neural Network Satisfying a Given Set of Constraints
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.356-359.

[PER86c] PERSONNAZ L., GUYON I., JOHANNET A., DREYFUS G., TOULOUSE G.
A Simple Selectionist Learning Rule for Neural Networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.360-363.

[PER86d] PERSONNAZ L., GUYON I., DREYFUS G.
Neural Network Design for Efficient Information Retrieval
Disorded Systems and Biological Organization, Springer 1986.

[PIT87] PITT K.E.G.
WSI Packaging Problems
IFIP Workshop on WSI, Brunel University, England, Septembre 1987.

[QUI84] QUINTON P.
Automatic Synthesis of Systolic Arrays from Recurent Uniform Equations
11th Annual International Symposium on Computer Architecture, Ann Arbor, USA, Juin 1984, pp.208-214.

[ROB88] ROBERT F., WANG S.
Implementation of a Neural Network on a Hypercube F.P.S. T20
IFIP, Pise, Italie, Avril 1988.

[RUC88] RUCKERT U., GOSER K.
VLSI Design of Associative Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[RUE87] RUECKERT U., KREUZER I., GOSER K.
A VLSI Concept for an Adaptative Matrix Based on Neural Networks
COMP EURO, Hamburg, RFA, 1987, pp.31-34.

[RUM86a] RUMELHART D.E., ZISPER D.
Feature Discovery by Competitive Learning
in D.E. Rumelhart, J.L. McClelland, & the PDP research group (Eds.), Parallel Distributed Processing : Explorations in the microstructure of cognition , Vol.1, Cambridge, MA, USA, Bradford Books, 1986, pp.151-193.

[RUM86b] RUMELHART D.E., HINTON G.E., WILLIAMS R.J.
Learning Internal Representations by Error Propagation
in D.E. Rumelhart, J.L. McClelland, & the PDP research group (Eds.), Parallel Distributed Processing : Explorations in the microstructure of cognition , Vol.1, Cambridge, MA, USA, Bradford Books, 1986, pp.318-364.

[SAG86] SAGE J.P., THOMPSON K., WITHERS R.S.
An artificial Neural Network Integrated Circuit Base on MNOS/CCD Principles
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.381-385.

[SAM86] SAMI M. G., STEFFANELLI R.
Reconfigurable Architectures for VLSI Processing Arrays
Proceedings of the IEEE, Vol.74, No.5, Mai 1986, pp.712-722.

[SAY86] SAYLOR J., STORK D.G.
Parallel Analog Neural Networks for Tree Searching
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.392-397.

[SIA85] SIARRY P., BERGONZI L., DREYFUS G.
Optimisation du placement de blocs par la méthode thermodynamique : application à la conception du plan de masse d'un circuit
1er Colloque National sur les réseaux prédiffusés et précaractérisés, Grenoble, France, Mai 1985, pp.342.

[SIA86] SIARRY P.
La méthode du recuit simulé : application à la conception de circuits électroniques
Thèse de Doctorat de Physique, Université Paris 6, France, Novembre 1986.

[SIV86] SIVILOTTI M.A., EMERLING M.R., MEAD C.A.
VLSI Architectures for Implementation of Neural Networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.408-413.

[SPE86] SPENCER E.G.
Programmable Bistable Switches and Resistors for Neural Networks
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.414-419.

[STA86] STAPPER C.
Yield Statistics for Large Area ICs
IEEE International Solid-State Circuits Conference Digest of Technical Papers, Février 1986, pp.168-169.

[STI87] STILES G. S., DENQ D. L.
A Quantitative Comparison of the Performance of Three Distributed Associative Memory Models
IEEE Transactions on Computers, Vol.C-36, No 3, Mars 1987, pp.257-263.

[TAN87] TANK D.W., HOPFIELD J.J.
Collective Computation in Neuronlike Circuits
Scientific American, Vol.257, Number 6, Décembre 1987, pp.62-70.

[TAR88] TARASSENKO L., MURRAY A.
Fully-programmable Analogue VLSI Devices for the Implementation of Neural Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[THA86] THAKOOR A.P., LAMB J.L., MOOPENN A., LAMBE J.
Binary Synaptic Connections Based on Memory Switching in a-Si:H
In J.S Denker (Ed), AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, pp.426-431.

[URQ84] URQUART R.B., WOOD D.
Systolic Matrix and Vector Multiplication Methods for Signal Processing
IEEE Proceedings, Vol.131, Pt.F., No.6, Octobre 1984, pp.623-631.

[VAL86] VAL C.
Wafer Scale Integration Packaging
IFIP Workshop on WSI, Grenoble, France, Mars 1986.

[VER88] VERLEYSEN M., SIRLETTI B., JESPERS P.G.A.
A New CMOS Architecture for Neural Networks
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[VIL88] VILLA A.
Les réseaux de Neurones : Le Point de Vue du Physiologiste
Journée Internationale, "Automates Cellulaires et Neurones Artificiels", LAMI, EPFL, Lausanne, Suisse, Mars 1988.

[WAL87] WALKER D. M. H.
Yield Simulation for Integrated Circuits
Kluwer Academic Publishers, 1987.

[WEI88] WEINFELD M.
A Fully Digital CMOS Integrated Hopfield Neural Network Including the Learning Algorithm
International Workshop on VLSI for Artificial Intelligence, University of Oxford, England, Juillet 1988.

[WID88] WIDROW B., WINTER R.
Neural Nets for Adaptative Filtering and Adaptative Pattern Recognition
IEEE Computers, Mars 1988, pp. 25-39.

Table des matières

Table des illustrations

Chapitre VI. Perspectives 169

Chapitre VII. Intégration sur tranche 181